Complexity of Token Swapping and its Variants*

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

1 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)**
2 **Faculty of Mathematics and Information Science, Warsaw University of Technology**
edouard.bonnet@dauphine.fr, t.miltzow@gmail.com, 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. That 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 different 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

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

A big family of problems in graph theory involves moving tokens along the edges of a given graph, in order to reach some final configuration [1, 4, 6, 15]. In this paper we study on of them, namely, the Token Swapping problem introduced by Yamanaka *et al.* [16]. The instance of this problem is a connected graph $G = (V, E)$ (where $|V| = n$) and an initial configuration of tokens given by a permutation $\pi_0 : V \to T$ with $T := \{t_v \mid v \in V\}$. For any vertex v of the graph, $\pi_0(v) = t_w$ is interpreted as „the token initially on v is t_w". Since π_0 is a permutation, there is exactly one token on each vertex. A *swap* operation consists of interchanging the tokens of two adjacent vertices. The *target configuration* satisfies that the token on each vertex $v \in V$ is precisely t_v. The Token Swapping problem asks if the target configuration can be reached in at most k swaps. Thus, a solution for the Token

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

Swapping problem is a sequence of edges, where the swaps take place. The solution is optimal if its length is shortest possible. Yamanaka *et al.* [16] observed 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 although the problem in its full generality was introduced only recently [16], some special cases were studied before in different contexts (see Knuth [10, Section 5.2.2] for paths, Pak [13] for stars, and Cayley [2] for cliques).

The complexity of the Token Swapping problem was investigated by Miltzow *et al.* [12]. They show that the problem is NP-complete and APX-complete. Moreover, they show that an algorithm solving the Token Swapping problem in time $2^{o(n)}$ would refute the ETH.

The results of Miltzow *et al.* [12] carry over also to some generalization of the Token Swapping problem, called Colored Token Swapping. 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. NP-hardness of Colored Token Swapping was first shown by Yamanaka*et al.* [17], even in the case that only 3 colours exist. In this setting, the goal is to reach, with the minimum number of swaps, a configuration in which each vertex contains a token of its color. Token Swapping corresponds to the special case where each color class comprises exactly one token and one vertex.

The Subset Token Swapping problem is even further generalization of Token Swapping where a function $D : T \to 2^V$ specifies the set $D(t)$ of possible destinations $D(t)$ for token t. 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.* [12]. As the main contribution, we show in Section 3 that Token Swapping is $W[1]$-hard parameterized by the number of swaps k.

▶ **Theorem 1** (Parameterized hardness). *Unless the ETH fails, for any computable function f,* Token Swapping *cannot be solved in time $f(k)(n)^{o(k/\log k)}$ where k is the allowed number of swaps, and n equals the number of vertices of the input graph.*

We even show that, under the ETH, the brute-force algorithm running in time $n^{O(k)}$ is almost the best possible. Let us explain why this result was not that predictable. Observe that if more than $2k$ tokens are misplaced, then one can claim that it is a NO-instance. Further, one can safely remove from the graph all the vertices that are at distance more than k of every misplaced tokens. This preprocessing yields a new graph whose connected components have *diameter* $O(k^2)$. The connected components of $f(k)$ *size* can be solved separately by exhaustively guessing (in FPT time) the number of swaps to perform in each of them. For those reasons, one could have hoped for an FPT algorithm. Indeed, the parameter for which we show hardness is in fact *number of swaps + number of initially misplaced tokens + diameter of the graph.*

For the proof of Theorem 1, we introduce *linker gadgets.* They are simple and can be used to give a significantly simpler proof of the lower bounds given by [12].

▶ **Theorem 2** (Exact Lower Bound [12]). Token Swapping *is NP-hard and is not solvable in $2^{o(n)}$ on graphs with n vertices, unless the ETH fails.*

It seems that the linker gadget can be used to show inapproximability lower bounds as well. A discussion can be found in Section 6.1 of the appendix.

Although there is no FPT algorithm for the Token Swapping problem (parameterized by k = number of swaps), unless FPT = $W[1]$, we show that FPT algorithms exist, if we restrict our input to the so-called *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 [7] proved, that every property definable as a first-order formula, is solvable in $O(n^{1+\varepsilon})$ time on nowhere-dense classes of graphs, for all $\varepsilon > 0$. The formal definition of nowhere-dense graphs is rather involved, so we refer the reader to the paper of Grohe, Kreutzer, and Siebertz [7].

▶ **Theorem 3.** Subset Token Swapping *is FPT parameterized by k on nowhere-dense graph classes.*

We defer this proof into Section 6.2 of the appendix.

We derive the following corollary.

▶ **Corollary 4.** 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 becomes tractable on classes of graphs with bounded treewidth (or, at least, with bounded tree-depth[1]); be it with FPT algorithms running in time $f(\mathrm{tw})n^{O(1)}$ (or $f(\mathrm{td})n^{O(1)}$) or XP algorithms in time $n^{f(\mathrm{tw})}$ (or $n^{f(\mathrm{td})}$) for some computable functions f. We rule out such algorithms by showing that Token Swapping remains hard when restricted to graphs with tree-depth 4 (treewidth and pathwidth 2, diameter 6, distance 1 to a forest).

▶ **Theorem 5** (Hardness for restricted graphs)**.** Token Swapping *remains NP-hard even when both the treewidth and the diameter of the input graph are constant.*

It might very well be that Token Swapping is NP-hard on trees. In contrast, we give an algorithm for Colored Token Swapping on stars using some connection to Eulerian graphs.

▶ **Theorem 6** (Colored Token Swapping on stars)**.** Colored Token Swapping *can be solved in polynomial time on stars.*

Complimentary to this, we can show that Subset Token Swapping is already hard on stars. This gives an example of a graph class that distinguishes Subset Token Swapping and Colored Token Swapping.

▶ **Theorem 7** (Subset Token Swapping on stars)**.** *On stars,* Subset Token Swapping *remains NP-hard and cannot be solved in time $2^{o(n)}$ unless the ETH fails, even for target sets of size at most 2.*

Recall that Token Swapping can be solved easily on cliques. The following theorem shows hardness of Colored Token Swapping on cliques. This gives an example of a graph class that distinguishes Token Swapping and Colored Token Swapping.

