The Perfect Matching Reconfiguration Problem

Marthe Bonamy, Nicolas Bousquet, Marc Heinrich, Takehiro Ito, Yusuke Kobayashi, Arnaud Mary, Moritz Mühlenthaler, Kunihiro Wasa

The Perfect Matching Reconfiguration Problem

Marthe Bonamy
CNRS, LaBRI, Université de Bordeaux, Talence, France
marthe.bonamy@u-bordeaux.fr

Nicolas Bousquet
CNRS, Laboratoire G-SCOP, Grenoble-INP, Univ. Grenoble-Alpes, Grenoble, France
nicolas.bousquet@grenoble-inp.fr

Marc Heinrich
LIRIS, Université Claude Bernard Lyon 1, Lyon, France
marc.heinrich@univ-lyon1.fr

Takehiro Ito
Graduate School of Information Sciences, Tohoku University, Sendai, Japan
takehiro@ecei.tohoku.ac.jp

Yusuke Kobayashi
Research Institute for Mathematical Sciences, Kyoto University, Kyoto, Japan
yusuke@kurims.kyoto-u.ac.jp

Arnaud Mary
LBBE, Université Claude Bernard Lyon 1, Lyon, France
arnaud.mary@univ-lyon1.fr

Moritz Mühlenthaler
Fakultät für Mathematik, TU Dortmund University, Dortmund, Germany
moritz.muehlenthaler@math.tu-dortmund.de

Kunihiro Wasa
Principles of Informatics Research Division, National Institute of Informatics, Tokyo, Japan
wasa@nii.ac.jp

Abstract

We study the PERFECT MATCHING RECONFIGURATION problem: Given two perfect matchings of a graph, is there a sequence of flip operations that transforms one into the other? Here, a flip operation exchanges the edges in an alternating cycle of length four. We are interested in the complexity of this decision problem from the viewpoint of graph classes. We first prove that the problem is PSPACE-complete even for split graphs and for bipartite graphs of bounded bandwidth with maximum degree five. We then investigate polynomial-time solvable cases. Specifically, we prove that the problem is solvable in polynomial time for strongly orderable graphs (that include interval graphs and strongly chordal graphs), for outerplanar graphs, and for cographs (also known as P_4-free graphs). Furthermore, for each yes-instance from these graph classes, we show that a linear number of flip operations is sufficient and we can exhibit a corresponding sequence of flip operations in polynomial time.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Combinatorial Reconfiguration, Graph Algorithms, Perfect Matching

Digital Object Identifier 10.4230/LIPIcs.MFCS.2019.80

Funding *Marthe Bonamy*: Partially supported by ANR project GrR (ANR-18-CE40-0032).
Nicolas Bousquet: Partially supported by ANR project GrR (ANR-18-CE40-0032).
Marc Heinrich: Partially supported by ANR project GrR (ANR-18-CE40-0032).
Takehiro Ito: Partially supported by JST CREST Grant Number JPMJCR1402, and JSPS KAKENHI Grant Numbers JP18H04091 and JP19K11814, Japan.
Yusuke Kobayashi: Supported by JST ACT-I Grant Number JPMJPR17UB, and JSPS KAKENHI Grant Numbers JP16K16010, JP16H03118, JP17K19960 and JP18H05291, Japan.
Arnaud Mary: Partially supported by ANR project GrR (ANR-18-CE40-0032).
Kunihiro Wasa: Partially supported by JST CREST Grant Number JPMJCR1401 and JPMJCR18K3, and JSPS KAKENHI Grant Number JP19K20350, Japan.

Acknowledgements This work is partially supported by JSPS and MAEDI under the Japan-France Integrated Action Program (SAKURA).

1 Introduction

Given an instance of some combinatorial search problem and two of its feasible solutions, a *reconfiguration problem* asks whether one solution can be transformed into the other in a step-by-step fashion, such that each intermediate solution is also feasible. Reconfiguration problems capture dynamic situations, where some solution is in place and we would like to move to a desired alternative solution without becoming infeasible. A systematic study of the complexity of reconfiguration problems was initiated in [21]. Recently the topic has gained a lot of attention in the context of constraint satisfaction problems and graph problems, such as the independent set problem, the matching problem, and the dominating set problem. Reconfiguration problems naturally arise for operational research problems but also are closely related to uniform sampling (using Markov chains) or enumeration of solutions of a problem. For an overview of recent results on reconfiguration problems, the reader is referred to the surveys of van den Heuvel [17] and Nishimura [27].

In order to define valid step-by-step transformations, an adjacency relation on the set of feasible solutions is needed. Depending on the problem, there may be different natural choices of adjacency relations. For instance, we may assume that two matchings of a graph are adjacent if one can be obtained from the other by exchanging precisely one edge, i.e., there exist $e \in M$ and $f \in M'$ such that $M \setminus \{e\} = M' \setminus \{f\}$. The corresponding modification of a matching is usually referred to as *token jumping* (TJ). Here, the *tokens* are placed on the edges of a matching and a token may be "moved" from an edge of the matching to another edge so that we obtain another matching. There is another similar adjacency relation, where two matchings are adjacent if one can be obtained from the other by moving a token to some incident edge. This adjacency relation is called *token sliding* (TS). Ito et al. [21] gave a polynomial-time algorithm that decides if there is a transformation between two given matchings under the TJ and TS operations.

1.1 The perfect matching reconfiguration problem

Recall that a matching of a graph is *perfect* if it covers each vertex. We study the complexity of deciding if there is a step-by-step transformation between two given perfect matchings of a graph. However, according to the adjacency relations given by the TS and TJ operations, there is no transformation between any two distinct perfect matchings of a graph. Since the symmetric difference of any two perfect matchings of a graph consists of even-length disjoint cycles, it is natural to consider a different adjacency relation for perfect matchings. We say that two perfect matchings of a graph differ by a *flip* (or *swap*) if their symmetric difference induces a cycle of length four. We consider two perfect matchings to be *adjacent* if they differ by a flip. Intuitively, for two adjacent perfect matchings M and M', we think of a flip as an operation that exchanges edges in $M \setminus M'$ for edges in $M' \setminus M$.

