Complexity of token swapping and its variants

Edouard Bonnet, Miltzow Tillmann, Pawel Rzążewski

Algorithmica manuscript No.
(will be inserted by the editor)

Complexity of token swapping and its variants

Édouard Bonnet · Tillmann Miltzow · Paweł Rzążewski

Received: date / Accepted: date

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,

The extended abstract of this paper was presented at STACS 2017 [3]. The research was partially supported by the ERC grant PARAMTIGHT: "Parameterized complexity and the search for tight complexity results", no. 280152.

É. Bonnet
Middlesex University, Department of Computer Science, London, UK
E-mail: edouard.bonnet@dauphine.fr

T. Miltzow
Université libre de Bruxelles (ULB), Brussels, Belgium
E-mail: t.miltzow@gmail.com

P. Rzążewski
Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland
E-mail: p.rzazewski@mini.pw.edu.pl

and paths, trying to identify the borderlines between polynomial and NP-hard cases.

1 Introduction

In reconfiguration problems, we are interested to transform 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 motion planning can be reduced to the shortest path problem in some higher dimensional Euclidean space with obstacles [8].

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 [1, 5, 9, 11, 14, 16, 23, 31, 35]. In this paper, we study one of them.

The Token Swapping problem, introduced by Yamanaka *et al.* [36], fits nicely into this long history of reconfiguration problems and can be regarded as a sorting problem with special constraints.

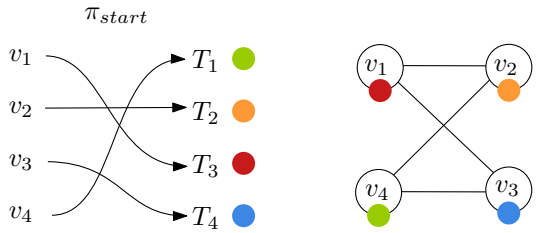

Fig. 1 Every token placement can be uniquely described by a permutation.

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, respectively. 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 TOKEN SWAPPING 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.

Yamanaka *et al.* [36] 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 [36], some special cases were studied before in the context of sorting permutations with additional restrictions (see Knuth [24, Section 5.2.2] for paths, Pak [30] for stars, Cayley [6] for cliques, and Heath and Vergara [19] for squares of a path). Recently the problem was also solved for a special case of complete split graphs (see Gaku *et. al.* [38]). It is also worth mentioning that a very closely related concept of sorting permutations using cost-constrained transitions was considered by Farnoud, Chen, and Milenkovic [13], and Farnoud and Milenkovic[12].

The computational complexity of TOKEN SWAPPING was investigated by Miltzow *et al.* [28]. They show that the problem is NP-complete and APX-complete. Moreover, they show that any algorithm solving TOKEN SWAPPING in time $2^{o(n)}$ would refute the Exponential Time Hypothesis (ETH) [21]. The results of Miltzow *et al.* [28] carry over also to a generalization of TOKEN SWAPPING, called COLORED TOKEN SWAPPING, first introduced by Yamanaka *et al.* [37]. 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 question is whether k swaps are enough to reach 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.* [37], even in the case that only 3 colors exist.

We introduce SUBSET TOKEN SWAPPING, which is an even further generalization of TOKEN SWAPPING. Here a function $D : T \to 2^V$ specifies the set $D(t)$ of possible destinations for the token t. We ask if k swaps are enough to reach a configuration, when each token t is placed on a vertex from $D(t)$. 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.* [28]. They presented a very simple algorithm which solves the instance of TOKEN SWAPPING 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. Next, we present a slightly slower exact algorithm, whose advantage is only polynomial (in fact, only slightly super-linear) space complexity.

The algorithm by Miltzow *et al.* [28] shows that Token Swapping is in XP. A natural next step is to investigate whether the problem can be solved in FPT time (i.e., $f(k) \cdot n^{O(1)}$, for some function f). There is some evidence indicating that this could be possible. First, observe that if more than $2k$ tokens are misplaced, then one can immediately answer that we deal with a No-instance, as each swap involves exactly 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 for bounded maximum degree Δ each component has size $f(k)$, for some function f. 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 3). 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 not possible.

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.* [28]. One might also use them to establish a simpler and potentially stronger inapproximability result.

Since there is no FPT algorithm for Token Swapping (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 [17] 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 Subset Token Swapping, 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 [29, 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.*

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.

	$k + \Delta$	$k + \mathrm{diam}$	k, nowhere-dense / $k + \mathrm{tw}$	$\mathrm{tw} + \mathrm{diam}$
TS	FPT ([28])	**W[1]-h** (Th 1)	FPT	**paraNP-c** (Th 4)
CTS	FPT	W[1]-h	FPT	paraNP-c
STS	**FPT** (Prop 3)	W[1]-h	**FPT** (Th 2)	paraNP-c

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

While we think that our results give a fairly detailed view on the complexity landscape of Token Swapping, we also want to point out that our reductions are significantly simpler than those by Miltzow *et al.* [28].

