Complexity of Token Swapping and its Variants*

Édouard Bonnet[1], Tillmann Miltzow[2], and Paweł Rzążewski[3]

1 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
edouard.bonnet@dauphine.fr

2 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
t.miltzow@gmail.com

3 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary; and Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland**
p.rzazewski@mini.pw.edu.pl

Abstract

In the Token Swapping problem we are given a graph with a token placed on each vertex. Each token has exactly one destination vertex, and we try to move all the tokens to their destinations, using the minimum number of swaps, i.e., operations of exchanging the tokens on two adjacent vertices. As the main result of this paper, we show that Token Swapping is $W[1]$-hard parameterized by the length k of a shortest sequence of swaps. In fact, we prove that, for any computable function f, it cannot be solved in time $f(k)n^{o(k/\log k)}$ where n is the number of vertices of the input graph, unless the ETH fails. This lower bound almost matches the trivial $n^{O(k)}$-time algorithm.

We also consider two generalizations of the Token Swapping, namely Colored Token Swapping (where the tokens have colors and tokens of the same color are indistinguishable), and Subset Token Swapping (where each token has a set of possible destinations). To complement the hardness result, we prove that even the most general variant, Subset Token Swapping, is FPT in nowhere-dense graph classes.

Finally, we consider the complexities of all three problems in very restricted classes of graphs: graphs of bounded treewidth and diameter, stars, cliques, and paths, trying to identify the borderlines between polynomial and NP-hard cases.

1998 ACM Subject Classification G.2.2 Graph Theory, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases token swapping, parameterized complexity, NP-hardness, W[1]-hardness

Digital Object Identifier 10.4230/LIPIcs.STACS.2017.16

1 Introduction

In reconfiguration problems, one is interested in transforming a combinatorial or geometric object from one state to another, by performing a sequence of simple operations. An important example is motion planning, where we want to move an object from one configuration to another. Elementary operations are usually translations and rotations. It turns out that

* Supported by the ERC grant PARAMTIGHT: "Parameterized complexity and the search for tight complexity results", no. 280152.

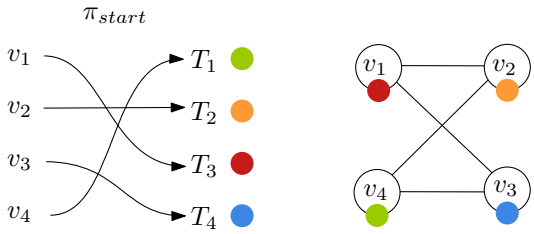

Figure 1 Every token placement can be uniquely described by a permutation.

motion planning can be reduced to the shortest path problem is some higher dimensional Euclidean space with obstacles [7].

Finding the shortest flip sequence between any two triangulations of a convex polygon is a major open problem in computational geometry. Interestingly it is equivalent to a myriad of other reconfiguration problems of so-called Catalan structures [4]. Examples include: binary trees, perfect matchings of points in convex position, Dyck words, monotonic lattice paths, and many more. Reconfiguring permutations under various constraints is heavily studied and usually called *sorting*.

An important class of reconfiguration problems is a big family of problems in graph theory that involves moving tokens, pebbles, cops or robbers along the edges of a given graph, in order to reach some final configuration [30, 5, 9, 14, 2, 28, 20, 8, 12]. In this paper, we study one of them.

The Token Swapping problem, introduced by Akers and Krishnamurthy [1], and stated more recently by Yamanaka *et al.* [31], fits nicely into this long history of reconfiguration problems and can be regarded as a sorting problem with special constraints.

The problem is defined as follows, see also Figure 1. We are given an undirected connected graph with n vertices $v_1, \ldots, v_n$, a set of tokens $T = \{t_1, \ldots, t_n\}$ and two permutations π_{start} and π_{target}. These permutations are called start permutation and target permutation. Initially vertex v_i holds token $t_{\pi_{\text{start}}(i)}$. In one step, we are allowed to *swap* tokens on a pair of adjacent vertices, that is, if v and w are adjacent, v holds the token s, and w holds the token t, then the swap between v and w results in the configuration where v holds t, w holds s, and all the other tokens stay in place. The Token Swapping problem asks if the target configuration can be reached in at most k swaps. Thus, a solution for the Token Swapping problem is a sequence of edges, where the swaps take place. The solution is optimal if its length is shortest possible. To see the correspondence to sorting note that every placement of tokens can be regarded as a permutation and the target permutation can be regarded as the sorted state.

It has been observed, for example in [1, 31], that every instance of Token Swapping has a solution, and its length is $O(n^2)$. Moreover, $\Omega(n^2)$ swaps are sometimes necessary. It is interesting to note that some special cases of Token Swapping have been studied in the context of sorting permutations with additional restrictions (see Knuth [21, Section 5.2.2] for paths, Pak [27] for stars, Cayley [6] for cliques, and Heath and Vergara [16] for squares of a path). Recently the problem was also solved for a special case of complete split graphs (see Yasui *et al.* [33]). Is is also worth mentioning that a very closely related concept of sorting permutations using cost-constrained transitions was considered by Farnoud *et al.* [11], and Farnoud and Milenkovic [10].

The complexity of the Token Swapping problem was investigated by Miltzow *et al.* [25]. They show that the problem is NP-complete and APX-complete. Moreover, they show that any algorithm solving the Token Swapping problem in time $2^{o(n)}$ would refute the Exponential Time Hypothesis (ETH) [17]. The results of Miltzow *et al.* [25] carry over also to

some generalization of the Token Swapping problem, called Colored Token Swapping, first introduced by Yamanaka *et al.* [32]. In this problem, vertices and tokens are partitioned into color classes. For each color c, the number of tokens colored c equals the number of vertices colored c. The goal is to reach, with the minimum number of swaps, a configuration in which each vertex contains a token of its own color. Token Swapping corresponds to the special case where each color class comprises exactly one token and one vertex. NP-hardness of Colored Token Swapping was first shown by Yamanaka *et al.* [32], even in the case where only 3 colors exist.