[1] We do not define tree-depth here since it is only a side observation.

▶ **Theorem 8.** *On cliques,* Colored Token Swapping *remains NP-hard and cannot be solved in time* $2^{o(n)}$ *unless the ETH fails.*

In order to get a more complete picture, we considered paths and got the following result.

▶ **Theorem 9.** Colored Token Swapping *can be solved in polynomial time on paths.*

In Section 5, we give a summary of our results and some open questions. The proofs of Theorem 6, 7 and Theorem 8 can be found in Section 6.3 of the appendix.

2 Preliminary results

Yamanaka *et al.* [16] showed, that in every instance of the Token Swapping problem, the length of the optimal solution is $O(n^2)$ and this bound is asymptotically tight for paths. Here we show that long induced paths are the only structures forcing solutions of superlinear length. We leave the proof as an excercise.

▶ **Proposition 10.** *The length of the optimal solution for* Token Swapping *in an* n*-vertex* P_{r+1}*-free graph* G *is at most* $r \cdot n$*.*

We observe that the bound from Proposition 10 holds also for Colored Token Swapping and Subset Token Swapping problems. Indeed, we can fix one destination for each of the tokens (by choosing a perfect matching in the token-destination graph) to obtain an instance of Token Swapping problem, whose solution is also the solution for the original problem.

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", thus we will call them *happy swaps*. Since each swap involves two tokens, we get the following lower bound.

▶ **Proposition 11.** *The length of the 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 with only happy swaps.*

Miltzow *et al.* [12] show that an optimal solution for the Token Swapping problem can be found by performing a breath-first-search on the *configuration graph* (i.e. a graph, whose vertices are all possible configurations of tokens on vertices, and two configurations are adjacent when we can obtain one from another with a single swap). We point out 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 final configuration.

For a later purpose, we define here the Multicolored Subgraph Isomorphism problem (see Figure 1). 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 $u_i u_j \in E(P) \Rightarrow \varphi(u_i)\varphi(u_j) \in E(H)$ and $\varphi(u_i) \in V_i$. Thus we can assume that each V_i forms an empty set. Further, we assume without loss of generality that $E_{i,j} := E(V_i, V_j)$ is non-empty iff $v_i v_j \in E(P)$. In other words, we try to find k vertices $v_1 \in V_1$, $v_2 \in V_2$, ...,

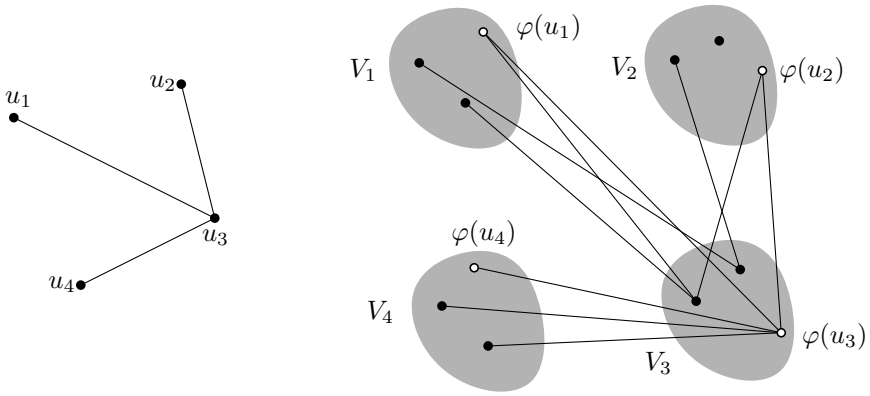

Figure 1 On the left is the pattern graph P; on the right, the host graph H. We indicate the image of φ with white vertices. To keep the example small, we did not make P 3-regular.

$v_k \in V_k$ such that, for any $i < j \in [k]$,[2] there is an edge between v_i and v_j iff $E_{i,j} := E(V_i, V_j)$ is non-empty. As shown by Marx [11], 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. In particular, k has to be an even integer since $|E(P)|$ is exactly $3k/2$. We finally assume that $\forall i \in [k]$, $|V_i| = t$, 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.

3 Lower Bounds on parameterized Token Swapping

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

▶ **Theorem 1** (Parameterized hardness). *Unless the ETH fails, for any computable function f,* Token Swapping *cannot be solved in time $f(k)(n)^{o(k/\log k)}$ where k is the allowed number of swaps, and n equals the number of vertices of the input graph.*

Linker gadget. To show parametrized 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. A quite direct usage of linker gadgets also gives an alternative proof of Theorem 2.

We describe the *linker gadget* and provide some intuitive reason why it works (see Figure 2). Given two integers, the linker gadget $L_{a,b}$ contains a set of a vertices, called *finishing set* and a path of length a, 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 2. 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

[2] Throughout the paper, we denote $\{1, \ldots, p\}$ by $[p]$ for any positive integer p

disjoint induced paths each of length a arranged in a grid, see Figure 2. 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.