Since the investigated problems seem to be immensely intractable, in Section 7 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 on cliques Token Swapping is in P, while Colored Token Swapping (and thus also Subset Token Swapping) is NP-hard. On the other hand, on stars Colored Token Swapping (and thus also Token Swapping) is in P and Subset Token Swapping is NP-hard.

	trees	cliques	stars	paths
TS	?	P (see [28])	P (see [28])	P (see [28])
CTS	?	**NP-c** (Th 10)	**P** (Th 8)	**P** (Th 12)
STS	NP-c	NP-c	**NP-c** (Th 9)	NP-c [18]

Table 2 The complexity of Token Swapping (TS), Colored Token Swapping (CTS), and Subset Token Swapping (STS) on very restricted classes of graphs.

The paper is concluded with several open problems in Section 8.

2 Preliminaries

Yamanaka *et al.* [36] showed that in every instance of Token Swapping, 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.

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

Proof. We can assume that G is connected, since otherwise we can solve the problem on connected components separately. Let P be the longest path in G and let v be its end-vertex. Observe that $G - v$ is connected (otherwise P is not longest) and P_{r+1}-free. First, we move the token with destination v to this vertex, which requires at most $\mathrm{diam}(G) \leqslant r$ swaps. Then we can recursively continue with the graph $G - v$ (we never touch v again). Such a solution has length at most $r \cdot n$. □

Note that this bound is asymptotically tight – to see this, consider a graph, whose every connected component is isomorphic to P_r and has the reverse permutation of tokens (if we want to have our instance connected, we can add one additional vertex, adjacent to one of the end-vertices of each path, and put a well-placed token on it). Moreover, we observe that the bound from Proposition 1 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, 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 Token Swapping, 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 2 ([28]) *The length of the optimal solution for an instance I of* Token Swapping *is at least $L(I)/2$. Besides, it is* exactly *$L(I)/2$ if and only if there is a solution using happy swaps only.*

When designing algorithms, especially for computationally hard problems, it is natural to ask about lower bounds. However, the standard complexity assumption used for distinguishing easy and hard problems, i.e., P $\neq$ NP, is too weak to tell us something meaningful about possible complexities of algorithms. The stronger assumption that is typically used for this purpose is the so-called *Exponential Time Hypothesis* (usually referred to as the ETH), formulated by Impagliazzo and Paturi [21]. We refer the reader to the survey by Lokshtanov, Marx, and Saurabh for more information about ETH and conditional lower bounds [25]. 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 [22]).

Exponential Time Hypothesis (Impagliazzo and Paturi [21])**.** *There is no algorithm solving every instance of* 3-Sat *with N variables and M clauses in time $2^{o(N+M)}$.*

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.* [28] for Token Swapping. Furthermore, we show that the simple algorithm for Token Swapping, presented by Miltzow *et al.* [28], 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 3 Subset Token Swapping *is FPT in $k + \Delta$ and admits a kernel of size $2k + 2k^2 \cdot \Delta^k$.*

Proof. Let I be an instance of Subset Token Swapping on a graph G with maximum degree Δ and suppose I has a solution $\boldsymbol{s}$ of length at most k.

Let V_m be the set of such vertices v of G, that the token initially placed on v does not accept v as its destination. First, observe that every vertex from

V_m has to be involved in some swap in $\boldsymbol{s}$. Thus we can assume that $|V_m| \leq 2k$ (otherwise we immediately report a No-instance).

Let E' be the set of edges that appear in $\boldsymbol{s}$ and let G' be the subgraph of G induced by E'. Consider a connected component C of G'. Suppose first that the vertex set of C does not contain any vertex from V_m. Observe that the sequence $\boldsymbol{s}'$ obtained from $\boldsymbol{s}$ by removing all edges from C is also a solution for I of length at most k. So, without loss of generality, every connected component C of G' contains a vertex from V_m, and has at most k edges. Let G'' be the subgraph of G induced by the vertices at distance at most k from V_m (we find it by running a breadth-first search, starting from V_m). We observe that every vertex incident to an edge in E' is in G''. Thus the instance I' of Subset Token Swapping, restricted to G'', is equivalent to I. Note that the maximum degree of G'' is at most Δ, and the number of vertices in G'' is at most $2k + 2k^2\Delta^k$. Thus I' is a kernel for I. □

Miltzow *et al.* [28] show that an optimal solution for Token Swapping 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 observe that the same approach works for Colored Token Swapping and Subset Token Swapping, the only difference is that we terminate on any feasible target configuration.

Proposition 4 *Let G be a graph with n vertices, and let k be the maximum number of allowed swaps. The* Colored Token Swapping *and the* Subset Token Swapping *problems on G can be solved in time:*

- $O(n! \cdot n^2) = 2^{O(n \log n)}$,
- $n^{O(k)} = 2^{O(k \log n)}$,

using exponential space. □

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 [34].

Theorem 5 *Let G be a graph with n vertices, and let k be the maximum number of allowed swaps.* Subset Token Swapping *on G can be solved in time $2^{O(n \log n \log k)} = 2^{O(n \log^2 n)}$ and space $O(n \log n \log k) = O(n \log^2 n)$.*

Proof. Consider the algorithm `Reach` (see Algorithm 1). It is easy to verify that it returns *true* if the configuration π_s can be reached from the configuration π_0 with **exactly** k swaps, and *false* otherwise.