We introduce the Subset Token Swapping problem, which is an even further generalization of Token Swapping. Here a function $D : T \to 2^V$ specifies the set $D(t_i)$ of possible destinations $D(t_i)$ for the token t_i. Observe that Subset Token Swapping also generalizes Colored Token Swapping. It might happen that there is no satisfying swapping sequence at all to this new problem. Though, this can be checked in polynomial time by deciding if there is a perfect matching in the bipartite token-destination graph. Thus we shall always assume that we have a satisfiable instance.

In this paper we continue and extend the work of Miltzow *et al.* [25]. They presented a very simple algorithm which solves the instance of the Token Swapping problem in $n^{O(k)}$ time and space, where k denotes the number of allowed swaps. In Section 3 we show that this algorithm can be easily generalized to Colored Token Swapping and Subset Token Swapping problems. One of the main bottlenecks for exponential-time algorithms is not time, but space consumption. Thus we present a slightly slower exact algorithm, using only polynomial space (in fact, only slightly super-linear).

The existence of an XP algorithm (i.e., with time complexity $O(n^{f(k)})$ for some computable function f) for the Token Swapping problem gives rise to the question whether the problem can be solved in FPT time (i.e., $f(k) \cdot n^{O(1)}$, for some computable function f). There is some evidence indicating that this could be possible. First, observe that an instance with more than $2k$ misplaced tokens, is a No-instance, as each swap moves only two tokens. Further, one can safely remove all vertices from the graph that are at distance more than k from all misplaced tokens. This preprocessing yields an equivalent instance, where every connected component has *diameter* $O(k^2)$. Thus if the maximum degree Δ is bounded by k, each component has size bounded by a function of k. The connected components of $f(k)$ *size* can be solved separately by exhaustively guessing (still in FPT time) the number of swaps to perform in each of them. Moreover, even the generalized Subset Token Swapping problem is FPT in $k + \Delta$ (see Proposition 6). For those reasons, one could have hoped for an FPT algorithm for general graphs. However, as the main result of this paper, we show in Section 4 that this is very unlikely.

▶ **Theorem 1** (Parameterized Hardness). Token Swapping *is* $W[1]$*-hard, parameterized by the number* k *of allowed swaps. Moreover, assuming the ETH, for any computable function* f*,* Token Swapping *cannot be solved in time* $f(k)(n + m)^{o(k/\log k)}$ *where* n *and* m *are respectively the number of vertices and edges of the input graph.*

Observe that this lower bound shows that the simple $n^{O(k)}$-time algorithm is almost best possible. It is worth mentioning that the parameter for which we show hardness is in fact *number of swaps + number of initially misplaced tokens + diameter of the graph*, which matches the reasoning presented in the previous paragraph.

To show the lower bound, we introduce handy gadgets called *linkers*. They are simple and can be used to give a significantly simpler proof of the lower bounds given by Miltzow *et al.* [25].

Table 1 The parameterized complexity of Token Swapping, Colored Token Swapping, and Subset Token Swapping.

	$k + \Delta$	$k +$ diam	k, nowhere-dense / $k +$ tw	tw + diam
TS	FPT ([25])	**W[1]-h** (Th 1)	FPT	**paraNP-c** (Th 4)
CTS	FPT	W[1]-h	FPT	paraNP-c
STS	**FPT** (Prop 6)	W[1]-h	**FPT** (Th 2)	paraNP-c

Since there is no FPT algorithm for the Token Swapping problem (parameterized by the number k of swaps), unless FPT $= W[1]$, a natural approach is to try to restrict the input graph classes, in hope to obtain some positive results. Indeed, in Section 5 we show that FPT algorithms exist, if we restrict our input to the so-called *nowhere-dense graph classes*.

▶ **Theorem 2** (FPT in nowhere dense graphs). Subset Token Swapping *is FPT parameterized by k on nowhere-dense graph classes.*

The notion of nowhere-dense graph classes has been introduced as a common generalization of several previously known notions of sparsity in graphs such as planar graphs, graphs with forbidden (topological) minors, graphs with (locally) bounded treewidth or graphs with bounded maximum degree. Grohe, Kreutzer, and Siebertz [15] proved that every property definable as a first-order formula φ is solvable in $O(f(|\varphi|, \varepsilon)\, n^{1+\varepsilon})$ time on nowhere-dense classes of graphs, for every $\varepsilon > 0$. We use this meta-theorem to show the existence of an FPT time algorithm for the Subset Token Swapping problem, restricted to nowhere-dense graph classes. In particular, this implies the following results.

▶ **Corollary 3.** Subset Token Swapping *is FPT*
(a) *parameterized by $k + \mathrm{tw}(G)$,*
(b) *parameterized by k in planar graphs.*

It is often observed that NP-hard graph problems become tractable on classes of graphs with bounded treewidth (or, at least, with bounded tree-depth; see Nešetřil and Ossona de Mendez [26, Chapter 10] for the definition and some background of tree-depth and related parameters). It is not uncommon to see FPT algorithms running in time $f(\mathrm{tw})n^{O(1)}$ (or $f(\mathrm{td})n^{O(1)}$) or XP algorithms running in time $n^{f(\mathrm{tw})}$ (or $n^{f(\mathrm{td})}$), for some computable functions f. Especially, in light of Corollary 3(a), we want to know if there exists an algorithm that runs in polynomial time for constant treewidth. In Section 6 we rule out the existence of such algorithms by showing that Token Swapping remains NP-hard when restricted to graphs with tree-depth 4 (treewidth and pathwidth 2; diameter 6; distance 1 to a forest).

▶ **Theorem 4** (Hard on Almost Trees). Token Swapping *remains* NP*-hard even when both the treewidth and the diameter of the input graph are constant, and cannot be solved in time $2^{o(n)}$, unless the ETH fails.*

The Table 1 shows the current state of our knowledge about the parameterized complexity of Token Swapping (TS), Colored Token Swapping (CTS), and Subset Token Swapping (STS) problems, for different choices of parameters.