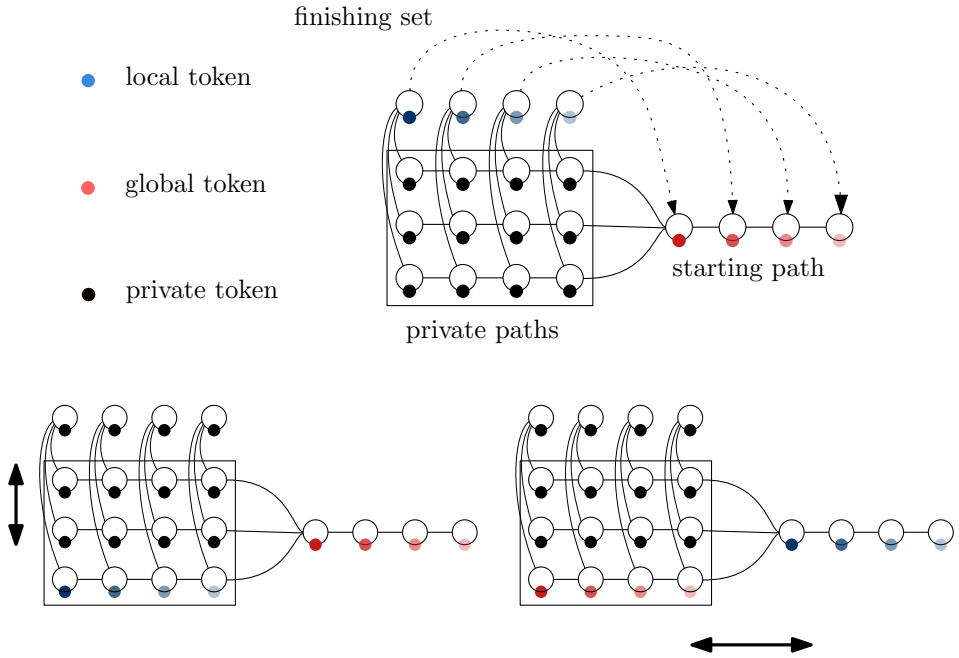

Figure 2 The linker gadget $L_{a,b}$. Black tokens are initially properly placed. Dashed arcs represent where tokens of the finishing set should go in the starting path. At the bottom left, we depict the gadget after all the local tokens are swapped to a single private path. At the bottom right, we see the result after swapping all the local tokens to the starting path. In this case, the global tokens go to that private path.

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[3] between the a local tokens and the a global tokens is to use a unique private path. That results in a kind of configuration as depicted in the bottom right of Figure 2, 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. For any color class $V_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,t}\}$ of the Multicolored Subgraph Isomorphism instance (H, P), 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. $L_{3,t}$ 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.

[3] The destination of the global tokens will make those swaps happy.

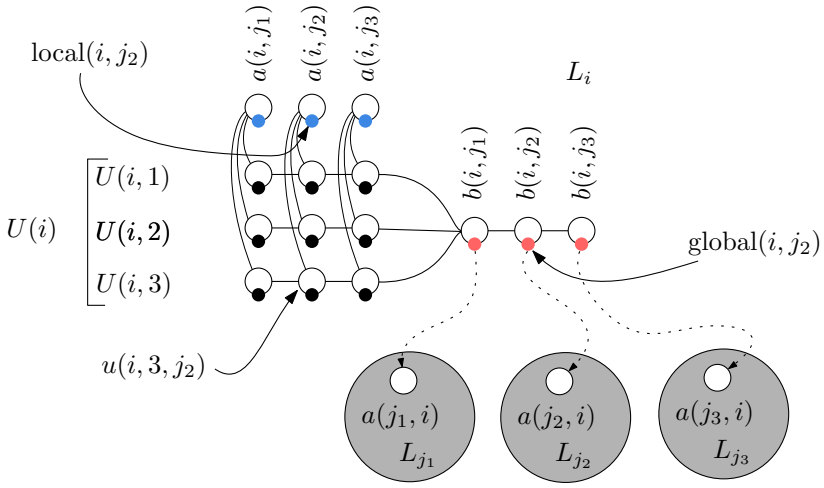

Figure 3 The different labels for tokens, vertices, and sets of vertices.

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

For each $p \in [3]$, local(i, j_p) shall go to vertex $b(i, j_p)$, whereas 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 $h \in [t]$, we name $\mathrm{U}(i, h)$ the h-th private path. This path represents the vertex $v_{i,h}$. The path $\mathrm{U}(i, h)$ is made 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.

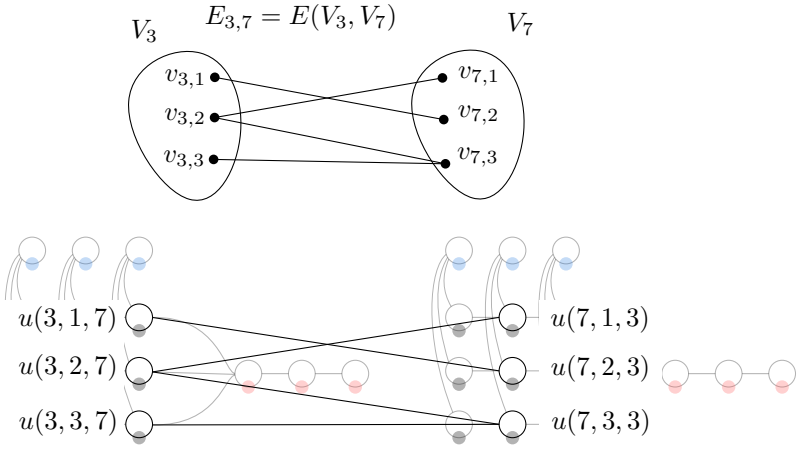

Figure 4 The way linkers (in that case, L_3 and L_7) are assembled together, with $t = 3$.

We complete the construction by adding every edge of the form $u(i, h, j)u(j, h', i)$ whenever $v_{i,h}v_{j,h'}$ is an edge in $E_{i,j}$ (see Figure 4). We call G the graph that we built and I the whole instance (with the initial position of the tokens). We ask for a solution of length at most $\ell := 16.5k = O(k)$. Recall that k is even, so $16.5k$ is an integer.

Correctness. ($\Rightarrow$) We first assume that there is a solution $\{v_{1,h_1}, v_{2,h_2}, \ldots, v_{k,h_k}\}$ to the Multicolored Subgraph Isomorphism instance. We perform the following sequence of swaps. The orderings that *we do not specify* among those swaps are not important; which means that they can be done in an arbitrary fashion. In each gadget L_i, we first bring local(i, j_3) to $b(i, j_3)$, then local(i, j_2) to $b(i, j_2)$, and finally local(i, j_1) to $b(i, j_1)$, each time passing through the same private path U(i, h_i). This corresponds to a total of 12 swaps per gadget and $12k$ swaps in total. Note that global(i, j_p) is moved to $u(i, h_i, j_p)$. Now, for each edge $v_{i,h_i}v_{j,h_j}$ of the host graph H (i.e., $u_iu_j \in E(P)$), we swap the tokens global(i, j) and global(j, i). By construction of G, $u(i, h_i, j)u(j, h_j, i)$ is indeed an edge in $E(G)$, so this swap is legal. This adds $3k/2$ swaps. At this point, the token global(j, i) is on vertex $u(i, h_i, j)$. Therefore, we move the token global(j, i) to vertex $a(i, j)$ in one swap. That corresponds to $3k$ additional swaps. Observe that it has also the effect of putting the private tokens back to their original private path. Thus, every token is well placed. The overall number of swaps of this solution is $12k + 3k/2 + 3k = 16.5k = \ell$.