The depth of the recursion is $O(\log k)$. The configurations can be generated with polynomial delay, using only linear (in n) memory. Thus the time complexity of the algorithm is $n!^{\log k} \cdot n^{O(1)} = 2^{O(n \log n \log k)}$. The space needed to keep track of the recursive stack is $O(n \log n \log k)$. Recall that $k = O(n^2)$ – otherwise we immediately report a Yes-instance.

Algorithm 1: Reach(G, π_0, π_s, k)

Input: $G = (V, E)$ – a graph, π_0, π_s – configurations of tokens on G, integer $k \geq 0$

```
if k = 0 then
    if π_0 = π_s then return true
    else return false
if k = 1 then
    foreach e ∈ E do
        if π_s can be obtained from π_0 with a swap on e then
            return true
    return false
else
    foreach configuration π' of tokens on G do
        if Reach(G, π_0, π', ⌈k/2⌉) = true and Reach(G, π', π_s, ⌊k/2⌋) = true then
            return true
    return false
```

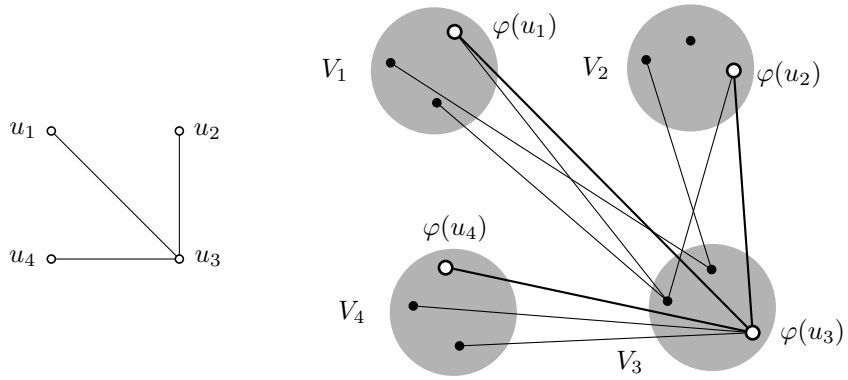

Fig. 2 On the left is the pattern graph P; on the right, the host graph H. We indicate the image of φ with white vertices. To keep the example small, we did not make P 3-regular.

To use the algorithm for Subset Token Swapping, we can enumerate all possible target configurations in $n! \cdot n^{O(1)} = 2^{O(n \log n)}$ time and polynomial space, and then solve the instance of Token Swapping for each of them. □

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).

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 $u_i u_j \in E(P)$ 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 if and only if $u_iu_j \in E(P)$. 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 $i < j \in [k]$,[1] there is an edge between v_i and v_j if and only if $E(V_i, V_j)$ is non-empty. The $W[1]$-hardness of Multicolored Subgraph Isomorphism problem follows from the $W[1]$-hardness of the Multicolored Clique. Marx [26] 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 [27]). 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. To show parameterized hardness of Token Swapping, 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.

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

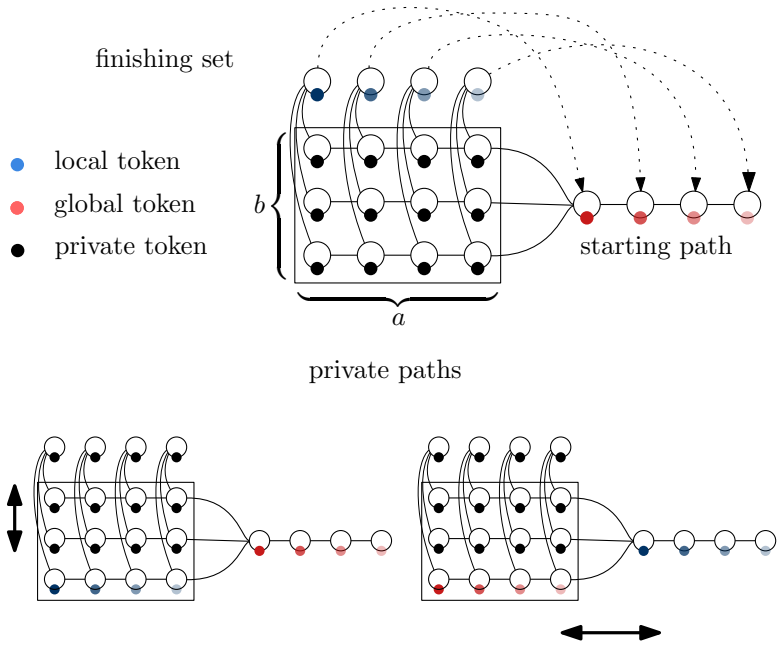

Fig. 3 The linker gadget $L_{a,b}$. Black (private) tokens are initially properly placed. Dashed arcs represent destinations of tokens of the finishing set (they all go to the starting path). In the intended solution, all local tokens are moved to a single private path (bottom left). Next, they are swapped with the tokens on the starting path (bottom right). 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 between the a local tokens and the a global tokens is to use a unique private path (note that the destination 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.