Table 2 The complexity of Token Swapping (TS), Colored Token Swapping (CTS), and Subset Token Swapping (STS) on very restricted classes of graphs. The results in bold are proved in this paper. The three polynomial algorithms for Token Swapping on cliques, stars, and paths, are folklore and can be found for instance in [25, 19].

	trees	cliques	stars	paths
TS	?	P	P	P
CTS	?	**NP-c**	**P**	**P**
STS	**NP-c**	**NP-c**	**NP-c**	?

While we think that our results give a fairly detailed view on the complexity landscape of the Token Swapping problem, we also want to point out that our reductions are significantly simpler than those by Miltzow *et al.* [25].

Since the investigated problems seem to be immensely intractable, we investigate their complexities in very restricted classes of graphs, namely cliques, stars, and paths. We focus on finding the borderlines between easy (polynomially solvable) and hard (NP-hard) cases. The summary of these results is given in Table 2. Observe that cliques distinguish the complexities of the Token Swapping and the Colored Token Swapping problems, while stars distinguish the complexities of the Colored Token Swapping and the Subset Token Swapping problems.

The paper concludes with several open problems in Section 7.

2 Preliminaries

For a token t, let $\mathrm{dist}(t)$ denote the distance from the position of t to its destination. For an instance I of the Token Swapping problem, we define $L(I) := \sum_t \mathrm{dist}(t)$, i.e., the sum of distances to the destination over all the tokens. Clearly, after performing a single swap, $\mathrm{dist}(t)$ may change by at most 1. We shall also use the following classification of swaps: for $x, y \in \{-1, 0, 1\}$, $x \leq y$, by a (x, y)-*swap* we mean a swap, in which one token changes its distance by x, and the other one by y. Intuitively, $(-1, -1)$-swaps are the most "efficient" ones, thus we will call them *happy swaps*. Since each swap involves two tokens, we get the following lower bound.

▶ **Proposition 5** ([25]). *The length of an optimal solution for an instance I of* Token Swapping *is at least $L(I)/2$. Besides, it is* exactly *$L(I)/2$ iff there is a solution using happy swaps only.*

When designing algorithms, it is natural to ask about lower bounds. However, the standard complexity assumption used for distinguishing easy and hard problems, P $\neq$ NP, cannot rule out, say, subexponential time algorithms. The stronger assumption that is typically used for this purpose is the so-called *Exponential Time Hypothesis* (ETH), formulated by Impagliazzo and Paturi [17]. We refer the reader to the survey by Lokshtanov and Marx for more information about ETH and conditional lower bounds [22]. The version we present below (and is most commonly used) is not the original statement of this hypothesis, but its weaker version (see also Impagliazzo, Paturi, and Zane [18]).

▶ **Exponential Time Hypothesis** (Impagliazzo and Paturi [17]). *There is no algorithm solving every instance of* 3-Sat *with N variables and M clauses in time $2^{o(N+M)}$.*

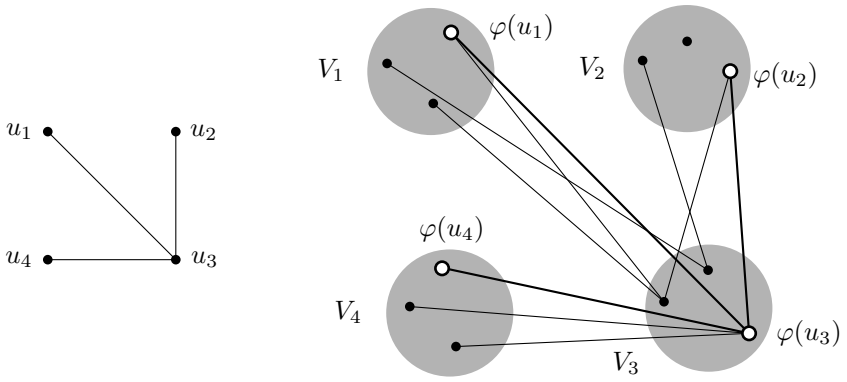

Figure 2 On the left is the pattern graph P; on the right, the host graph H. We indicate the image of φ with white vertices. To keep the example small, we did not make P 3-regular.

3 Algorithms

First, we prove that Subset Token Swapping (and therefore also Colored Token Swapping as its restriction) is FPT in $k + \Delta$, where k is the number of allowed swaps, and Δ is the maximum degree of the input graph. This generalizes the observation of Miltzow *et al.* [25] for the Token Swapping problem. Furthermore, we show that the simple algorithm for the Token Swapping problem, presented by Miltzow *et al.* [25], carries over to the generalized problems, i.e., Colored Token Swapping and Subset Token Swapping. At last, we will present an algorithm that has polynomial space complexity.

▶ **Proposition 6.** Subset Token Swapping *problem is FPT in* $k + \Delta$ *and admits a kernel of size* $2k + 2k^2 \cdot \Delta^k$.

Miltzow *et al.* [25] show that an optimal solution for the Token Swapping problem can be found by performing a breath-first-search on the *configuration graph*, that is, the graph whose vertices are all possible configurations of tokens on vertices, and two configurations are adjacent when one can be obtained from the other with a single swap[1]. We observe that the same approach works for the Colored Token Swapping and the Subset Token Swapping problems, the only difference is that we terminate on any feasible target configuration.

The main drawback of such an approach is an exponential space complexity. Here we show the following complementary result, inspired by the ideas of Savitch [29].

▶ **Theorem 7.** *Let* G *be a graph with* n *vertices, and let* k *be the maximum number of allowed swaps. The* Subset Token Swapping *problem on* G *can be solved in time* $2^{O(n \log n \log k)} = 2^{O(n \log^2 n)}$ *and space* $O(n \log n \log k) = O(n \log^2 n)$.