An example of a transformation between two perfect matchings of a graph is given in Figure 1. We formalize the task of deciding the existence of a transformation between two given perfect matchings as follows.

Perfect Matching Reconfiguration
Input: Graph G, perfect matchings M_s and M_t of G.
Question: Is there a sequence of flips that transforms M_s into M_t?

We take the flip operation on a cycle of length four as the adjacency relation in this paper, because a flip is in some sense a minimal modification of a perfect matching. Note that if we do not restrict the length of a cycle in the definition of a flip, then for any two perfect

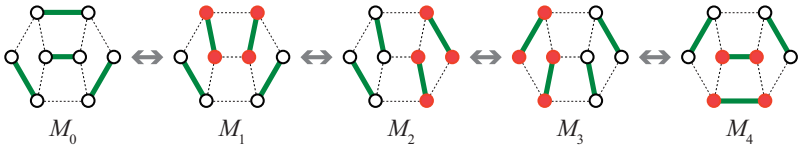


Figure 1 A transformation between perfect matchings M_0 and M_4 under the flip operation. For each i, $1 \leq i \leq 4$, the matching M_i can be obtained from M_{i-1} by applying the flip operation to the cycle induced by the four painted (red) vertices in M_i.

matchings M_s and M_t of a graph, there always exists a sequence of flip operations that transforms M_s into M_t, since we can perform a flip on each (disjoint) cycle in the symmetric difference of M_s and M_t. As a compromise, we may extend our problem definition to flips on cycles of fixed constant length ℓ, where $\ell \geq 4$ and ℓ is even. We refer to the corresponding flip operation as the ℓ-flip operation, and the corresponding reconfiguration problem as Perfect Matching ℓ-Flip Reconfiguration. It should be noted that the ℓ-flip operation must be applied to a cycle of length *exactly* ℓ, and hence there is no guarantee of the existence of a transformation.

1.2 Related work

Transformation of matchings has been considered under several types of flip operations for generating random matchings. Under the TS and TJ operations, numerous algorithms and hardness results are available for finding transformations between matchings, more generally, between independent sets. Furthermore, similar types of flip operations are well known for stable matchings and some geometric matching problems related to finding transformations between triangulations. More directly, a restriction of Perfect Matching (4-Flip) Reconfiguration to grid graphs has been considered before in the setting of domino tilings. In this restricted setting, Saldanha et al. [28] gave a criterion for the existence of a transformation between two tilings (which correspond to perfect matchings of a grid graph) and a formula for their distance of a transformation.

Sampling random perfect matchings

The problem of sampling or enumerating perfect matchings in a graph received considerable attention (see, e.g., [31]). Determining the connectivity and the diameter of the solution space formed by perfect matchings under the flip operation provide some information on the ergodicity or the mixing time of the underlying Markov chain. Indeed, the connectivity of the chain ensures the irreducibility (and usually the ergodicity) of the underlying Markov chain. Additionally, the diameter of the solution space provides a lower bound on the mixing time of the chain.

The use of flips for sampling random perfect matchings was first started in [8] where it is seen as a generalization of transpositions for permutations. Their work was later improved and generalized in [13] and [12]. The focus of these last two articles is to investigate the problem of sampling random perfect matchings using a Markov Chain called the switch chain. Starting from an arbitrary perfect matching, the chain proceeds by applying at each step a random 4-flip operation (called switch in these papers). The aim of these papers is to characterize classes of graphs for which simulating this chain for a polynomial number of steps is enough to generate a perfect matching close to uniformly distributed. Some of their results can be reformulated in the reconfiguration terminology. In [13], it is proved that the largest

hereditary class of bipartite graphs for which the solution space formed by perfect matchings under the 4-flip operation is connected is the class of chordal bipartite graphs. This result is generalized in [12] where they characterize the hereditary class of general (non-bipartite) graphs for which the solution space is connected. They call this class Switchable. Note that it is not clear whether graphs in this class can be recognized in polynomial time. The question of the complexity of Perfect Matching (4-Flip) Reconfiguration is also mentioned in [12].

Reconfiguration of matchings and independent sets

Recall that matchings of a graph correspond to independent sets of its line graph. Although reconfiguration of independent sets received a considerable attention in the last decade (e.g., [5, 6, 7, 16, 22, 23, 33]), all the known results for reconfiguration of independent sets are based on the TJ or TS operations as adjacency relations. Thus, none of these results carry over to the Perfect Matching Reconfiguration problem.

A related problem can be found in a more general setting: The problem of determining, enumerating, or randomly generating graphs with a fixed degree sequence has received a considerable attention since the fifties (see, e.g., [30, 15, 32]). Given two graphs with a fixed degree sequence, one might want to know if it is possible to transform the one into the other via a sequence of 4-flip operations and if yes, how many steps are needed for such a transformation; note that the host graph (i.e., the graph G in our problem) is a clique in this setting. Hakimi [15] proved that such a transformation always exists. Will [32] proved that the problem of finding a shortest transformation is NP-complete, and Bereg and Ito [2] provide a $\frac{3}{2}$-approximation algorithm for this problem.

Stable matchings

Suppose we are given a bipartite graph and for each vertex a linear preference order of its neighbors. A matching M is not stable if there is an edge vw not in M such that v prefers w and w prefers v to their respective partners in M. The well-known algorithm by Gale and Shapley yields a stable matching in polynomial time [14]. It is known that any two stable matchings cover the same vertices, so the stable matchings are perfect matchings of some subgraph. Furthermore, they form a distributive lattice under *rotations* on preference-oriented cycles, see for example [14]. Essentially, the symmetric difference of two stable matchings consists of disjoint cycles (of several lengths) and we may exchange edges on these cycles to obtain another stable matching. If we drop the preferences, then the question is simply if we can find a transformation between two perfect matchings by exchanging edges on cycles in the symmetric difference. Clearly the answer is always yes, for example by processing the cycles in the symmetric difference one by one. We consider a similar setting, but restrict the length of the cycles.