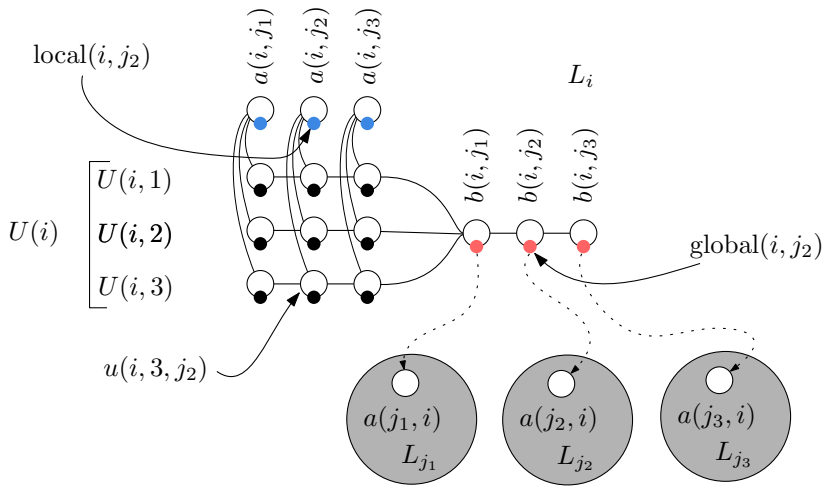

Fig. 4 The different labels for tokens, vertices, and sets of vertices.

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 $h \in [t]$, 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 an edge $u(i, h, j)u(j, h', i)$ whenever $v_{i,h}v_{j,h'}$ is an edge in $E(V_i, V_j)$ (see Figure 5). Let G be the graph that we built, and let I 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. ($\Rightarrow$) 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 $\mathrm{U}(i, h_i)$. 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

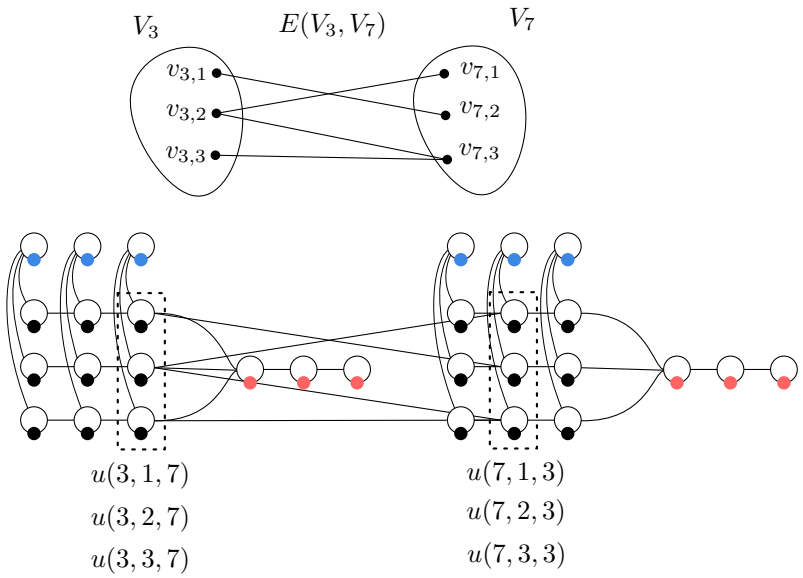

Fig. 5 The way linkers (in that case, L_3 and L_7) are assembled together, with $t = 3$.

point, the token global(j, i) is on vertex $u(i, h_i, j)$. Therefore, we move each token global(j, i) to the vertex $a(i, j)$ in one swap. This 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 now well placed. The overall number of swaps in 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 $Y := \{(i, j) \mid u_i u_j \in E(P)\}$. Note that $(i, j) \in Y$ implies $(j, i) \in Y$, and $|Y| = 3k$. We compute the sum $L(I)$ 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 2, the length of any solution for I is at least $13.5k$.

Claim 1. *In any solution $\boldsymbol{s}$ for I, at least $3k$ initially well-placed tokens have to move.*

Proof of Claim 1. There are $3k$ local tokens and each has a disjoint neighborhood from all the others. Furthermore, 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). Claim 1 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 $\mathrm{U}(i)$ 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.

Claim 2. *In any solution $\boldsymbol{s}$, no token local(i, j) leaves the gadget L_i.*

Proof of Claim 2. 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 $(i, j_l) \in Y$ for every $l \in [3]$. 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 $b(i, j_l)$ 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 Claim 1, it has to be a swap with a well-placed token. That means that this swap is at best a $(0/+1)$-swap. This is only possible if it is a $(+1/+1)$-swap between two well-placed tokens; hence, a contradiction. ■

Claim 3. *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 of Claim 3. 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 local(i, j_2). It first moves to a vertex $u(i, h_i, j_2)$ (for some $h_i \in [t]$). By Claim 2, its only way to its destination $b(i, j_2)$ is via $u(i, h_i, j_3)$. This 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 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 $(i, j) \in Y$, global(i, j) has to go to $a(j, i)$. By Claim 3, 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 $v_{i,h_i} v_{j,h_j} \in E(H)$.

The graph 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 time $f(\ell)(|V(G)|+|E(G)|)^{o(\ell/\log \ell)}$, for some computable function f, could be used to solve Multicolored Subgraph Isomorphism in time $f'(k)(|V(H)|+|E(H)|)^{o(k/\log k)}$; and would contradict the ETH. This completes the proof of Theorem 1. □

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 Token Swapping (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.