4 Lower Bounds on parameterized Token Swapping

Let us start by defining an auxiliary problem, called Multicolored Subgraph Isomorphism (also known as Partitioned Subgraph Isomorphism; see Figure 2).

[1] The configuration graph of a Token Swapping instance on a graph G with n vertices can also be seen as the Cayley graph $\Gamma(\mathcal{P}_n, S)$ where $\mathcal{P}_n$ is the symmetric group on n elements and S is the set of all transpositions $(u\ v)$ where uv is an edge of G.

In Multicolored Subgraph Isomorphism, one is given a host graph H whose vertex set is partitioned into k color classes $V_1 \uplus V_2 \uplus \ldots \uplus V_k$ and a pattern graph P with k vertices: $V(P) = \{u_1, \ldots, u_k\}$. The goal is to find an injection $\varphi : V(P) \to V(H)$ such that $u_iu_j \in E(P)$ implies that $\varphi(u_i)\varphi(u_j) \in E(H)$ and $\varphi(u_i) \in V_i$ for all i, j. Thus we can assume that each V_i forms an independent set. Further, we assume without loss of generality that $E(V_i, V_j) := \{ab \in E(H) \colon a \in V_i, b \in V_j\}$ is non-empty iff $u_iu_j \in E(P)$. In other words, we try to find k vertices $v_1 \in V_1$, $v_2 \in V_2$, ..., $v_k \in V_k$ such that, for any $i < j \in [k]$,[2] there is an edge between v_i and v_j iff $E(V_i, V_j)$ is non-empty. The $W[1]$-hardness of Multicolored Subgraph Isomorphism problem follows from the $W[1]$-hardness of the Multicolored Clique. Marx [23] showed that assuming the ETH, Multicolored Subgraph Isomorphism cannot be solved in time $f(k)(|V(H)| + |E(H)|)^{o(k/\log k)}$, for any computable function f, even when the pattern graph P is 3-regular and bipartite (see also Marx and Pilipczuk [24]). In particular, k has to be an even integer since $|E(P)|$ is exactly $3k/2$. We finally assume that for every $i \in [k]$ it holds that $|V_i| = t$, by padding potentially smaller classes with isolated vertices. This can only increase the size of the host graph by a factor of k, and does not create any new solution nor destroy any existing one.

Now we are ready to prove the following theorem.

▶ **Theorem 1** (Parameterized Hardness). Token Swapping *is $W[1]$-hard, parameterized by the number k of allowed swaps. Moreover, assuming the ETH, for any computable function f,* Token Swapping *cannot be solved in time $f(k)(n + m)^{o(k/\log k)}$ where n and m are respectively the number of vertices and edges of the input graph.*

Proof (Sketch). We will present a reduction from Multicolored Subgraph Isomorphism. To show the parameterized hardness of the Token Swapping problem, we introduce a very handy *linker gadget*. This gadget has a robust and general ability to link decisions. As such, it permits to reduce from a wide range of problems. Its description is short and its soundness is intuitive. Because it yields very light constructions, we can rule out fairly easily unwanted swap sequences. We describe the linker gadget and provide some intuitive reason why it works (see Figure 3).

Linker gadget. Given two integers a and b, the linker gadget $L_{a,b}$ contains a set of a vertices, called *finishing set* and a path on a vertices, that we call *starting path*. The tokens initially on vertices of the finishing set are called *local tokens*; they shall go to the vertices of the starting path in the way depicted in Figure 3. The tokens initially on vertices of the starting path are called *global tokens*. Global tokens have their destination in some other linker gadget. To be more specific, their destination is in the finishing set of another linker.

We describe and always imagine the finishing set and the starting paths *to be ordered from left to right*. Below the finishing set and to the left of the starting path, stand b disjoint induced paths, each with a vertices, arranged in a grid, see Figure 3. We call those paths *private paths*. The *private tokens* on private paths are already well-placed. Every vertex in the finishing set is adjacent to all private vertices below it and the leftmost vertex of the starting path is adjacent to all rightmost vertices of the private paths.

For local tokens to go to the starting path, they must go through a private path. As its name suggests, the linker gadget aims at linking the choice of the private path used for every local token. Intuitively, the only way of benefiting from a^2 happy swaps between the a local tokens and the a global tokens is to use a common private path (note that the destination

[2] For an integer p, by $[p]$ we denote the set $\{1, \ldots, p\}$.

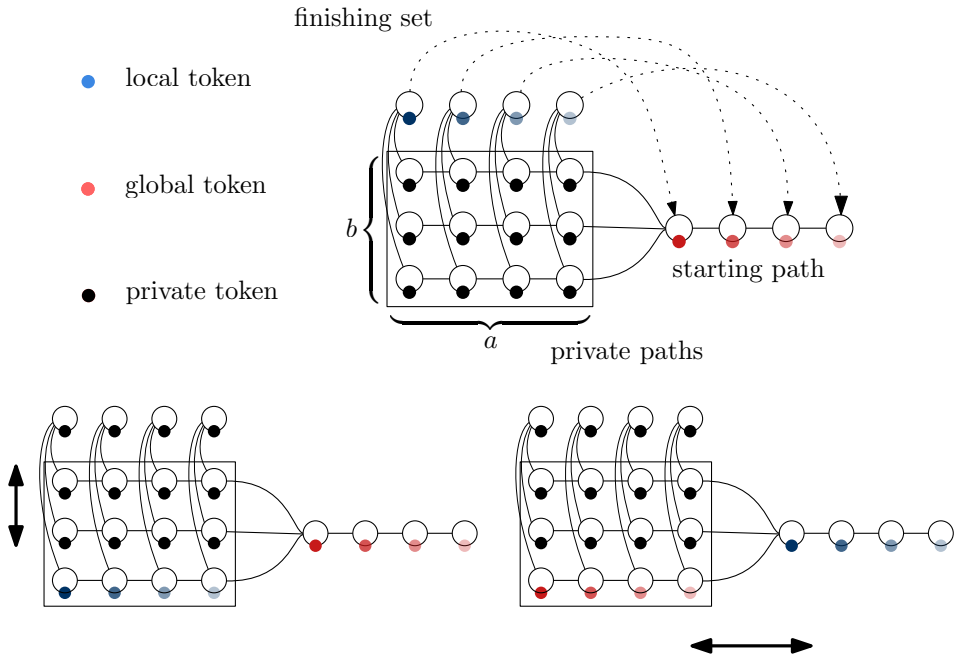

Figure 3 The linker gadget $L_{a,b}$. Black tokens are initially properly placed. Dashed arcs represent where tokens of the finishing set should go in the starting path. At the bottom left, we depict the gadget after all the local tokens are swapped to a single private path. At the bottom right, we see the result after swapping all the local tokens to the starting path. In this case, the global tokens go to that private path.