Diagonal-flips of triangulations

A diagonal-flip of a triangulation in geometry is similar to our 4-flip operation in the sense that we switch between two states of a quadrilateral. In the context of triangulations, a diagonal-flip operation switches the diagonal of a quadrilateral. Transformations between triangulations of point sets and polygons have been studied mostly in the plane. It is known that the solution space formed by triangulations of point sets and polygons in the plane is connected and has diameter $O(n^2)$, where n is the number of points [20, 24]. Recently, NP-completeness has been proved for deciding the distance in the solution space between triangulations of a point set in the plane [25] and triangulations of a simple polygon [1].

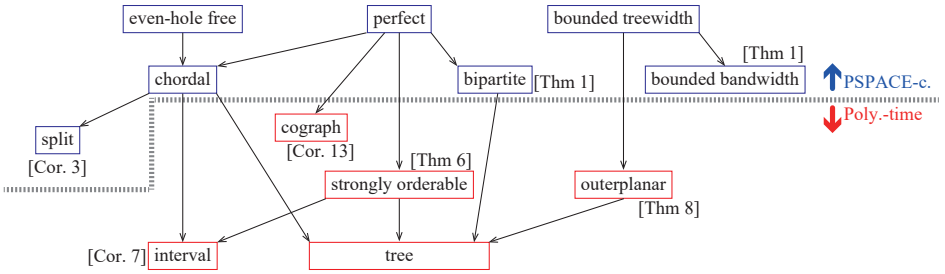


Figure 2 Our results, where each arrow represents the inclusion relationship between graph classes: $A \to B$ represents that the graph class B is properly included in the graph class A.

Houle et al. [18] have considered triangulations of point sets in the plane that admit a perfect matching. They show that any two such triangulations are connected under the diagonal-flip operation. For this purpose they consider the graph of non-crossing perfect matchings, where two matchings are adjacent if they differ by a single non-crossing cycle (of arbitrary length). They show that the graph of non-crossing perfect matchings is connected and conclude from this that any two triangulations that admit a perfect matching must be connected. In contrast to their setting, we remove all geometric requirements, but restrict the length of the cycles allowed for our flip operation.

1.3 Our results

In this paper, we study the complexity of Perfect Matching Reconfiguration from the viewpoint of graph classes. Figure 2 summarizes our results.

Recall that reconfiguration of matchings under the TS and TJ operations can be solved in polynomial time for any graph [21]. In contrast, we prove that Perfect Matching Reconfiguration is PSPACE-complete, even for split graphs, and for bipartite graphs of bounded bandwidth and of maximum degree five. We note that our hardness result for bipartite graphs gives contrast to chordal bipartite graphs for which there always exists a transformation between any two perfect matchings [13]. In addition, we extend our hardness result to a more general setting, namely the reconfiguration of k-factor subgraphs under the ℓ-flip operation for any fixed $k \geq 1$ and any fixed even integer $\ell \geq 4$.

We then investigate polynomial-time solvable cases. We prove that Perfect Matching Reconfiguration admits a polynomial-time algorithm on strongly orderable graphs (these include interval graphs and strongly chordal graphs), outerplanar graphs, and cographs (also known as P_4-free graphs). More specifically, we give the following results:

- For strongly orderable graphs, a transformation between two perfect matchings always exists; hence the answer is always yes. Furthermore, there is a transformation of linear length (i.e., a linear number of flip operations) between any two perfect matchings and such a transformation can be found in polynomial time.
- Perfect Matching Reconfiguration on outerplanar graphs can be solved in linear time, and we can find a transformation of linear length for a yes-instance in linear time. (Note that there are no-instances, e.g., long cycles).
- Perfect Matching Reconfiguration on cographs can be solved in polynomial time, and we can find a transformation of linear length for a yes-instance in polynomial time. (Again, there are no-instances).

Due to the page limitation, we omit proofs of the claims marked with $(*)$.

1.4 Notation

For standard definitions and notations on graphs, we refer the reader to [9]. Let $G = (V, E)$ be a simple graph. We sometimes denote by $V(G)$ and $E(G)$ the vertex set and edge set of G, respectively. A *matching* $M \subseteq E$ of G is a set of edges that share no endpoint. A vertex v is *covered* by a matching M if v is incident to an edge in M. For a vertex set $V' \subseteq V$, we denote by $G[V']$ the subgraph of G induced by V'. For a vertex set $W \subseteq V$, let $G - W := G[V \setminus W]$. For a vertex $v \in V$, we denote by $N(v)$ the *neighborhood* of v, that is, $N(v) := \{w \in V \mid vw \in E\}$.

Two perfect matchings M and M' of G are *adjacent* if their symmetric difference $M \triangle M'$ induces a cycle of length four. A sequence $M_0, M_1, \ldots, M_q$ of perfect matchings in G is called a *reconfiguration sequence between M and M'* if $M_0 = M$, $M_q = M'$, and M_{i-1} and M_i are adjacent for each i, $1 \leq i \leq q$. Given two perfect matchings M_s and M_t of a graph G, Perfect Matching Reconfiguration is to determine whether there exists a reconfiguration sequence between M_s and M_t. We denote by a triple $(G, M_\mathrm{s}, M_\mathrm{t})$ an instance of the problem.

2 PSPACE-completeness

In this section, we prove that perfect matching reconfiguration is PSPACE-complete. Interestingly, the problem remains intractable even for bipartite graphs, even though matchings in bipartite graphs satisfy several nice properties.