($\Leftarrow$) We now assume that there is a solution $\boldsymbol{s}$ to Token Swapping of length at most ℓ. We define $Y := \{(i, j) \mid i \neq j \wedge u_iu_j \in E(P)\}$. Note that $(i, j) \in Y$ implies $(j, i) \in Y$, and $|Y| = 3k$. We compute $L(I)$ the sum of the distances *token to destination*. For any $(i, j) \in Y$, local(i, j) is at distance 4 of its destination $b(i, j)$ (via any private path). For any $(i, j) \in Y$, global(i, j) is at distance 5 of its destination $a(j, i)$ (following any private path of L_i, then an edge to gadget L_j, and a last edge to $a(j, i)$). The rest of the tokens are initially well-placed. Therefore, $L := L(I) = (4 + 5) \cdot 3k = 27k$. By Proposition 11, a solution for I is at least of length $13.5k$.

▶ **Lemma 12.** *In any solution $\boldsymbol{s}$ for I, at least $3k$ initially well-placed tokens have to move.*

Proof. There are $3k$ local tokens and each has a disjoint neighborhood from all the others. Further all tokens in their neighborhood are private tokens, which are already well placed. ◀

In solution $\boldsymbol{s}$, let x be the number of swaps between a well-placed token and a misplaced token (in the best case, (-1/+1)-swaps), and y the number of swaps between two well-placed tokens ((+1/+1)-swaps). Lemma 12 implies that $x + 2y \geqslant 3k$. Those $x + y$ swaps increase the sum of distances *token to destination* by $2y$; its value reaches $L+2y$. As $\ell \leqslant 16.5k$, there can only be at most $16.5k - (x + y) \leqslant 13.5k + y = \frac{L+2y}{2}$ other swaps. Therefore, all those swaps shall be happy. It also implies that, in each U(i) exactly 3 well-placed tokens move in solution $\boldsymbol{s}$. A last consequence is that all the swaps strictly worse than (-1/+1)-swaps (that is, (0/+1)-swaps and (+1/+1)-swaps) have to be swaps between two well-placed tokens.

▶ **Lemma 13.** *In any solution $\boldsymbol{s}$, no token local(i, j) leaves the gadget L_i.*

Proof. It should first be noted that the token local(i, j) can only increase its distance to its destination by leaving L_i. Let $j_1 < j_2 < j_3$ be such that $\forall l \in [3]$, $(i, j_l) \in Y$. The distance of local(i, j) to its destination is *its distance to $b(i, j_1)$ plus $l - 1$*. Besides, local(i, j) can only leave L_i via a vertex $u(i, h, j')$ with $h \in [t]$ and $(i, j') \in Y$. From this vertex, it can go to $u(j', h', i)$ for some $h' \in [t]$. Now, the distance of local(i, j) to $b(i, j_1)$ is 2 if $l = 3$, and at least 3 otherwise. In both cases, the swap that puts local(i, j) cannot be happy. Therefore, by the consequences of Lemma 12, it has to be a swap with a well-placed token. That means that this swap is at best a (0/+1)-swap. Which is only possible if it is a (+1/+1)-swap between two well-placed tokens; hence, a contradiction. ◀

▶ **Lemma 14.** *For every $i \in [k]$, the 3 tokens of U(i) which moved in solution $\boldsymbol{s}$, are in the same U(i, h_i), for some $h_i \in [t]$.*

Proof. Let $j_1 < j_2 < j_3$ such that $(i, j_1), (i, j_2)$, and (i, j_3) are all in Y. Consider the token $\text{local}(i, j_2)$. It first moves to a vertex $u(i, h_i, j_2)$ (for some $h_i \in [t]$). By Lemma 13, its only way to its destination $b(i, j_2)$ is via $u(i, h_i, j_3)$. Which means that the token initially well-placed on $u(i, h_i, j_3)$ is one of those 3 tokens of $\mathrm{U}(i)$ which moved. Now, by considering the token $\text{local}(i, j_1)$, the same argument shows that the three tokens of $\mathrm{U}(i)$ which are moved by solution $\boldsymbol{s}$ are $u(i, h_i, j_1)$, $u(i, h_i, j_2)$, and $u(i, h_i, j_3)$. ◀

We now claim that $\{v_{1,h_1}, v_{2,h_2}, \ldots, v_{k,h_k}\}$ is a solution to the Multicolored Subgraph Isomorphism instance. Indeed, for any $(i, j) \in Y$, $\text{global}(i, j)$ has to go to $a(j, i)$. By Lemma 14, it has to be via vertices of $\mathrm{U}(i, h_i)$ and $\mathrm{U}(j, h_j)$, and there is an edge between those two sets only if $v_{i,h_i}v_{j,h_j} \in E(H)$.

G has $3(t + 2)k$ vertices and $O(t^2k^2)$ edges. We recall that $\ell = O(k)$. Therefore, any algorithm solving Token Swapping in $f(\ell)(|V(G)| + |E(G)|)^{o(\ell/\log \ell)}$, for some computable function f, would also solve Multicolored Subgraph Isomorphism in time $f(k)(|V(H)|)^{o(k/\log k)}$; and would contradict the ETH. This finishes the proof of Theorem 1.

4 Token Swapping on very restricted classes of graphs

▶ **Theorem 5** (Hardness for restricted graphs). Token Swapping *remains NP-hard even when both the treewidth and the diameter of the input graph are constant.*

Proof. In 3-Exact Cover, one is given a set $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of m 3-element subsets over a universe $X = \{x_1, x_2, \ldots, x_n\}$, where n divides 3. The goal is to find $n/3$ subsets of $\mathcal{S}$ that partitions (or here, equivalently, covers) X. 3-Exact Cover remains NP-complete when each element belongs to *exactly* 3 sets of $\mathcal{S}$ [5]. We will reduce from that restriction of the problem. Thus, n the number of elements of the universe equals m the number of sets of $\mathcal{S}$.