of the global tokens will make those swaps happy). That results in a kind of configuration as depicted in the bottom right of Figure 3, where each global token is in the same private path. The fate of the global tokens has been linked.

Construction. We present a reduction from Multicolored Subgraph Isomorphism with cubic pattern graphs to Token Swapping where the number of allowed swaps is linear in k. Let (H, P) be an instance of Multicolored Subgraph Isomorphism. For any color class $V_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,t}\}$ of H, we add a copy of the linker $L_{3,t}$ that we denote by L_i. We denote by $j_1 < j_2 < j_3$ the indices of the neighbors of u_i in the pattern graph P. The linker L_i will be linked to 3 other gadgets and it has t private paths (or *choices*). The finishing set of L_i contains, from left to right, the vertices $a(i, j_1)$, $a(i, j_2)$, and $a(i, j_3)$. We denote the tokens initially on the vertices $a(i, j_1)$, $a(i, j_2)$, and $a(i, j_3)$ by $\text{local}(i, j_1), \text{local}(i, j_2), \text{local}(i, j_3)$, respectively.

The starting path contains, from left to right, vertices $b(i, j_1)$, $b(i, j_2)$, and $b(i, j_3)$ with tokens $\text{global}(i, j_1)$, $\text{global}(i, j_2)$, and $\text{global}(i, j_3)$.

For each $p \in [3]$, $\text{local}(i, j_p)$ shall go to vertex $b(i, j_p)$, whereas $\text{global}(i, j_p)$ shall go to $a(j_p, i)$ in the gadget L_{j_p}. Observe that the former transfer is internal and may remain within the gadget L_i, while the latter requires some interplay between the gadgets L_i and L_{j_p}. For any $h \in [t]$, by $\mathrm{U}(i, h)$ we denote the h-th private path. This path represents the vertex $v_{i,h}$. The path $\mathrm{U}(i, h)$ consists of, from left to right, vertices $u(i, h, j_1)$, $u(i, h, j_2)$, $u(i, h, j_3)$. We set $\mathrm{U}(i) := \bigcup_{h \in [t]} \mathrm{U}(i, h)$. Initially, all the tokens placed on vertices of $\mathrm{U}(i)$ are already well placed.

We complete the construction by adding every edge of the form $u(i, h, j)u(j, h', i)$ if $v_{i,h}v_{j,h'}$ is an edge in $E(V_i, V_j)$ (see Figure 5). Let G be the graph that we built, and let I

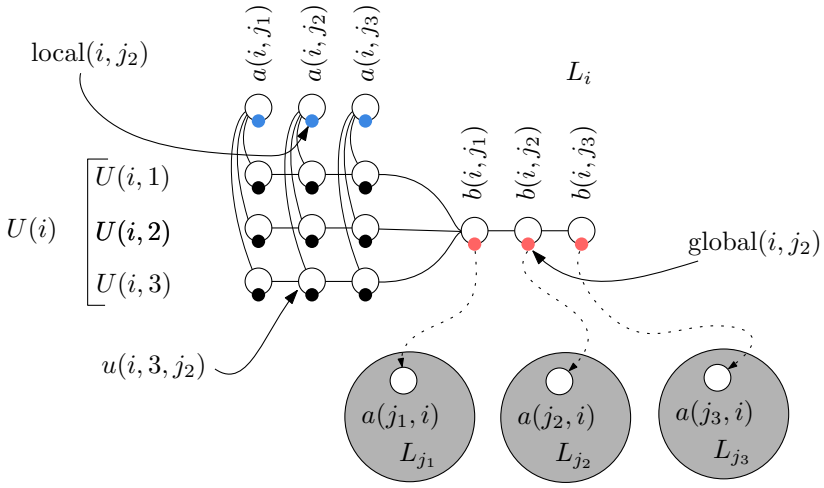

Figure 4 The different labels for tokens, vertices, and sets of vertices.

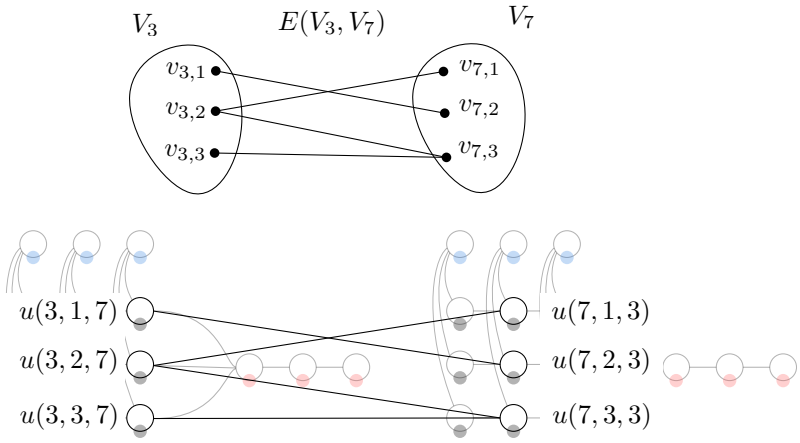

Figure 5 The way linkers (in that case, L_3 and L_7) are assembled together, with $t = 3$.

be the whole instance of Token Swapping (with the initial position of the tokens). We claim that (H, P) is a Yes-instance of Multicolored Subgraph Isomorphism if and only if I has a solution of length at most $\ell := 16.5k = O(k)$. Recall that k is even, so $16.5k$ is an integer.