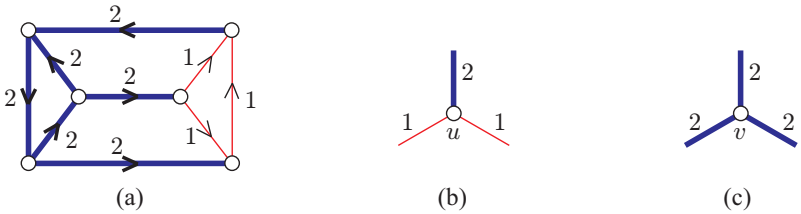
▶ **Theorem 1.** Perfect matching reconfiguration *is* PSPACE*-complete for bipartite graphs whose maximum degree is five and whose bandwidth is bounded by a fixed constant.*

Proof. Observe that the problem can be solved in (most conveniently, nondeterministic [29]) polynomial space, and hence it is in PSPACE. As a proof of Theorem 1, we thus prove that the problem is PSPACE-hard for such graphs, by giving a polynomial-time reduction from the Nondeterministic Constraint Logic problem (NCL for short) [16].

Definition of nondeterministic constraint logic

An NCL "machine" is an undirected graph together with an assignment of weights from $\{1, 2\}$ to each edge of the graph. An (*NCL*) *configuration* of this machine is an orientation (direction) of the edges such that the sum of weights of in-coming arcs at each vertex is at least two. Figure 3(a) illustrates a configuration of an NCL machine, where each weight-2 edge is depicted by a (blue) thick line and each weight-1 edge by a (red) thin line. Then, two NCL configurations are *adjacent* if they differ in a single edge direction. Given an NCL machine and its two configurations, it is known to be PSPACE-complete to determine whether there exists a sequence of adjacent NCL configurations which transforms one into the other [16].

An NCL machine is called an AND/OR *constraint graph* if it consists of only two types of vertices, called "NCL AND vertices" and "NCL OR vertices" defined as follows: A vertex of degree three is called an *NCL AND vertex* if its three incident edges have weights 1, 1, and 2. (See Figure 3(b).) An NCL AND vertex u behaves as a logical AND, in the following sense: the weight-2 edge can be directed outward for u only if both two weight-1 edges are directed inward for u. Note that, however, the weight-2 edge is not necessarily directed outward even when both weight-1 edges are directed inward. A vertex of degree three is called an *NCL OR vertex* if its three incident edges have weights 2, 2, and 2. (See Figure 3(c).) An NCL



(a) (b) (c)

Figure 3 (a) A configuration of an NCL machine, (b) an NCL AND vertex u, and (c) an NCL OR vertex v.

OR vertex v behaves as a logical OR: one of the three edges can be directed outward for v if and only if at least one of the other two edges is directed inward for v. It should be noted that, although it is natural to think of NCL AND/OR vertices as having inputs and outputs, there is nothing enforcing this interpretation; especially for NCL OR vertices, the choice of input and output is entirely arbitrary because an NCL OR vertex is symmetric. For example, the NCL machine in Figure 3(a) is an AND/OR constraint graph. From now on, we call an AND/OR constraint graph simply an *NCL machine*, and call an edge in an NCL machine an *NCL edge*. NCL remains PSPACE-complete even if an input NCL machine is planar, bounded bandwidth, and of maximum degree three [34].

Gadgets and reduction

Suppose that we are given an instance of NCL, that is, an NCL machine and two configurations of the machine. We will replace each of the NCL edges and NCL AND/OR vertices with its corresponding gadget; if an NCL edge e is incident to an NCL vertex v, then we connect the corresponding gadgets for e and v by a pair of vertices, called *connectors* (*between v and e*) or (v, e)-*connectors*, as illustrated in Figure 4(a) and (b). Thus, each edge gadget has two pairs of connectors, and each AND/OR gadget has three pairs of connectors. Our gadgets are all edge-disjoint, and share only connectors.

Figure 5 shows our three types of gadgets which correspond to NCL edges and NCL AND/OR vertices. As illustrated in Figure 4, we replace each of the NCL edges and NCL AND/OR vertices with its corresponding gadget; let G be the resulting graph. Notice that each of our three gadgets is of maximum degree three, and connectors in the edge gadget are of degree two; thus, G is of maximum degree five. In addition, each of our three gadgets is a bipartite graph such that two connectors in the same pair belong to different sides of the bipartition; therefore, G is bipartite. Furthermore, since NCL remains PSPACE-complete even if an input NCL machine is bounded bandwidth [34], the resulting graph G is also bounded bandwidth and of maximum degree five; notice that, since each gadget consists of only a constant number of edges, the bandwidth of G is also bounded.

We next construct two perfect matchings of G which correspond to two given NCL configurations C_{s} and C_{t} of the NCL machine. In our reduction, we construct the correspondence between orientations of an NCL machine and perfect matchings of the corresponding graph, as follows: We regard that the orientation of an NCL edge $e = vw$ is inward direction for v if the two (v, e)-connectors are both covered by (edges in) the AND/OR gadget for v. On the other hand, we regard that the orientation of $e = vw$ is outward direction for w if the two (w, e)-connectors are both covered by the edge gadget for e. To achieve this correspondence, our gadgets are constructed so that both two (v, e)-connectors are always covered by exactly one of the gadgets for v and e. Note that there are (in general, exponentially) many perfect matchings which correspond to the same NCL configuration. However, by the construction of

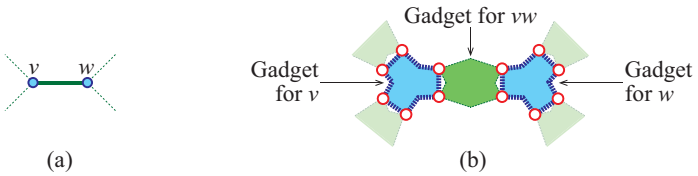


Figure 4 (a) An NCL edge vw, and (b) its corresponding gadgets, where the connectors are depicted by (red) circles.