To define nowhere-dense graphs, first let us introduce a notion of a *shallow minor*. A shallow minor of a graph G at depth d is a subgraph of a graph obtained from G by contracting subgraphs of G, each of radius at most d, into single vertices, and removing loops and multiple edges. A class $\mathcal{G}$ is nowhere-dense if for every d the class of shallow minors at depth d of graphs in $\mathcal{G}$ has bounded clique number. For more information about this topic, we refer the reader to the comprehensive book of Nešetřil and Ossona de Mendez [29, Chapter 13].

As graphs with bounded degree are nowhere-dense, this result generalizes Proposition 3.

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

Proof. If we are able to express Subset Token Swapping as a first-order formula, then the result follows immediately from the meta-theorem by Grohe, Kreutzer, and Siebertz [17].

Theorem 6 (Grohe, Kreutzer, and Siebertz [17]) *For every nowhere-dense class C and every $\varepsilon > 0$, every property of graphs definable by a first-order formula φ can be decided in time $O(f(|\varphi|, \varepsilon) \cdot n^{1+\varepsilon})$ on C, where f is some function depending only on φ and ε.*

We will define the instance of Subset Token Swapping as a first-order formula $\Phi_{\leq k}$ of size $O(k^4)$. Recall 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 Subset Token Swapping 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 in the i-th swap for $i = 1, 2, \ldots, k$ – by st_i^1, st_i^2 we denote the tokens that were swapped in the i-th swap, i.e., st_i^p denotes the token on vertex s_i^p before performing the i-th swap,

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 Φ_k.

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

$$\wedge \bigwedge_{i=1}^{k} \bigwedge_{j=1}^{2k} \bigwedge_{j'=1}^{2k} \tag{19}$$

$$\big((j \neq j' \wedge st_i^1 = t_j \wedge st_i^2 = t_{j'}) \rightarrow (pos_{j,i+1} = pos_{j',i} \wedge pos_{j',i+1} = pos_{j,i})\big) \tag{20}$$

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 and 20 , we say that the tokens involved in the current swap exchange their positions. □

We derive the following corollary.

Corollary 3. 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. Therefore 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, we are given a finite family, denoted by $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, 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 [2, 15]. 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.

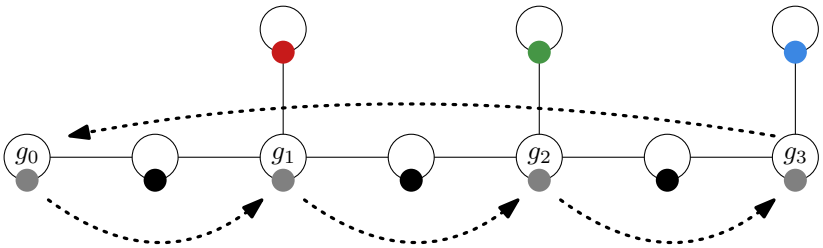

Fig. 6 The set gadget for red, green and blue. We voluntarily omit the superscript j.

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

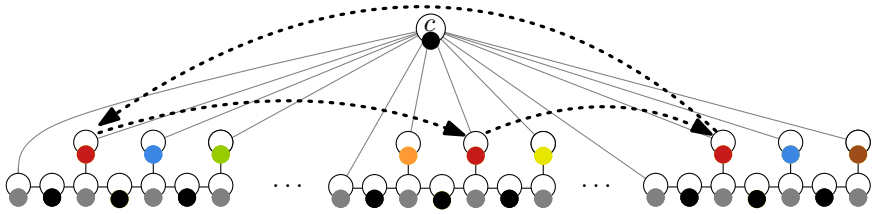

Fig. 7 The overall picture. Each element appears exactly 3 times, so there are 3 red tokens.

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 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 if and only if 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 4. *Swapping the gray tokens internally requires* 9 *swaps.*

Proof of Claim 4. 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 5. *Swapping the gray tokens via c requires* 11 *swaps.*

Proof of Claim 5. 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 5, 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 4. 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 4 and Claim 5) 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 5). 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 corresponding 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 linear in n. □

7 Variants of Token Swapping on stars, cliques, and paths

In this section we investigate the complexities of the variants of Token Swapping on very simple classes of graphs.

Let us start with defining an auxiliary digraph, which will be useful in coping with Colored Token Swapping. For an instance of Colored Token Swapping 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 $e(v) = cc'$, 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 7. *The following hold:*

(i) if G^* is the color digraph of some instance of Colored Token Swapping*, then every connected component of G^* is Eulerian;*

(ii) for every Eulerian digraph H with n edges, and for any graph G with n vertices, there exists an instance of Colored Token Swapping *on G, such that its color digraph G^* is isomorphic to H.*

Proof. To 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 Colored Token Swapping 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, we will first try to fix appropriate destinations (by analyzing circuits in G^*), and then we will solve the instance of Token Swapping.

7.1 Stars

To prove the next theorem we will use the following result by Pak [30]. We state in the language of tokens and swaps, although the original motivation of Pak was sorting a permutation by transpositions with the first element.

