Improved Hardness-Of-Approximation for Token-Swapping

Sam Hiken
Department of Computer Science and Engineering, University of Michigan, Ann Arbor, MI, USA

Nicole Wein
Department of Computer Science and Engineering, University of Michigan, Ann Arbor, MI, USA

Abstract

We study the token swapping problem, in which we are given a graph with an initial assignment of one distinct token to each vertex, and a final desired assignment (again with one token per vertex). The goal is to find the minimum length sequence of *swaps* of adjacent tokens required to get from the initial to the final assignment.

The token swapping problem is known to be NP-complete. It is also known to have a polynomial-time 4-approximation algorithm. From the hardness-of-approximation side, it is known to be NP-hard to approximate with a ratio better than 1001/1000.

Our main result is an improvement of the approximation ratio of the lower bound: We show that it is NP-hard to approximate with ratio better than 14/13.

We then turn our attention to the *0/1-weighted* version, in which every token has a weight of either 0 or 1, and the cost of a swap is the sum of the weights of the two participating tokens. Unlike standard token swapping, no constant-factor approximation is known for this version, and we provide an explanation. We prove that 0/1-weighted token swapping is NP-hard to approximate with ratio better than $(1-\varepsilon)\ln(n)$ for any constant $\epsilon > 0$.

Lastly, we prove two barrier results for the standard (unweighted) token swapping problem. We show that one cannot beat the current best known approximation ratio of 4 using a large class of algorithms which includes all known algorithms, nor can one beat it using a common analysis framework.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Graph algorithms analysis

Keywords and phrases algorithms, token-swapping, hardness-of-approximation, lower-bounds

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.57

Related Version *Full Version*: `https://arxiv.org/abs/2410.19638`

Funding *Sam Hiken*: NSF grant CNS-2150186, Paglia Post-Baccalaureate Fellowship.

Acknowledgements We would like to thank anonymous reviewers for their helpful suggestions.

1 Introduction

In the *token swapping* problem [1, 8, 2, 25, 30, 31, 32, 35, 18, 9, 10, 16, 20, 17, 36, 21, 29], we are given an undirected n-vertex graph $G = (V, E)$, n distinct tokens, and two one-to-one assignments of tokens to vertices: a *starting* assignment, and a *target* assignment. A swap along an edge $(u, v) \in E$ switches the locations of the tokens on u and v. The token swapping problem asks how many swaps are needed to arrive at the target configuration from the starting configuration.

The token swapping problem is a fundamental and well-studied problem, and one of the central problems in the area of *reconfiguration* algorithms. It has also found relevance in a number of disparate areas including network engineering [2], robot motion planning [12, 28], and game theory [14]. Token swapping (mainly its parallel variant [3, 6, 12, 17, 37]) also

has an extensively studied application to qubit routing (e.g. [5, 26, 19, 7, 11, 15, 24, 33]). Algorithms and heuristics for the token swapping problem have also undergone experimental evaluation [23, 27].

The token swapping problem is NP-complete [18], even when the underlying graph is a tree [1]. As a result, the research literature has focused on approximation algorithms. Miltzow, Narins, Okamoto, Rote, Thomas, and Uno gave a 4-approximation [18], which remains the best known. For the special case of trees, there is a 2-approximation, which was independently discovered 3 times using different algorithms [2, 32, 35].

From the hardness side, the above work [18] proved that token swapping is APX-hard (on general graphs). Thus, unless P = NP, token swapping does not admit a PTAS, and instead there exists some positive constant c for which there is no c-approximation. Regarding this constant c, they state *"We want to point out that a crude estimate for the constant c in [the NP-hardness result] is $c \approx 1 + 1/1000$. We do not believe that it is worth to compute c exactly. Instead, we hope that future research might find reductions with better constants."* This question is the main focus of our work.

We also consider the *0/1-weighted* version of token swapping, in which each token has a weight of either 0 or 1, and the cost of a swap is the sum of the weights of the two tokens. Despite being studied in prior work [8, 1], there is no known approximation algorithm for 0/1-weighted token swapping with a non-trivial approximation ratio. Furthermore, there is no known separation between 0/1-weighted token swapping and standard token swapping.

See the introductions of [8] and [1] for a more detailed survey of related work, including a sizable body of work on token swapping for a number of special classes of graphs.

1.1 Our Results

Our main result is the first hardness of approximation result for token swapping with an explicit approximation ratio of "reasonable" magnitude. Specifically, we show that it is NP-hard to obtain better than a 14/13-approximation:

▶ **Theorem 1.** *For any constant $\epsilon > 0$, it is NP-hard to approximate token swapping on graphs within a factor of $14/13 - \epsilon$.*

Our result is via a reduction from the label cover problem. We note that our result does not rely on the Unique Games Conjecture, and is instead a gap-preserving reduction from a regime of the label cover problem known to be NP-complete.