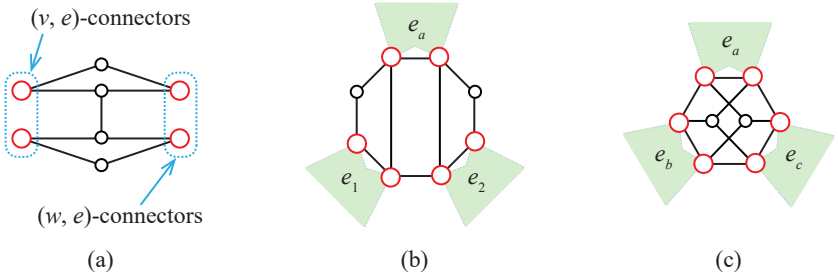


Figure 5 Illustrations of (a) an edge gadget for an NCL edge $e = vw$, (b) an AND gadget, and (c) an OR gadget. In the AND/OR gadget, the three light green parts represent the edge gadgets corresponding to the edges incident to the NCL vertex; e_1 and e_2 in the AND gadget correspond to weight-1 edges.

the three gadgets, no two distinct NCL configurations correspond to the same perfect matching of G. We arbitrarily choose two perfect matchings M_s and M_t of G which correspond to C_s and C_t, respectively.

This completes the construction of our corresponding instance of PERFECT MATCHING RECONFIGURATION. The construction can be done in polynomial time. Furthermore, the following lemma gives the correctness of our reduction.

▶ **Lemma 2** (∗). *There exists a desired sequence of NCL configurations between C_s and C_t if and only if there exists a reconfiguration sequence between M_s and M_t.*

This completes the proof of Theorem 1. ◀

Remarks

We conclude this section by giving some remarks that can be obtained from Theorem 1. We first prove that the problem remains intractable even for split graphs. A graph is *split* if its vertex set can be partitioned into a clique and an independent set.

▶ **Corollary 3.** PERFECT MATCHING RECONFIGURATION *is* PSPACE*-complete for split graphs.*

Proof. By Theorem 1 the problem remains PSPACE-complete for bipartite graphs. Consider the graph obtained by adding new edges so that one side of the bipartition forms a clique. The resulting graph is a split graph. We claim that these new edges can never be part of any perfect matching of the graph. Indeed, since the original graph was bipartite, there must be the same number of vertices on each side of the bipartition. In a perfect matching of the split graph, all the vertices from the independent set must be matched with vertices from the clique, and no vertex from the clique remains to be matched together. Thus, the claim holds, and hence the corollary follows. ◀

We finally extend Theorem 1 into two directions: regular spanning subgraphs and flip operations on alternating cycles of a fixed length. Let k be a positive integer. A spanning subgraph H of a graph G is a *k-factor* if all the vertices in H have degree exactly k. Thus, a 1-factor of G is a perfect matching of G. Based on the proof of Theorem 1, we can obtain the following theorem.

▶ **Theorem 4** (∗). *Let $k \geq 1$ be a fixed integer, and let $\ell \geq 4$ be a fixed even integer. Given two k-factors H_s and H_t of a graph G, it is* PSPACE*-complete to decide if there is a sequence of ℓ-flip operations transforming H_s into H_t.*

3 Polynomial-time algorithms

In this section, we investigate the polynomial-time solvability of Perfect Matching Reconfiguration from the viewpoint of graph classes. We first give the following lemma, which holds for any graph.

▶ **Lemma 5.** *It suffices to solve* Perfect Matching Reconfiguration *for 2-connected graphs having at least four vertices.*

Proof. Let (G, M_s, M_t) be a given instance of Perfect Matching Reconfiguration. If G is not connected, then we can simply consider each connected component separately.

Since the input graph $G = (V, E)$ has a perfect matching, it has an even number of vertices. If $|V| = 2$, then it must hold that $M_s = M_t = E$, and hence the instance is trivially a yes-instance. Therefore, it suffices to solve the problem for $|V| \geq 4$.

Suppose that G is not 2-connected and has a cut vertex $v \in V$, that is, $G - \{v\}$ consists of more than one connected component. Since $|V|$ is even, there exists a vertex subset $X \subseteq V \setminus \{v\}$ such that $|X|$ is odd and $G[X]$ forms a connected component of $G - \{v\}$. Then, any perfect matching in G contains an edge connecting v and X. This shows that we can consider two subgraphs $G_1 := G[X \cup \{v\}]$ and $G_2 := G - (X \cup \{v\})$, separately. That is, we output "yes" if $(G_i, M_s \cap E_i, M_t \cap E_i)$ is a yes-instance for every $i \in \{1, 2\}$, where E_i is the edge set of G_i, and output "no" otherwise. Thus, the lemma follows. ◀

3.1 Strongly orderable graphs

Interval graphs form easy instances for many NP-hard problems, and the situation is no different here. In fact, we prove that any instance on an interval graph is a yes-instance. Our argument also yields a linear-time algorithm to compute a reconfiguration sequence of a linear number of flip operations between any two perfect matchings.

For the sake of generality, we consider a wider class of graphs, called strongly orderable graphs. A graph $G = (V, E)$ is *strongly orderable* if there is a *strong ordering* on its vertices, defined as follows: an order $(v_1, v_2, \ldots, v_n)$ of V such that for every i, j, k, ℓ with $i < j$ and $k < \ell$, if all of v_iv_k, v_iv_ℓ and v_jv_k are edges, then v_jv_ℓ is an edge. Note that the class of strongly orderable graphs is hereditary: every induced subgraph of a strongly orderable graph is strongly orderable.