Lemma 1 (Pak [30]) *Let I be an instance of* Token Swapping *on a star with n leaves, with the initial configuration of tokens π. If the decomposition of π into cycles consists of one cycle involving the central vertex, m cycles of length at least 2, and b cycles of length 1, then the length of an optimal solution to I is $n + m - b$.*

Theorem 8 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, such that the token t that is initially placed there has color $c(v)$ as well. Let $\boldsymbol{s}$ be an optimal solution and consider a permutation π of tokens given by $\boldsymbol{s}$. We want to show that $\pi(v) = v$. Using the solution of Pak [30], this implies that t is never swapped.

For the purpose of contradiction, suppose $\pi(v) \neq v$. Then, there exists a token t' initially on vertex u with $\pi(u) = v$ and a vertex w with $\pi(v) = w$. In other words, token t ends at vertex w. So neither u, v nor w is involved in a 1-cycle, but all three vertices must have the same color. Thus we can alter this solution to a new permutation π', by setting $\pi(v) = v$ and $\pi(u) = w$. This increases the number b of 1-cycles by 1 and the number m stays the same. This contradicts the optimality of $\boldsymbol{s}$ by Lemma 1 and we conclude $\pi(v) = v$. Thus for any leaf v with a token t of color $c(v)$ holds, that the solution does not change after removal of v. Thus from now on we assume that no leaf v contains a token colored with color $c(v)$.

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 7(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 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 Lemma 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 9 *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 [32]. 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 Colored Token Swapping with length $n + 1$.

On the other hand, suppose there is a solution $\boldsymbol{s}'$ for Subset Token Swapping 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. Moreover, 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 $n + 1$; 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. □

7.2 Cliques

If G is a complete graph, then the optimal solution for Token Swapping is n minus the number of cycles in the permutation given by initial positions of tokens [6]. Thus, the problem is solvable in polynomial time. On the other hand, we can show that Colored Token Swapping is NP-complete on cliques. Before we prove it, let us prove an auxiliary lemma. In the Directed Triangle Decomposition we are given a digraph $H = (V, A)$, and we ask whether the arc set A can be decomposed into disjoint directed triangles.

Lemma 2 Directed Triangle Decomposition *is* NP*-complete, even if the input digraph $H = (V, A)$ 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 satisfiable.

The main part of the construction is essentially the same as the construction of Holyer [20], 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 [7]). 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 $v \in A$. 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 is 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 10 *On cliques,* Colored Token Swapping *remains* NP*-hard and cannot be solved in time $2^{o(n)}$, unless the ETH fails.*

Proof. We reduce from Directed Triangle Decomposition. Let H be an Eulerian directed graph with n arcs, having no 2-cycles. Consider an instance of Colored Token Swapping on $G = K_n$, such that H is its color digraph (it exists by Observation 7(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 2).

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 Token Swapping, while for the case when G is a star, the situation is opposite.

Theorems 9 and 10 can be used to show a slightly more general hardness result. A class $\mathcal{G}$ of graphs is *hereditary*, if for any $G \in \mathcal{G}$ and any induced subgraph G' of G we have $G' \in \mathcal{G}$. We say that that a class $\mathcal{G}$ of graphs has unbounded degree, if for every $d \in \mathbb{N}$ there exists $G \in \mathcal{G}$, such that $\Delta(G) \geq d$.

Theorem 11 *Let $\mathcal{G}$ be a hereditary class containing an infinite number of connected graphs with unbounded degree.* Subset Token Swapping *is* NP*-complete, when restricted to graphs from $\mathcal{G}$. Moreover, if there exists an algorithm solving* Subset Token Swapping *in time $2^{o(n)}$ for every graph in $G \in \mathcal{G}$ with n vertices, then the ETH fails.*

Proof. We shall reduce from Directed Hamiltonian Cycle in digraphs with out-degree at most 2. Let H be such a digraph with n vertices.

First, assume that $K_{1,n} \in \mathcal{G}$. Then we are done by Theorem 9. So assume that $K_{1,n} \notin \mathcal{G}$. Since $\mathcal{G}$ is hereditary, we know that $K_{1,n'} \notin \mathcal{G}$ for any $n' \geq n$. Since decomposing the arc set of an Eulerian digraph with no 2-cycles into directed triangles is NP-complete (see Lemma 2), there exists a polynomial reduction from Directed Hamiltonian Cycle to this problem. Consider the digraph H^* obtained with this reduction. Its arc set can be decomposed into triangles if and only if H has a Hamiltonian cycle. Let m denote the number of edges in H^* and set $N = \max(m, n)$.

By Ramsey theorem [33] (see also Erdős, Szekeres [10]) we know that there exists an absolute constant c such that every graph with more than $c \cdot 4^N$ vertices has either a clique or an independent set of size N.

Since $\mathcal{G}$ has unbounded degree, there exists a graph $G \in \mathcal{G}$, such that $\Delta(G) \geq c \cdot 4^N$. Let v be a vertex of G with degree at least $c \cdot 4^N$ and let G' be a subgraph of G induced by the neighborhood of v. If G' has an independent set U of size N, then $G[U \cup \{v\}] \sim K_{1,N}$, so we obtain a contradiction (recall that $\mathcal{G}$ is hereditary). Thus G' has a subset C inducing a clique of size N. Since $\mathcal{G}$ is hereditary and $N \geq m$, we obtain that $K_m \in \mathcal{G}$. Thus we can use the construction from Theorem 10. □

7.3 Paths

Finally, we turn our attention to paths.