Construction. For each set $S_j \in \mathcal{S}$, we add a *set gadget* consisting of a tree on 10 vertices (see Figure 5). In the set gadget, the four gray tokens should cyclically swap as indicated by the arrows: g_i^j shall go where $g_{i+1 \mod 4}^j$ is, for each $i \in [4]$. 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.

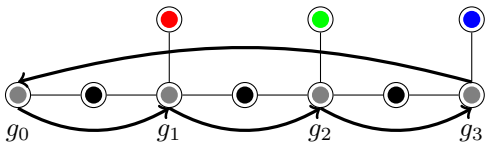

Figure 5 The set gadget for {●,●,●}. We voluntarily omit the superscript j.

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 6). The token initially on c is well placed (it could be drawn as a black token).

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 particular, treewidth at most 2. G has its diameter bounded by 6, since all the vertices are at

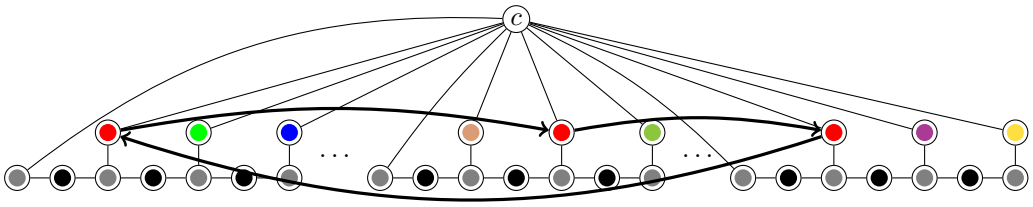

Figure 6 The overall picture. Each element appears exactly 3 times, so there are 3 red tokens.

distance at most 3 of the vertex c. We now show that the instance of 3-Exact Cover admits a solution iff the optimal number of swaps is no greater than $\ell := 11 \cdot n/3 + 9 \cdot 2n/3 + 2n = 35n/3 = 11n + 2n/3$.

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.*

▶ **Lemma 15.** *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.

▶ **Lemma 16.** *Swapping the gray tokens via c requires* 11 *swaps.*

Proof. Steps (a), (b), and (c) takes 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 3-Exact Cover 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 Lemma 16, 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 Lemma 15. 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 Lemma 15 and Lemma 16) 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 move only one *element* token to c (compared to moving all three of them in the strategy of Lemma 16). 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. Which means that this element belongs to one S_{a_i}. Hence, $S_{a_1}, \ldots, S_{a_{n/3}}$ is a solution to the instance of 3-Exact Cover. ◀

5 Conclusion

The Table 1 shows the current state of our knowledge about complexities of Token Swapping, Colored Token Swapping, and Subset Token Swapping problems, parameterized by k, and by the treewidth and the diameter of G.

	$k +$ diam	k, nowhere-dense	tw $+$ diam
Token Swapping	**W[1]-h** (Th 1)	FPT	**paraNP-c** (Th 5)
Colored Token Swapping	W[1]-h	FPT	paraNP-c
Subset Token Swapping	W[1]-h	**FPT** (Th 3)	paraNP-c

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

In Table 2 we summarize the complexities of Token Swapping, Colored Token Swapping, and Subset Token Swapping problems in restricted classes of graphs.

	trees	cliques	stars	paths
Token Swapping	?	P (see [12])	P (see [12])	P (see [12])
Colored Token Swapping	?	**NP-c** (Th 8)	**P** (Th 6)	**P** (Th 9)
Subset Token Swapping	NP-c	NP-c	**NP-c** (Th 7)	?

Table 2 The complexity of Token Swapping, Colored Token Swapping, and Subset Token Swapping on very restricted classes of graphs.

We believe that it would be interesting to fill the missing entries. In particular, we conjecture that the Token Swapping problem is NP-complete even for trees.

Another interesting problem is the following. By Miltzow *et al.* [12, 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 be also interesting to find single-exponential algorithms for some restricted graph classes, such as graphs with bounded treewidth or planar graphs.

References

1 G. Calinescu, A. Dumitrescu, and J. Pach. Reconfigurations in graphs and grids. In *LATIN 2006*, pages 262–273, Berlin, Heidelberg, 2006. Springer.

2 A. Cayley. LXXVII. Note on the theory of permutations. *Philosophical Magazine Series 3*, 34(232):527–529, 1849.

3 C. J. Colbourn. The complexity of completing partial latin squares. *Discrete Applied Mathematics*, 8(1):25 – 30, 1984.

4 R. Fabila-Monroy, D. Flores-Peñaloza, C. Huemer, F. Hurtado, J. Urrutia, and D. R. Wood. Token graphs. *Graphs and Combinatorics*, 28(3):365–380, 2012.

5 T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

6 D. Graf. How to sort by walking on a tree. In *ESA*, pages 643–655, Berlin, Heidelberg, 2015. Springer.

7 M. Grohe, S. Kreutzer, and S. Siebertz. Deciding first-order properties of nowhere dense graphs. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 89–98, New York, NY, USA, 2014. ACM.

8 I. Holyer. The NP-Completeness of Some Edge-Partition Problems. *SIAM J. Comput.*, 10(4):713–717, 1981.

9 R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

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

11 D. Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010.

12 T. Miltzow, L. Narins, Y. Okamoto, G. Rote, A. Thomas, and T. Uno. Approximation and hardness of token swapping. In *ESA*, 2016 (to appear).

13 I. Pak. Reduced decompositions of permutations in terms of star transpositions, generalized Catalan numbers and k-ARY trees. *Disc. Math.*, 204(1):329 – 335, 1999.

14 J. Plesník. The NP-Completeness of the Hamiltonian Cycle Problem in Planar Digraphs with Degree Bound Two. *Inf. Process. Lett.*, 8(4):199–201, 1979.