Correctness. As already described above, the local tokens can reach their target vertices more efficiently if they all use the same private path. This private path represents a choice of the corresponding vertex in the original Multicolored Subgraph Isomorphism instance. Thereafter, each global token can go with a single swap to its correct target gadget, if there were an edge between the corresponding vertices. This sequence needs exactly $16.5k$ swaps.

The reverse direction is more difficult. Observe that, except from the swaps involving private tokens, all the swaps in the described solution are happy. However, it can be shown that the swaps that are not happy are necessary. In particular, any deviation from the intended solution requires additional swaps. ◀

5 Token Swapping on nowhere-dense classes of graphs

As we have seen in Section 4, there is little hope for an FPT algorithm for the Token Swapping problem (parameterized by k), unless FPT $= W[1]$. Now let us show that FPT algorithms exist, if we restrict our input to nowhere-dense graph classes.

The formal definition of nowhere-dense graphs is technical, so we refer the reader to the comprehensive book of Nešetřil and Ossona de Mendez [26, Chapter 13].

As graphs with bounded degree are nowhere-dense, this result generalizes Proposition 6.

▶ **Theorem 2** (FPT in nowhere dense graphs). Subset Token Swapping *is FPT parameterized by k on nowhere-dense graph classes.*

We derive the following corollary.

▶ **Corollary 8.** Subset Token Swapping *is FPT*
(a) *parameterized by $k + \mathrm{tw}(G)$,*
(b) *parameterized by k in planar graphs.*

To see Corollary 3 *(a)*, recall that bounded-treewidth graphs are nowhere-dense. Thus by Theorem 2 there exists an algorithm with running time $O(f(k)\, n^{1+\varepsilon})$, for any $\varepsilon > 0$ and treewidth bounded by some constant c. Observe that the constant hidden in the big-O notation depends on the constant c. In particular c has no influence on the exponent of n.

6 Token Swapping on almost trees

This section is devoted to the proof of the following theorem.

▶ **Theorem 4** (Hard on Almost Trees). Token Swapping *remains* NP*-hard even when both the treewidth and the diameter of the input graph are constant, and cannot be solved in time $2^{o(n)}$, unless the ETH fails.*

Proof. In Exact Cover by 3-Sets, one is given a family $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of 3-element subsets of the universe $X = \{x_1, x_2, \ldots, x_n\}$, where 3 divides n. The goal is to find $n/3$ subsets in $\mathcal{S}$ that partition (or here, equivalently, cover) X. The problem can be seen as a straightforward generalization of the 3-Dimensional Matching problem. This problem is NP-complete and has no $2^{o(n)}$ algorithm, unless the ETH fails, even if each element belongs to exactly 3 triples [13, 3]. Therefore we can reduce from the restriction of the Exact Cover by 3-Sets problem, where each element belongs to 3 sets of $\mathcal{S}$, and obviously $|\mathcal{S}| = |X| = n$.

Construction. For each set $S_j \in \mathcal{S}$, we add a *set gadget* consisting of a tree on 10 vertices (see Figure 6). In the set gadget, the four gray tokens should cyclically swap as indicated by the dotted arrows: g_i^j shall go where g_{i+1}^j is, for each $i \in [4]$ (addition is computed modulo 4). The three black tokens, as usual, are initially well placed. The three remaining vertices are called *element* vertices. They represent the three elements of the set. The tokens initially on the *element* vertices are called *element* tokens. For each element of X, there are 3 *element* tokens and 3 *element* vertices.

We add a vertex c that is linked to all the *element* vertices of the set gadgets and to all the vertices g_0^j. Each token originally on an *element* vertex should cyclically go to *its next occurrence* (see Figure 7). The token initially on c is well placed.

The constructed graph G has $10n + 1$ vertices. If one removes the vertex c the remaining graph is a forest, which means that the graph has a feedback vertex set of size 1 and, in

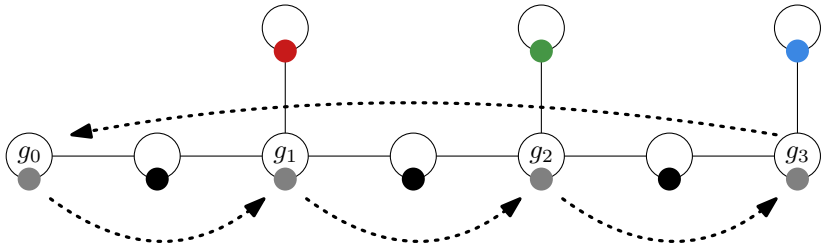

Figure 6 The set gadget for red, green and blue. We voluntarily omit the superscript j.

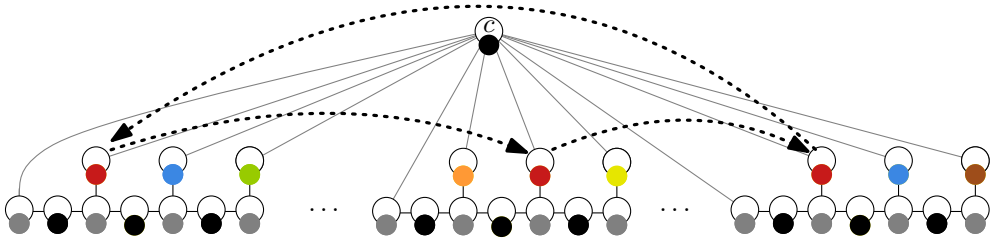

Figure 7 The overall picture. Each element appears exactly 3 times, so there are 3 red tokens.

particular, treewidth 2. G has its diameter bounded by 6, since all the vertices are at distance at most 3 of the vertex c. We now show that the instance $\mathcal{S}$ of Exact Cover by 3-Sets admits a solution iff there exists a solution for our instance of Token Swapping of length at most $\ell := 11 \cdot n/3 + 9 \cdot 2n/3 + 2n = 35n/3 = 11n + 2n/3$.