Theorem 12 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, which can be solved in polynomial time (see [28]). □

Now we will discuss the complexity of Subset Token Swapping on paths. We want to point out an equivalent, interesting formulation of this problem. Consider an instance $\mathcal{I}$ of Subset Token Swapping defined on a path with n vertices $v_1, v_2, \ldots, v_n$. For a vertex v_i, let t_i denote the token initially placed on v_i, and let $D(t_i)$ denote the set of possible destinations of t_i. Now consider a bipartite graph G with bipartition classes $\{v_1, v_2, \ldots, v_n\}$ and $\{t_1, t_2, \ldots, t_n\}$. The edge $t_i v_j$ is present in G if and only if $v_j \in D(t_i)$. Fix two distinct vertical lines ℓ and ℓ' on a plane and fix the positions of vertices of G on these lines; $v_1, v_2, \ldots, v_n$ lie on ℓ (in this ordering from top to bottom), and $t_1, t_2, \ldots, t_n$ lie on ℓ' (also in this ordering from top to bottom); see Figure 8.

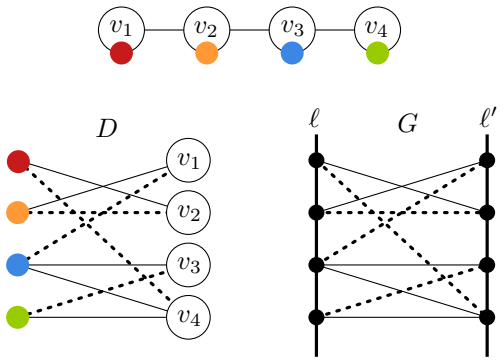

Fig. 8 A bipartite graph G constructed from an instance of Subset Token Swapping on a path.

Consider a feasible solution $\boldsymbol{s}$ of $\mathcal{I}$ and let σ be the permutation assigning destinations to tokens, according to $\boldsymbol{s}$. Since after fixing the destinations we obtain an instance of Token Swapping, which is polynomially solvable on paths, we observe that each feasible solution $\boldsymbol{s}$ for $\mathcal{I}$ corresponds to a perfect matching in G (and vice versa).

Recall that the number of swaps required to solve an instance of Token Swapping on a path is equal to the number of inversions in the initial permutation of tokens. Suppose there is such an inversion in σ, i.e. $\sigma(t_i) > \sigma(t_j)$ for

some $i < j$. Observe that this is exactly equivalent to saying that the edges $t_i\sigma(t_i)$ and $t_j\sigma(t_j)$ of G cross (see Figure 8).

So let us formally define the problem Minimum Crossing Bipartite Matching, which is equivalent to Subset Token Swapping on a path. The instance of Minimum Crossing Bipartite Matching is (G, k), where k is an integer and G is a bipartite graph with n vertices in each bipartition class. Moreover, the vertices of G are positioned on two parallel lines, one for each bipartition class. We can also assume that G has at least one perfect matching. The problem asks if G has a perfect matching with at most k pairwise crossing pairs of edges.

The problem Minimum Crossing Bipartite Matching (and thus also Subset Token Swapping on paths) was recently shown to be NP-hard by Guśpiel [18].

Theorem 13 (Guśpiel [18]) Subset Token Swapping *remains NP-hard for paths, even if each token has at most 2 possible destinations, and each vertex is a destination of at most 2 tokens. Moreover, the problem cannot be solved in time $2^{o(n)}$ (where n is the number of vertices of the path), unless the ETH fails.*

This result allows us to generalize Theorem 11 to all hereditary classes of graphs.

Theorem 14 *Let $\mathcal{G}$ be a hereditary class containing an infinite number of connected graphs.* Subset Token Swapping *is* NP*-complete, when restricted to graphs from $\mathcal{G}$. Moreover, if there exists an algorithm solving* Subset Token Swapping *in time $2^{o(n)}$ for every graph in $G \in \mathcal{G}$ with n vertices, then the ETH fails.*

Proof. If $\mathcal{G}$ has unbounded degree, then the claim holds by Theorem 11. On the other hand, if there is a constant d, such that $\Delta(G) \leq d$ for all $G \in \mathcal{G}$, then $\mathcal{G}$ contains all paths. Indeed, let n be an integer and let $G \in \mathcal{G}$ be a graph with at least $n \cdot d^n$ vertices (it always exists, since $\mathcal{G}$ is infinite). Run a BFS algorithm on G, starting from an arbitrary vertex, and consider the obtained BFS-layers. The number of such layers is at least n, so G contains P_n as an induced subgraph. Since $\mathcal{G}$ is hereditary, we have $P_n \in \mathcal{G}$. The claim follows by Theorem 13. □

8 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 Token Swapping remains NP-complete even if the input graph is a tree.