15 R. M. Wilson. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory, Series B*, 16(1):86 – 96, 1974.

16 K. Yamanaka, E. D. Demaine, T. Ito, J. Kawahara, M. Kiyomi, Y. Okamoto, T. Saitoh, A. Suzuki, K. Uchizawa, and T. Uno. Swapping labeled tokens on graphs. In *Fun*, pages 364–375. Springer, 2014.

17 K. Yamanaka, T. Horiyama, D. G. Kirkpatrick, Y. Otachi, T. Saitoh, R. Uehara, and Y. Uno. Swapping colored tokens on graphs. In *WADS 2015*, pages 619–628, 2015.

6 Appendix

6.1 A simpler NP- and APX-hardness proof.

Recently, Miltzow et al. gave a ten-page long intricate chain of reductions showing that Token Swapping is NP-hard, and even APX-hard and not solvable in subexponential time under the ETH [12]. It turns out that those results can be obtained by using linker gadgets. We sketch a much simpler construction.

▶ **Theorem 2** (Exact Lower Bound [12])**.** Token Swapping *is NP-hard and is not solvable in* $2^{o(n)}$ *on graphs with* n *vertices, unless the ETH fails.*

sketch. We reduce from 3Sat. By the Sparsification Lemma, we can assume that the number of clauses is linear in the number of variables [9]. Further, we can assume that each variable occurs a constant number of times. From this constant number of occurrences, one can go down to only three occurrences per variable with a linear reduction. Let then ϕ be a 3CNF-formula where each variable appears 3 times. We introduce a linker gadget $L_{2,3}$ for each variable and a linker gadget $L_{3,7}$ for each clause of ϕ. In each *variable* gadget, we have the choice between a *true* or a *false* assignment. Each *clause* gadget has 7 different choices corresponding to the 7 different satisfying assignments. Each *variable* gadget is linked to all *clause* gadgets that it appears in (therefore, every *clause* gadget is linked to the three *variable* gadgets it contains). Two choices are consistent if the variable assignment agrees with the chosen satisfying assignment of the clause. The correctness of this approach can be shown in the same way as the proof of Theorem 1. As each gadget has constant size, the reduction is linear. Thus any $2^{o(n)}$ algorithm would contradict the ETH. ◀

With some additional care, one can also get inapproximability results with the linker gadgets.

6.2 Proof of Theorem 3

▶ **Theorem 3.** Subset Token Swapping *is FPT parameterized by* k *on nowhere-dense graph classes.*

Proof. If we are able to express the Subset Token Swapping problem as a first-order formula, then the result follows immediately from the meta-theorem by Grohe, Kreutzer, and Siebertz [7].

Miltzow *et al.* [12] observed that if the length of an optimal solution is k, then at most $2k$ tokens are swapped. In our formula variables will denote vertices of G. The relation $edge(x, y)$ denotes the existence of an edge xy. The subsets of possible destinations of tokens will be represented by relation $target(x, y)$, which means that the vertex y is a possible destination for the token initially starting on vertex x. Moreover, each token will be identified by its initial position.

Let Φ_k denote the formula encoding the solution of the Subset Token Swapping problem with **exactly** k swaps. If we are interested in a solution using at most k swaps, it is given by $\Phi_{\leqslant k} = \bigvee_{i=1}^{k} \Phi_i$.

We use variables to represent:

1. the „traced" tokens $t_1, t_2, \ldots, t_{2k}$ that are involved in the solution (some of them may stay intact, if the solution uses less than $2k$ tokens),
2. the final positions $dest_1, dest_2, \ldots, dest_{2k}$ of the „traced" tokens ($dest_j$ is the final position of token t_j),

3. the swaps $s_1^1, s_1^2, \ldots, s_k^1, s_k^2$ (in the i-th swap we exchange the tokens on edge $s_i^1 s_i^2$),
4. the tokens that are swapped – by st_i^1, st_i^2 we denote the tokens that were swapped in the i-th swap, i.e. before the swap the position of st_i^p was s_i^p,
5. the positions of „traced" tokens in each round – $pos_{j,i}$ is the vertex, where token t_j is after i-th swap.

Now we are ready to present the formula.

$$\Phi_k = \exists(t_1, t_2, \ldots, t_{2k}) \tag{1}$$
$$\exists(dest_1, dest_2, \ldots, dest_{2k}) \tag{2}$$
$$\exists(st_1^1, st_1^2, st_2^1, st_2^2, \ldots, st_k^1, st_k^2) \tag{3}$$
$$\exists(s_1^1, s_1^2, s_2^1, s_2^2, \ldots, s_k^1, s_k^2) \tag{4}$$
$$\exists(pos_{1,0}, pos_{2,0}, \ldots, pos_{2k,0}) \tag{5}$$
$$\exists(pos_{1,1}, pos_{2,1}, \ldots, pos_{2k,1}) \tag{6}$$
$$\exists(pos_{1,2}, pos_{2,2}, \ldots, pos_{2k,2}) \tag{7}$$
$$\vdots \tag{8}$$
$$\exists(pos_{1,k}, pos_{2,k}, \ldots, pos_{2k,k}) \tag{9}$$
$$\forall(x)\Big(\bigwedge_{j=1}^{2k} x \neq t_j\Big) \to subset(x, x) \tag{10}$$
$$\wedge \bigwedge_{j=1}^{2k} \bigwedge_{j'=1}^{2k} (j \neq j' \to t_j \neq t_{j'}) \tag{11}$$
$$\wedge \bigwedge_{j=1}^{2k} subset(t_j, dest_j) \tag{12}$$
$$\wedge \bigwedge_{i=1}^{k} edge(s_i^1, s_i^2) \tag{13}$$
$$\wedge \bigwedge_{j=1}^{2k} pos_{j,0} = t_j \tag{14}$$
$$\wedge \bigwedge_{j=1}^{2k} pos_{j,k} = dest_j \tag{15}$$
$$\wedge \bigwedge_{i=1}^{k} \left(\bigvee_{j=1}^{2k} st_i^1 = t_j \wedge pos_{j,i} = s_i^1 \right) \tag{16}$$
$$\wedge \bigwedge_{i=1}^{k} \left(\bigvee_{j=1}^{2k} st_i^2 = t_j \wedge pos_{j,i} = s_i^2 \right) \tag{17}$$
$$\wedge \bigwedge_{i=1}^{k} \bigwedge_{j=1}^{2k} \big(\neg(st_i^1 = t_j \vee st_i^2 = t_j) \to pos_{j,i+1} = pos_{j,i}\big) \tag{18}$$
$$\wedge \bigwedge_{i=1}^{k} \bigwedge_{j=1}^{2k} \bigwedge_{j'=1}^{2k} \big((j \neq j' \wedge st_i^1 = t_j \wedge st_i^2 = t_{j'}) \to (pos_{j,i+1} = pos_{j',i} \wedge pos_{j',i+1} = pos_{j,i})\big) \tag{19}$$