Soundness. The correctness of the construction relies mainly on the fact that there are two competitive ways of placing the gray tokens. The first way is the most direct. It consists of only swapping along the *spine* of the set gadget. By *spine*, we mean the 7 vertices initially containing gray or black tokens. From hereon, we call that *swapping the gray tokens internally*.

▶ **Claim 9.** *Swapping the gray tokens internally requires* 9 *swaps.*

Proof. In 6 swaps, we can first move g_3 to its destination (where g_0 is initially). Then, g_0, g_1, and g_2 need one additional swap each to be correctly placed. We observe that, after we do so, the black tokens are back to their respective destination. ◀

We call the second way *swapping the gray tokens via c*. Basically, it is the way one would have to place the gray tokens if the black tokens (except the one in c) were removed from the graph. It consists of, first (a) swapping g_0 with the token on c, then moving g_0 to its destination, then (b) swapping g_1 with the current token on c, moving g_1 to its destination, (c) swapping g_2 with the token on c, moving g_2 to its destination, finally (d) swapping g_3 with the token on c and moving it to its destination.

▶ **Claim 10.** *Swapping the gray tokens via c requires* 11 *swaps.*

Proof. Steps (a), (b), and (c) take 3 swaps each, while step (d) takes 2 swaps. ◀

Considering that swapping the gray tokens via c takes 2 more swaps than swapping them internally, and leads to the exact same configuration where both the black tokens and the *element* tokens are back to their initial position, one can question the interest of the second

way of swapping the gray tokens. It turns out that, at the end of steps (a), (b), and (c), an *element* token is on vertex c. We will take advantage of that situation to perform two consecutive happy swaps with its two other occurrences. By doing so, observe that the first swap of steps (b), (c), and (d) are also happy and place the last occurrence of the *element* tokens at its destination.

We assume that there is a solution $S_{a_1}, \ldots, S_{a_{n/3}}$ to the Exact Cover by 3-Sets instance. In the corresponding $n/3$ set gadgets, swap the gray tokens via c and interleave those swaps with doing the two happy swaps over *element* tokens, whenever such a token reaches c. By Claim 10, this requires $11 \cdot n/3 + 2n$ swaps. At this point, the tokens that are misplaced are the $4 \cdot 2n/3$ gray tokens in the $2n/3$ remaining set gadgets. Swap those gray tokens internally. This adds $9 \cdot 2n/3$ swaps, by Claim 9. Overall, this solution consists of $29n/3 + 2n = 35n/3 = \ell$.

Let us now suppose that there is a solution $\boldsymbol{s}$ of length at most ℓ to the Token Swapping instance. At this point, we should observe that there are alternative ways (to Claim 9 and Claim 10) of placing the gray tokens at their destination. For instance, one can move g_3 to g_1 along the spine, place tokens g_2 and g_3, then exchange g_0 with the token on c, move g_0 to its destination, swap g_3 with the token on c, and finally move it to its destination. This also takes 11 swaps but moves only one *element* token to c (compared to moving all three of them in the strategy of Claim 10). One can check that all those alternative ways take 11 swaps or more. Let $r \in [0, n]$ be such that $\boldsymbol{s}$ does *not* swap the gray tokens internally in r set gadgets (and swap them internally in the remaining $n - r$ set gadgets). The length of $\boldsymbol{s}$ is at least $11r + 9(n - r) + 2(n - q) + 4q = 11n + 2(r + q)$, where q is the number of elements of X for which *none* occurrence of its three *element* tokens has been moved to c in the process of swapping the gray tokens. Indeed, for each of those q elements, 4 additional swaps will be eventually needed. For each of the remaining $n - q$ elements, only 2 additional happy swaps will place the three corresponding *element* tokens at their destination. It holds that $3r \geqslant n - q$, since the *element* tokens within the r set gadgets where $\boldsymbol{s}$ does not swap internally represent at most $3r$ distinct elements of X. Hence, $3r + q \geqslant n$. Also, $\boldsymbol{s}$ is of length at most $\ell = 11n + 2n/3$, which implies that $r + q \leqslant n/3$. Thus, $n \leqslant 3r + q \leqslant 3r + 3q \leqslant n$. Therefore, $q = 0$ and $r = n/3$. Let $S_{a_1}, \ldots, S_{a_{n/3}}$ be the $n/3$ sets for which $\boldsymbol{s}$ does not swap the gray tokens internally in the correponding set gadgets. For each element of X, an occurrence of a corresponding *element* token is moved to c when the gray tokens are swapped in one of those gadgets. So this element belongs to one S_{a_i} and therefore $S_{a_1}, \ldots, S_{a_{n/3}}$ is a solution to the instance of Exact Cover by 3-Sets.

The ETH lower bound follows from the fact, that the size of constructed graph is $O(n)$. ◀

7 Conclusion

We conclude the paper with several ideas for further research. First, we believe that it would be interesting to fill the missing entries in Table 2. In particular, we conjecture that the Token Swapping problem remains NP-complete even if the input graph is a tree.

Another interesting problem is the following. By Miltzow *et al.* [25, Theorem 1], the Token Swapping problem can be solved in time $2^{O(n \log n)}$, and there is no $2^{o(n)}$ algorithm, unless the ETH fails. We conjecture that the lower bound can be improved to $2^{o(n \log n)}$. It would also be interesting to find single-exponential algorithms for some restricted graph classes, such as graphs with bounded treewidth or planar graphs.

Finally, to prove Corollary 3, we use the powerful and very general meta-theorem by Grohe, Kreutzer, and Siebertz [15]. It would be interesting to obtain elementary FPT algorithms for planar graphs and graphs with bounded treewidth (or even trees).