Our proof strategy for the following theorem is to show that every perfect matching N of a strongly orderable graph G can be transformed into some particular perfect matching M of G, called the canonical perfect matching; then, any two perfect matchings N and N' of G admit a reconfiguration sequence between them via M. The *canonical perfect matching* of a graph G with respect to an order $\mathcal{O} = (v_1, v_2, \ldots, v_n)$ is a perfect matching of G (if any) greedily obtained by selecting, among the available edges, the one with endpoints of

smallest indices. Note that any strongly orderable graph that admits a perfect matching, also admits a canonical perfect matching with respect to a corresponding order on the vertices (see, e.g., [10]). We give the following theorem in this subsection.

▶ **Theorem 6** (∗). *Let G be a strongly orderable graph. Then, there is a reconfiguration sequence of linear length between any two perfect matchings of G. Furthermore, such a reconfiguration sequence can be found in linear time if we are given a strong ordering on the vertices of G as a part of the input.*

The natural question regarding Theorem 6 is whether a strong ordering can be computed efficiently. In general, Dragan [11] proved that strongly orderable graphs $G = (V, E)$ can be recognized in $O(|V| \cdot (|V| + |E|))$ time, and if so we can obtain its strong ordering in the same running time. However, when restricted to interval graphs, we can obtain a strong ordering in linear time [19]. We thus have the following corollary.

▶ **Corollary 7.** *Let G be an interval graph. Then, there is a reconfiguration sequence of linear length between any two perfect matchings of G. Furthermore, such a reconfiguration sequence can be found in linear time.*

3.2 Outerplanar graphs

In this subsection, we consider outerplanar graphs. Note that there are no-instances for outerplanar graphs, e.g., induced cycles of even length more than four. Nonetheless, we give the following theorem.

▶ **Theorem 8.** Perfect Matching Reconfiguration *can be solved in linear time for outerplanar graphs. Moreover, for a* yes*-instance, a reconfiguration sequence of linear length can be output in linear time.*

We give such an algorithm as a proof of Theorem 8. Suppose we are given a simple outerplanar graph $G = (V, E)$, and two perfect matchings M_{s} and M_{t} in G. By Lemma 5 we assume without loss of generality that G is 2-connected and $|V| \geq 4$. Then, G has a planar embedding such that the outer face boundary is a simple cycle and all the vertices of G are on the outer face boundary. Suppose that the vertices $v_1, v_2, \ldots, v_n$ appear in this order along the cycle. For notational convenience, we denote $v_{n+1} = v_1$, $v_{n+2} = v_2$, and $v_0 = v_n$. We first give the following assumption without loss of generality.

▶ **Lemma 9.** Perfect Matching Reconfiguration *for outerplanar graphs can be reduced to the case when $v_i v_j \notin E$ holds for any pair of indices $i, j \in \{1, 2, \ldots, n\}$ such that $|i - j|$ is even. In particular, $v_i v_{i+2} \notin E$ holds for any $i \in \{1, 2, \ldots, n\}$.*

Proof. Suppose that there exists a pair of indices $i, j \in \{1, 2, \ldots, n\}$ such that $|i - j|$ is even and $v_i v_j \in E$. Then, the edge $v_i v_j$ cannot be contained in any perfect matching of G, because $G[\{v_{i+1}, v_{i+2}, \ldots, v_{j-1}\}]$ forms a connected component in $G - \{v_i, v_j\}$ even though it contains an odd number of vertices. Therefore, we can remove $v_i v_j$ from G. ◀

We now show the following lemma for an outerplanar graph $G = (V, E)$.

▶ **Lemma 10** (∗). *If $v_i v_{i+2} \notin E$ for any $i \in \{1, 2, \ldots, n\}$, then there exists an index $k \in \{1, 2, \ldots, n\}$ such that both v_k and v_{k+1} are of degree two.*

Let $k \in \{1, 2, \ldots, n\}$ be an index such that both v_k and v_{k+1} are of degree two, and let $e_i := v_i v_{i+1}$ for each $i \in \{k - 1, k, k + 1\}$. Note that if a perfect matching of G does not contain $e_k = v_k v_{k+1}$, then it has to contain both $e_{k-1} = v_{k-1} v_k$ and $e_{k+1} = v_{k+1} v_{k+2}$. We consider the following two cases separately.

Case 1: We first consider the case with $v_{k-1}v_{k+2} \notin E$. In this case, we can see that e_k is not contained in any cycles of length four, and hence e_k is never touched by any flip operation. Thus, we consider one of the following three sub-cases.

- If $e_k \in M_s \triangle M_t$, then we can immediately conclude that (G, M_s, M_t) is a no-instance.
- If $e_k \in M_s \cap M_t$, then we solve the smaller instance $(G - \{v_k, v_{k+1}\}, M_s \setminus \{e_k\}, M_t \setminus \{e_k\})$.
- If $e_k \notin M_s \cup M_t$, then we solve the smaller instance $(G - \{v_{k-1}, v_k, v_{k+1}, v_{k+2}\}, M_s \setminus \{e_{k-1}, e_{k+1}\}, M_t \setminus \{e_{k-1}, e_{k+1}\})$.

Case 2: We next consider the case with $v_{k-1}v_{k+2} \in E$. Then, $G[\{v_{k-1}, v_k, v_{k+1}, v_{k+2}\}]$ forms a cycle of length four. For each $i \in \{s, t\}$, we define

$$M'_i := \begin{cases} M_i \setminus \{e_k\} & \text{if } e_k \in M_i, \\ (M_i \setminus \{e_{k-1}, e_{k+1}\}) \cup \{v_{k-1}v_{k+2}\} & \text{otherwise.} \end{cases}$$

Let $G' = G - \{v_k, v_{k+1}\}$, and we solve the smaller instance (G', M'_s, M'_t).

In either case, we can reduce the original instance (G, M_s, M_t) to a smaller instance, which implies that our algorithm runs in polynomial time. Indeed, we can implement the above arguments so that the algorithm runs in linear time. (The details are omitted.) The correctness of Case 1 is obvious, while the correctness of Case 2 is guaranteed as follows.