In lines 1–9 we define the variables. Line 10 says that the tokens that are not involved in any

swaps are already at feasible positions. Line 11 ensures that the traced tokens are pairwise different. Lines 12 and 13 say that the final positions of traced tokens should be feasible, and we can perform swaps only on edges. In lines 14 and 15 we synchronize the values of variables $pos_{j,0}$ and $pos_{j,k}$ with variables t_j and $dest_j$. In lines 16 and 17 we synchronize the values of variables sp_i^1, sp_i^2 and s_i^1, s_i^2. In line 18 we make sure that the tokens that are not involved in the current swap, stay on their positions. Finally, in line 19, we say that the tokens involved in the current swap exchange their positions. ◀

6.3 Classic Cases

Now, let us consider the complexities of Token Swapping, Colored Token Swapping, and Subset Token Swapping problems, restricted to some very simple classes of graphs, i.e. stars, cliques, and paths. We will be interested in exploring the boundaries between easy (i.e. polynomially solvable) and hard (NP-hard) cases.

Let us start with a defining an auxiliary digraph. For the instance Colored Token Swapping problem on a graph G, we define the *color digraph* G^*, whose vertices are colors of tokens on G, and arcs correspond to vertices of G. The vertex v corresponds to the arc cc', such that c is the color of v and c' is the color of the token placed in v. Note that both loops and multiple arcs are possible. There is a very close relation between color digraphs and Eulerian digraphs.

▶ **Observation 17.** *The following hold:*

(i) if G^* is the color digraph of some instance of Colored Token Swapping *problem, then every connected component of G^* is Eulerian;*
(ii) for every Eulerian digraph H with n edges, there exists an instance of Colored Token Swapping *G on n vertices, such that its color digraph G^* is isomorphic to H.*

Proof. Too see (i), consider a vertex c of G^*. Its out-degree is the number of tokens placed on vertices with color c. The in-degree of c is the number of tokens in color c. Thus the in-degree is equal the out-degree, from which (i) follows.

Now, to see (ii), consider a vertex c of G^*, let d be its out-degree (equal to the in-degree, as G^* is Eulerian). Then in G give the color c to any d vertices. Moreover, for each arc cc' in G^* we place a token in color c' on a vertex in color c. We repeat this for every vertex c in G^*, obtaining an instance of Colored Token Swapping, whose color digraph is exactly G^*. ◀

Now consider a solution $\boldsymbol{s}$ for the instance of the Colored Token Swapping problem in G and fix the destinations of tokens according to $\boldsymbol{s}$. We observe that the cycles in the permutation defined by these destinations correspond to circuits in G^*. Thus, when trying to find a solution for an instance of Colored Token Swapping problem, we will first try to fix appropriate destinations (by analyzing circuits in G^*), and then we will solve the instance of Token Swapping problem.

▶ **Theorem 6** (Colored Token Swapping on stars)**.** Colored Token Swapping *can be solved in polynomial time on stars.*

Proof. Let G be a star with center v_0 and leaves $v_1, v_2, \ldots, v_n$. The color of the vertex v will be denoted by $c(v)$. Also, let $c_0 := c(v_0)$.

First, suppose that there exists a leaf v_i (for $i \geq 1$), such that the token that is initially placed there has color $c(v_i)$. It is easy to observe that in an optimal solution this token is

never swapped, so we can continue with the graph $G - v_i$. Thus, we assume that no leaf v_i has a token colored $c(v_i)$.

Consider the color digraph G^*. By the previous paragraph, we observe that with just one possible exception c_0c_0, it has no loops. Let $C_0, C_2, \ldots, C_m$ be the connected components of G^*, and let $c_0 \in C_0$. Moreover, for $i \geq 0$, by p_i we denote the number of arcs in C_i. By Observation 17(i), the edges of G^* can be decomposed (in polynomial time) into $m+1$ circuits (Eulerian circuits of its connected components).

Let $e(v_1^i), e(v_2^i), \ldots, e(v_{p_i}^i)$ be such a circuit for C_i, also we assume that $v_1^0 = v_0$ (i.e. we start the circuit for C_0 with the arc corresponding to v_0). We construct the swapping strategy $\boldsymbol{s}$ by concatenating sequences $\boldsymbol{s}_i$, defined as follows:

$$\boldsymbol{s}_i = \begin{cases} v_0v_2^0, v_0v_3^0, \ldots, v_0v_{p_0}^0 & \text{for } i = 0, \\ v_0v_1^i, v_0v_2^i, v_0v_3^i, \ldots, v_0v_{p_i}^i & \text{for } i = 0. \end{cases}$$

It is straightforward to verify that $\boldsymbol{s}$ is a solution for our problem and its length is $n + m$. We claim this solution is optimal.

To see this, consider any solution $\boldsymbol{s}'$. Let us consider the instance of Token Swapping problem obtained by fixing the destinations of all tokens, according to $\boldsymbol{s}'$. Let $q_0, q_1, \ldots, q_{m'}$ be the cycles in the permutation given by the destinations, and assume q_0 contains vertex v_0. By the result of Pak [13, Lemma 2.1], the length of the optimal solution of this instance of Token Swapping is exactly $n + m'$. We observe that the set of colors of vertices in each cycle has to be entirely contained in one of the components C_i, so $m' \geq m$, thus the length of $\boldsymbol{s}'$ is at least $n + m' \geq n + m$, which completes the proof. ◀

▶ **Theorem 7** (Subset Token Swapping on stars). *On stars,* Subset Token Swapping *remains NP-hard and cannot be solved in time $2^{o(n)}$ unless the ETH fails, even for target sets of size at most 2.*

Proof. We will reduce from the Directed Hamiltonian Cycle problem restricted to digraphs with out-degree at most 2, which is known to be NP-complete [14]. Moreover, it follows from the proof that the problem cannot be solved in time $2^{o(n)}$, unless the ETH fails (the original proof considers planar instances, but if we drop the planarity assumption, we obtain claimed lower bound).