References

1. Sheldon B Akers and Balakrishnan Krishnamurthy. A group-theoretic model for symmetric interconnection networks. *IEEE transactions on Computers*, 38(4):555–566, 1989.
2. Elwyn R Berlekamp, John Horton Conway, and Richard K Guy. *Winning Ways, for Your Mathematical Plays: Games in particular*, volume 2. Academic Pr, 1982.
3. Hans L. Bodlaender and Jesper Nederlof. Subexponential time algorithms for finding small tree and path decompositions. In *ESA 2015 Proc.*, pages 179–190. Springer, 2015. doi:10.1007/978-3-662-48350-3_16.
4. Prosenjit Bose and Ferran Hurtado. Flips in planar graphs. *Computational Geometry*, 42(1):60–80, 2009.
5. Gruia Calinescu, Adrian Dumitrescu, and János Pach. Reconfigurations in graphs and grids. In *LATIN 2006 Proc.*, pages 262–273. Springer, 2006. doi:10.1007/11682462_27.
6. Arthur Cayley. LXXVII. Note on the theory of permutations. *Philosophical Magazine Series 3*, 34(232):527–529, 1849. doi:10.1080/14786444908646287.
7. Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. Computational geometry. In *Computational geometry*, pages 1–17. Springer, 2000.
8. Erik D. Demaine, Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Linear-time algorithm for sliding tokens on trees. *Theoretical Computer Science*, 600:132–142, 2015. doi:10.1016/j.tcs.2015.07.037.
9. Ruy Fabila-Monroy, David Flores-Peñaloza, Clemens Huemer, Ferran Hurtado, Jorge Urrutia, and David R. Wood. Token graphs. *Graphs and Combinatorics*, 28(3):365–380, 2012. doi:10.1007/s00373-011-1055-9.
10. F. Farnoud, C. Y. Chen, O. Milenkovic, and N. Kashyap. A graphical model for computing the minimum cost transposition distance. In *Information Theory Workshop (ITW), 2010 IEEE*, pages 1–5, Aug 2010. doi:10.1109/CIG.2010.5592890.
11. Farzad Farnoud and Olgica Milenkovic. Sorting of permutations by cost-constrained transpositions. *IEEE Trans. Information Theory*, 58(1):3–23, 2012. doi:10.1109/TIT.2011.2171532.
12. Eli Fox-Epstein, Duc A. Hoang, Yota Otachi, and Ryuhei Uehara. Sliding token on bipartite permutation graphs. In Khaled Elbassioni and Kazuhisa Makino, editors, *Algorithms and Computation*, volume 9472 of *Lecture Notes in Computer Science*, pages 237–247. Springer Berlin Heidelberg, 2015. doi:10.1007/978-3-662-48971-0_21.
13. Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985. doi:10.1016/0304-3975(85)90224-5.
14. Daniel Graf. How to sort by walking on a tree. In *ESA 2015 Proc.*, pages 643–655. Springer, 2015. doi:10.1007/978-3-662-48350-3_54.
15. Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. In *STOC 2014 Proc.*, pages 89–98. ACM, 2014. doi:10.1145/2591796.2591851.
16. Lenwood S. Heath and John Paul C. Vergara. Sorting by short swaps. *Journal of Computational Biology*, 10(5):775–789, 2003. doi:10.1089/106652703322539097.
17. Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.
18. Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.
19. Mark R. Jerrum. The complexity of finding minimum-length generator sequences. *Theoretical Computer Science*, 36:265–289, 1985. doi:10.1016/0304-3975(85)90047-7.

20 Takumi Kasai, Akeo Adachi, and Shigeki Iwata. Classes of pebble games and complete problems. *SIAM Journal on Computing*, 8(4):574–586, 1979. doi:10.1137/0208046.

21 Donald Erwin Knuth. *The Art of Computer Programming*, volume 3 / Sorting and Searching. Addison-Wesley, 1982. ISBN 0-201-03803-X.

22 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011. URL: http://albcom.lsi.upc.edu/ojs/index.php/beatcs/article/view/96.

23 Dániel Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010. doi:10.4086/toc.2010.v006a005.

24 Dániel Marx and Michal Pilipczuk. Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. *CoRR*, abs/1504.05476, 2015. URL: http://arxiv.org/abs/1504.05476.

25 Tillmann Miltzow, Lothar Narins, Yoshio Okamoto, Günter Rote, Antonis Thomas, and Takeaki Uno. Approximation and hardness of token swapping. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 66:1–66:15. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.66.

26 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity – Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

27 Igor Pak. Reduced decompositions of permutations in terms of star transpositions, generalized Catalan numbers and k-ARY trees. *Disc. Math.*, 204(1):329–335, 1999. doi:10.1016/S0012-365X(98)00377-X.

28 Torrence D Parsons. Pursuit-evasion in a graph. In *Theory and applications of graphs*, pages 426–441. Springer, 1978.

29 Walter J. Savitch. Relationships Between Nondeterministic and Deterministic Tape Complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970. doi:10.1016/S0022-0000(70)80006-X.

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

31 Katsuhisa Yamanaka, Erik D. Demaine, Takehiro Ito, Jun Kawahara, Masashi Kiyomi, Yoshio Okamoto, Toshiki Saitoh, Akira Suzuki, Kei Uchizawa, and Takeaki Uno. Swapping labeled tokens on graphs. In *FUN 2014 Proc.*, pages 364–375. Springer, 2014. doi:10.1007/978-3-319-07890-8_31.

32 Katsuhisa Yamanaka, Takashi Horiyama, David G. Kirkpatrick, Yota Otachi, Toshiki Saitoh, Ryuhei Uehara, and Yushi Uno. Swapping colored tokens on graphs. In *WADS 2015 Proc.*, pages 619–628, 2015. doi:10.1007/978-3-319-21840-3_51.

33 Gaku Yasui, Kouta Abe, Katsuhisa Yamanaka, and Takashi Hirayama. Swapping labeled tokens on complete split graphs. *SIG Technical Reports*, 2015-AL-153(14):1–4, 2015.