▶ **Lemma 11** (∗). *(G, M_s, M_t) is a* yes*-instance if and only if (G', M'_s, M'_t) is a* yes*-instance.*

This completes the proof of Theorem 8. ◀

3.3 Cographs

We consider cographs in this subsection. Cographs, also known as P_4-free graphs, are graphs without a path on four vertices as an induced subgraph. As examples concerning reconfiguration on this class of graphs, it is known that the problems Independent Set Reconfiguration and Steiner Tree Reconfiguration can be solved in polynomial time for cographs [3, 4, 26], while they are PSPACE-complete for general graphs [21, 26]. We will show that the situation is similar for Perfect Matching Reconfiguration.

To describe our algorithm for cographs, we generalize our problem to non-perfect matchings. In the generalized problem, we regard that two matchings are *adjacent* if their symmetric difference is either a cycle of length four, or a path on three vertices. Note that any two adjacent matchings have the same size. Then, the generalized problem is defined as follows:

General Matching Reconfiguration
Input: Graph G, two matchings M_s and M_t of G.
Question: Is there a sequence of adjacent matchings that transforms M_s into M_t?

An instance of the generalized problem is also denoted by a triple (G, M_s, M_t), and a sequence of adjacent matchings is also called a *reconfiguration sequence*. (G, M_s, M_t) is clearly a no-instance if $|M_s| \neq |M_t|$, and hence we assume that $|M_s| = |M_t|$ holds.

Our main result of this subsection is the following theorem.

▶ **Theorem 12** (∗). General Matching Reconfiguration *can be solved in polynomial time for cographs. Moreover, for a* yes*-instance, a reconfiguration sequence of linear length can be output in polynomial time.*

We note that when two input matchings are perfect, every connected component in their symmetric difference is a cycle. Therefore, even in the generalized problem, two adjacent perfect matchings differ by a flip operation on a cycle of length four. We thus obtain the following result as a corollary of Theorem 12.

▶ **Corollary 13.** Perfect Matching Reconfiguration *can be solved in polynomial time for cographs. Moreover, for a* yes*-instance, a reconfiguration sequence of linear length can be output in polynomial time.*

As a proof of Theorem 12, we give such an algorithm in this subsection. We will use the recursive characterization of cographs. For two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, their *disjoint union* $G_1 \cup G_2$ is the graph such that $V(G_1 \cup G_2) = V_1 \cup V_2$ and $E(G_1 \cup G_2) = E_1 \cup E_2$, while their *complete join* $G_1 \vee G_2$ is the graph such that $V(G_1 \vee G_2) = V_1 \cup V_2$ and $E(G_1 \vee G_2) = E_1 \cup E_2 \cup \{vw \mid v \in V_1, w \in V_2\}$. Then, a *cograph* can be recursively defined, as follows:

- a graph consisting of a single vertex is a cograph;
- if G_1 and G_2 are cographs, then their disjoint union $G_1 \cup G_2$ is a cograph; and
- if G_1 and G_2 are cographs, then their complete join $G_1 \vee G_2$ is a cograph.

Let (G, M_s, M_t) be a given instance of General Matching Reconfiguration such that $G = (V, E)$ is a cograph. By Lemma 5 we assume without loss of generality that G is connected and $|V| \geq 4$, and hence $G = G_1 \vee G_2$ for two cographs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Assume that $|V_1| \geq |V_2|$, and let $k := |M_s| = |M_t|$. We first give a sufficient condition for which there exists a reconfiguration sequence between M_s and M_t.

▶ **Lemma 14** (∗)**.** *There is a reconfiguration sequence between M_s and M_t if G has a matching M of size k such that at least one of the following two conditions holds*:
(C1) *$M \cap E_2 \neq \emptyset$; and*
(C2) *at least one vertex of G_2 is not covered by M.*
Furthermore, a reconfiguration sequence of linear length can be output in polynomial time.

We claim that this sufficient condition can be checked in polynomial time. The existence of a matching satisfying the condition **(C1)** can be checked as follows: For each edge $vw \in E_2$, we check if the graph $G - \{v, w\}$ has a matching of size $k - 1$, or not. This can be done in polynomial time since a maximum matching in a graph can be computed in polynomial time. Similarly, the condition **(C2)** can be checked in polynomial time, as follows: For each vertex $v \in V_2$, we check if the graph $G - \{v\}$ has a matching of size k, or not.

We then consider the case where the sufficient condition of Lemma 14 does not hold. Recall that $G = G_1 \vee G_2$ and $|V_1| \geq |V_2|$.

▶ **Lemma 15** (∗)**.** *Suppose that G does not have a matching of size k satisfying the conditions* **(C1)** *or* **(C2)** *of Lemma 14. Then, the following two claims hold.*

- *(G, M_s, M_t) is a* yes*-instance if and only if $(G_1, M_s \cap E_1, M_t \cap E_1)$ is a* yes*-instance.*
- *For a* yes*-instance (G, M_s, M_t), a reconfiguration sequence between M_s and M_t of linear length can be output in polynomial time.*

The above arguments can be implemented so that the algorithm runs in polynomial time, since we reduce the original instance (G, M_s, M_t) to a smaller instance $(G_1, M_s \cap E_1, M_t \cap E_1)$.

This completes the proof of Theorem 12. ◀

4 Conclusion

We introduced the Perfect Matching Reconfiguration problem and analyzed its complexity from the viewpoint of graph classes. We showed that this problem is PSPACE-complete on split graphs and bipartite graphs of bounded bandwidth and maximum degree five.

Furthermore, we gave polynomial-time algorithms for strongly orderable graphs, outerplanar graphs, and cographs. Each of the algorithms outputs a reconfiguration sequence of linear length in polynomial time.

A natural open question is on which graph classes a *shortest* reconfiguration sequence can be found in polynomial time. Furthermore, it would be interesting to investigate if the flip operation can be used in order to sample perfect matchings uniformly.

References

1 Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz. Flip distance between triangulations of a simple polygon is NP-complete. *Discrete & Computational Geometry*, 54(2):368–389, 2015.