Let $G = (V, E)$ be a digraph with all out-degrees at most 2, we can assume it has no loops. We will construct an instance $(G = (V', E'), D)$ of Subset Token Swapping with $|D(v)| \leqslant 2$ for all $v \in V'$, that has a solution of length at most $n + 1$ if an only if G has a Hamiltonian cycle.

The set V' is equal to $V \uplus \{c\}$ where c is the center of the star, and the leaves are the vertices of G. For each $v \in V' \setminus \{c\}$, we set $D(v) = N_G(v)$ (the set of out-neighbors of v in G) and $D(c) = \{c\}$.

Suppose G has a Hamiltonian cycle $v_1, v_2, v_3, \ldots, v_n$ (with v_1 adjacent to v_n). It is easy to observe that the sequence $cv_1, cv_2, \ldots, cv_n, cv_1$ of edges is a solution for the Colored Token Swapping problem with length $n + 1$.

On the other hand, suppose there is a solution $\boldsymbol{s}'$ for the Subset Token Swapping problem of length at most $n + 1$. Since G has no loops, every token starting at $v \in V$ must be moved to c at some point. Moveover, in the last swap we have to bring the token starting at c back to this vertex. Thus every feasible solution uses at least $n+1$ swaps, which implies that the length of $\boldsymbol{s}'$ is exactly $n+1$ (so let $\boldsymbol{s}' = cv_1, cv_2, \ldots, cv_n, cv_{n+1}$). Moreover, we have $v_1 = v_{n+1}$ and $v_i \neq v_j$ for all $1 \leq i < j \leq n$. Thus we observe that $v_1, v_2, v_3, \ldots, v_n$ is a Hamiltonian cycle in G. ◀

If G is a complete graph, then the optimal solution for the Token Swapping problem is n minus the number of cycles in the permutation given by initial positions of tokens [2]. Thus, the problem is solvable in polynomial time. On the other hand, we can show that Colored Token Swapping problem is NP-complete on complete graphs. Before we prove it, let us prove an auxiliary lemma.

▶ **Lemma 18.** *The problem of decomposing an arc set of a digraph $H = (V, A)$ into directed triangles is NP-complete, even if H is Eulerian and has no 2-cycles. Moreover, it cannot be solved in $2^{o(|A|)}$, unless the ETH fails.*

Proof. For a given 3-Sat formula Φ with N variables and M clauses, we will construct a digraph H, which can be decomposed into triangles if and only if Φ is satisfiabale.

The main part of the construction is essentially the same as the construction of Holyer [8], used to show NP-hardness of decomposing the edge set of an undirected graph into triangles (or, more generally, k-cliques). Thus we will just point out the modifications and refer the reader to the paper of Holyer for a complete description.

We observe that by the proper adjustment of constants the graph G_3 constructed by Holyer can be made three-partite (see also Colbourn [3]). Let A, B, C denote the partition classes. We obtain H by orienting all edges of G_3, according to the following pattern $A \to B \to C \to A$. Note that clearly H has no 2-cycles.

Consider a vertex v of G_3. Without loss of generality assume $c \in A$. We note that exactly half of the neighbors of v are in B, and the other half are in C. This implies that H is Eulerian.

We also point out that the number of arcs in H in linear in the number of vertices. Moreover, if we make the size of each variable gadget proportional to the number of occurrences of this variable in Φ (instead of proportional to M, as in the original proof), we obtain that $|A| = O(N + M)$. This shows that an existence of a subexponential (in $|A|$) algorithm for our problem contradicts the ETH. ◀

▶ **Theorem 8.** *On cliques,* Colored Token Swapping *remains NP-hard and cannot be solved in time $2^{o(n)}$ unless the ETH fails.*

Proof. Let H be an Eulerian directed graph with n arcs, having no 2-cycles. Consider an instance of Colored Token Swapping problem on $G = K_n$, such that H is its color digraph (it exists by Observation 17(ii)). We claim that there exists a solution for this instance of length at most $2n/3$ if and only if the arc set of H can be decomposed into directed triangles (see Lemma 18).

Suppose that the arc set of H can be decomposed into $n/3$ triangles. The vertices of G corresponding to the edges of the i-th triangle, are v_1^i, v_2^i, v_3^i. We construct the solution $\boldsymbol{s}$ by concatenating sequences $v_1^i v_2^i, v_1^i v_3^i$ for $i = 1, 2, \ldots, n/3$. It is easy to verify that $\boldsymbol{s}$ is a solution and its length is $2n/3$.

So now suppose we have a solution $\boldsymbol{s}$ of length at most $2n/3$. Recall that the length of any solution $\boldsymbol{s}'$ is at least n minus the number of cycles in the permutation obtained by fixing the destinations of tokens according to $\boldsymbol{s}'$. Thus the number of cycles in the permutation given by $\boldsymbol{s}$ is at least $n/3$. Since these cycles correspond to circuits in the color digraph H, and H has no 2-cycles, this is only possible if the arcs of H can be decomposed into triangles. ◀

It is interesting to point out that if G is a clique, then the presence of many cycles in the permutation of tokens yields a short solution for the Token Swapping problem, while for the case when G is a star, the situation is opposite. Finally, we turn our attention to paths.

▶ **Theorem 9.** Colored Token Swapping *can be solved in polynomial time on paths.*

Proof. Let c be the color of the vertex v at the left end of the path. Let t be the leftmost token with color c. It is clear that no optimal solution contains a swap involving two tokens of the same color, so in any optimal solution the token t will end up in v. Repeat this argument with the second leftmost vertex, and so on. This way we fix the destinations for all tokens, obtaining an equivalent instance of Token Swapping problem, which can be solved in polynomial time (see [12]). ◀