Another interesting problem is the following. By Miltzow *et al.* [28, Theorem 1] (see also Proposition 4), Token Swapping 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 [17]. It would be interesting to obtain elementary FPT algorithms for planar graphs and graph with bounded treewidth (or even trees), just as we did for graphs with bounded degree.

References

1. E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways, for Your Mathematical Plays: Games in particular*, volume 2. Academic Pr, 1982.
2. H. L. Bodlaender and J. Nederlof. Subexponential time algorithms for finding small tree and path decompositions. In *ESA 2015 Proc.*, pages 179–190. Springer, 2015.
3. É. Bonnet, T. Miltzow, and P. Rzazewski. Complexity of Token Swapping and its Variants. In H. Vollmer and B. Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*, volume 66 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
4. P. Bose and F. Hurtado. Flips in planar graphs. *Computational Geometry*, 42(1):60–80, 2009.
5. G. Calinescu, A. Dumitrescu, and J. Pach. Reconfigurations in graphs and grids. In *LATIN 2006 Proc.*, pages 262–273. Springer, 2006.
6. A. Cayley. LXXVII. Note on the theory of permutations. *Philosophical Magazine Series 3*, 34(232):527–529, 1849.
7. C. J. Colbourn. The complexity of completing partial latin squares. *Discrete Applied Mathematics*, 8(1):25 – 30, 1984.
8. M. De Berg, M. Van Kreveld, M. Overmars, and O. C. Schwarzkopf. Computational geometry. In *Computational geometry*, pages 1–17. Springer, 2000.
9. E. D. Demaine, M. L. Demaine, E. Fox-Epstein, D. A. Hoang, T. Ito, H. Ono, Y. Otachi, R. Uehara, and T. Yamada. Linear-time algorithm for sliding tokens on trees. *Theoretical Computer Science*, 600:132–142, 2015.
10. P. Erdős and G. Szekeres. *Classic Papers in Combinatorics*, chapter A Combinatorial Problem in Geometry, pages 49–56. Birkhäuser Boston, Boston, MA, 1987.
11. 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.
12. 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.

13. F. Farnoud and O. Milenkovic. Sorting of permutations by cost-constrained transpositions. *IEEE Trans. Information Theory*, 58(1):3–23, 2012.
14. E. Fox-Epstein, D. A. Hoang, Y. Otachi, and R. Uehara. Sliding token on bipartite permutation graphs. In K. Elbassioni and K. Makino, editors, *Algorithms and Computation*, volume 9472 of *Lecture Notes in Computer Science*, pages 237–247. Springer Berlin Heidelberg, 2015.
15. T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.
16. D. Graf. How to sort by walking on a tree. In *ESA 2015 Proc.*, pages 643–655. Springer, 2015.
17. M. Grohe, S. Kreutzer, and S. Siebertz. Deciding first-order properties of nowhere dense graphs. In *STOC 2014 Proc.*, pages 89–98. ACM, 2014.
18. G. Guśpiel. Complexity of finding perfect bipartite matchings minimizing the number of intersecting edges. *CoRR*, abs/1709.06805, 2017.
19. L. S. Heath and J. P. C. Vergara. Sorting by short swaps. *Journal of Computational Biology*, 10(5):775–789, 2003.
20. I. Holyer. The NP-Completeness of Some Edge-Partition Problems. *SIAM J. Comput.*, 10(4):713–717, 1981.
21. R. Impagliazzo and R. Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367 – 375, 2001.
22. R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.
23. T. Kasai, A. Adachi, and S. Iwata. Classes of pebble games and complete problems. *SIAM Journal on Computing*, 8(4):574–586, 1979.
24. D. E. Knuth. *The Art of Computer Programming*, volume 3 / Sorting and Searching. Addison-Wesley, 1982. ISBN 0-201-03803-X.
25. D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.
26. D. Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010.
27. D. Marx and M. Pilipczuk. Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. *CoRR*, abs/1504.05476, 2015.
28. T. Miltzow, L. Narins, Y. Okamoto, G. Rote, A. Thomas, and T. Uno. Approximation and Hardness of Token Swapping. In P. Sankowski and C. Zaroliagis, editors, *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 66:1–66:15, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
29. J. Nešetřil and P. Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012.
30. 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.

31. T. D. Parsons. Pursuit-evasion in a graph. In *Theory and applications of graphs*, pages 426–441. Springer, 1978.
32. 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.
33. F. P. Ramsey. On a problem in formal logic. *Proc. London Math. Soc. (3)*, 30:264–286, 1930.
34. W. J. Savitch. Relationships Between Nondeterministic and Deterministic Tape Complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.
35. R. M. Wilson. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory, Series B*, 16(1):86 – 96, 1974.
36. 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 2014 Proc.*, pages 364–375. Springer, 2014.
37. K. Yamanaka, T. Horiyama, D. G. Kirkpatrick, Y. Otachi, T. Saitoh, R. Uehara, and Y. Uno. Swapping colored tokens on graphs. In *WADS 2015 Proc.*, pages 619–628, 2015.
38. G. Yasui, K. Abe, K. Yamanaka, and T. Hirayama. Swapping labeled tokens on complete split graphs. *SIG Technical Reports*, 2015-AL-153(14):1–4, 2015.