2 Sergey Bereg and Hiro Ito. Transforming Graphs with the Same Graphic Sequence. *Journal of Information Processing*, 25:627–633, 2017.

3 Marthe Bonamy and Nicolas Bousquet. Reconfiguring independent sets in cographs. *arXiv preprint arXiv:1406.1433*, 2014.

4 Paul Bonsma. Independent set reconfiguration in cographs and their generalizations. *Journal of Graph Theory*, 83(2):164–195, 2016.

5 Paul Bonsma, Marcin Kamiński, and Marcin Wrochna. Reconfiguring Independent Sets in Claw-Free Graphs. In *Proceedings of the 14th Scandinavian Symposium and Workshops on Algorithm Theory* (*SWAT 2014*), volume 8503 of *Lecture Notes in Computer Science*, pages 86–97, 2014.

6 Nicolas Bousquet, Arnaud Mary, and Aline Parreau. Token Jumping in Minor-Closed Classes. In *Proceedings of the 21st International Symposium on Fundamentals of Computation Theory* (*FCT 2017*), volume 10472 of *Lecture Notes in Computer Science*, pages 136–149, 2017.

7 Erik D. Demaine, Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Linear-time algorithm for sliding tokens on trees. *Theoretical Computer Science*, 600:132–142, 2015.

8 Persi Diaconis, Ronald Graham, and Susan P. Holmes. *Statistical problems involving permutations with restricted positions*, volume 36 of *Lecture Notes–Monograph Series*, pages 195–222. Institute of Mathematical Statistics, Beachwood, OH, 2001.

9 Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, third edition, 2005.

10 Feodor F. Dragan. On Greedy Matching Ordering and Greedy Matchable Graphs (Extended Abstract). In *Proceedings of the 23rd International Workshop on Graph-Theoretic Concepts in Computer Science* (*WG 1997*), volume 1335 of *Lecture Notes in Computer Science*, pages 184–198, 1997.

11 Feodor F. Dragan. Strongly orderable graphs A common generalization of strongly chordal and chordal bipartite graphs. *Discrete Applied Mathematics*, 99(1–3):427–442, 2000.

12 Martin Dyer and Haiko Müller. Counting perfect matchings and the switch chain. *arXiv preprint arXiv:1705.05790*, 2017.

13 Martin E. Dyer, Mark Jerrum, and Haiko Müller. On the Switch Markov Chain for Perfect Matchings. *Journal of the ACM*, 64(2):12:1–12:33, 2017.

14 Dan Gusfield and Robert W. Irving. *The Stable Marriage Problem*: *Structure and Algorithms*. MIT press, 1989.

15 S. L. Hakimi. On Realizability of a Set of Integers as Degrees of the Vertices of a Linear Graph II. Uniqueness. *Journal of the Society for Industrial and Applied Mathematics*, 11(1):135–147, 1963.

16 Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1–2):72–96, 2005.

17 Jan van den Heuvel. The complexity of change. In *Surveys in Combinatorics 2013*, pages 127–160. Cambridge University Press, 2013.

18 Michael E. Houle, Ferran Hurtado, Marc Noy, and Eduardo Rivera-Campo. Graphs of triangulations and perfect matchings. *Graphs and Combinatorics*, 21(3):325–331, 2005.

19 Wen-Lian Hsu. A Simple Test for Interval Graphs. In *Proceedings of the 18th International Workshop on Graph-Theoretic Concepts in Computer Science* (*WG 1992*), volume 657 of *Lecture Notes in Computer Science*, pages 11–16, 1992.

20 Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999.

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, Marcin Kamiński, Hirotaka Ono, Akira Suzuki, Ryuhei Uehara, and Katsuhisa Yamanaka. On the Parameterized Complexity for Token Jumping on Graphs. In *Proceedings of the 11th Annual Conference on Theory and Applications of Models of Computation* (*TAMC 2014*), volume 8402 of *Lecture Notes in Computer Science*, pages 341–351, 2014.

23 Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theoretical Computer Science*, 439:9–15, 2012.

24 Charles L. Lawson. Software for C_1 surface interpolation. In *Mathematical software*, pages 161–194. Elsevier, 1977.

25 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry*, 49:17–23, 2015.

26 Haruka Mizuta, Takehiro Ito, and Xiao Zhou. Reconfiguration of Steiner Trees in an Unweighted Graph. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 100-A(7):1532–1540, 2017.

27 Naomi Nishimura. Introduction to Reconfiguration. *Algorithms*, 11(4):52, 2018.

28 Nicolau C. Saldanha, Carlos Tomei, Mario A. Casarin, and Domingos Romualdo. Spaces of domino tilings. *Discrete & Computational Geometry*, 14(2):207–233, 1995.

29 Walter J. Savitch. Relationships Between Nondeterministic and Deterministic Tape Complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.

30 James K. Senior. Partitions and Their Representative Graphs. *American Journal of Mathematics*, 73(3):663–689, 1951.

31 Daniel Stefankovic, Eric Vigoda, and John Wilmes. On Counting Perfect Matchings in General Graphs. In *Proceedings of the 13th Latin American Theoretical Informatics Symposium* (*LATIN 2018*), volume 10807 of *Lecture Notes in Computer Science*, pages 873–885, 2018.

32 Todd G. Will. Switching Distance Between Graphs with the Same Degrees. *SIAM Journal on Discrete Mathematics*, 12(3):298–306, 1999.

33 Marcin Wrochna. Reconfiguration in bounded bandwidth and tree-depth. *Journal of Computer and System Sciences*, 93:1–10, 2018.

34 Tom C. van der Zanden. Parameterized Complexity of Graph Constraint Logic. In *Proceedings of the 10th International Symposium on Parameterized and Exact Computation* (*IPEC 2015*), volume 43 of *Leibniz International Proceedings in Informatics*, pages 282–293, 2015.