For 0/1-weighted token swapping, our next result explains the aforementioned gap in the literature by showing a large separation between the 0/1-weighted and unweighted versions. Specifically, we show that it is NP-hard to obtain better than an $(\ln n)$-approximation for 0/1-weighted token swapping:

▶ **Theorem 2.** *For any constant $\epsilon > 0$, it is NP-hard to approximate weighted token swapping with $\{0, 1\}$ weights on n vertices within a factor of $(1 - \epsilon) \cdot \ln n$.*

Our reduction uses a completely different technique from our main result, and is a much simpler reduction, from the set cover problem. This is notable in comparison to the known reductions in the token swapping literature, which tend to be quite complicated [1, 8, 18, 9].

Our final two results are barrier results regarding the algorithm and analysis techniques that could possibly achieve better than the current best-known approximation ratio of 4 (for standard unweighted token swapping). All known token swapping algorithms have the

natural property of *local optimality*: they never perform a swap that brings *both* tokens farther from their destinations.[1] We show that, strangely, this property must be violated to achieve any approximation ratio better than 4.

▶ **Theorem 3.** *For any $\delta > 0$, there exists a token swapping instance K so that for any locally optimal swap sequence of length k, $(4-\delta) \cdot OPT(K) \leq k$.*

In our second barrier result, we show that a proof technique used by all known analyses of approximation algorithms for token swapping, cannot yield better than a 4-approximation. In particular, all known approximation algorithms for token swapping on general graphs and trees [18, 35, 2] use the following approach to bound the approximation factor. Given a Token Swapping instance K, consider the quantity $total(K) = \sum_{t \in T} \text{dist}(f_1(t), f_2(t))$. We observe that $\frac{1}{2} total(K) \leq OPT(K)$, as each swap moves at most two tokens closer to their destinations. Proofs of correctness for approximation algorithms then show that they yield swap sequences of length at most $c \cdot total(K)$, which implies that the given algorithm is a $2c$-approximation. A natural approach to obtaining a $(4-\epsilon)$-approximation for token swapping on graphs, then, is to show that on any input an algorithm yields a swap sequence of length at most $c \cdot total(K)$, for some $c < 2$. Below, we show that this approach will not work.

▶ **Theorem 4.** *For any $\delta > 0$, there exists a token swapping instance K so that $OPT(K) \geq (2-\delta) \cdot total(K)$.*

The proof is given in the full version.

Lastly, in the full version we provide an alternative algorithm that gives a 4-approximation for token swapping (in addition to the known algorithm [18]). While the algorithm of [18] is an extension of the 2-approximation "happy swap algorithm" for trees [2], our algorithm is an extension of the 2-approximation "cycle algorithm" for trees [35].

1.2 Additional Related Work

Token swapping and its variants have been studied from many angles. Exponential-time algorithms and hardness under the Exponential Time Hypothesis (ETH) were studied in [18]. Parameterized complexity was studied in [9], where the authors show that token swapping is W[1]-hard when the parameter is the number of swaps, and show further hardness under ETH. Token swapping has also been studied on a variety of special classes of graphs including cycles [16], stars [21, 20], brooms [17, 8, 31], complete bipartite graphs [35], complete split graphs [36], and cliques [10] (dating back to Cayley in 1849). *Colored* token swapping has also been studied, where each token has a color and same-colored tokens are indistinguishable [18, 8, 9, 34, 34]. There are also several other models of token movement on graphs such as token sliding, token rotation, and token permutation (see e.g. [28]). See also the introductions of [8] and [1] for more details on the wealth of related work.

[1] We compare the notion of a locally optimal algorithm to that of an *ℓ-straying* algorithm, introduced in prior work on token swapping for *trees* [1]. These two notions are incomparable: a swap sequence can have either property without having the other. However, any algorithm that solves token swapping on general graphs must generate an $\Omega(n)$-straying swap sequence: consider the example of a cycle where each token wants to shift over by 1. For this reason, we do not consider ℓ-straying algorithms for general graphs, and focus instead on locally optimal algorithms.

2 Preliminaries

▶ **Definition 5.** *A* Token Swapping *instance* $K = (G, T, f_1, f_2)$ *consists of a graph* $G = (V, E)$*, a set of tokens* T *where* $|T| = |V|$*, and two one-to-one assignments* $f_1, f_2 : T \to V$*. We call* f_1 *and* f_2 *the* starting *and* target *configurations respectively. A swap along an edge* $(u, v) \in E$ *switches the locations of the tokens on* u *and* v*. The token swapping problem asks how many swaps are needed to arrive at the target configuration from the starting configuration.*

For a token swapping instance K, we denote by $OPT(K)$ the length of the shortest swap sequence. For a vertex v, we denote by $f_1^{-1}(v)$ and $f_2^{-1}(v)$ the tokens that begin and end on v. When a token t lies on the vertex v_1 of the path $p = v_1, v_2, ..., v_k$, we use the phrase *bubbling* t *across* p to denote the sequence of swaps $(v_1, v_2), (v_2, v_3), ..., (v_{k-1}, v_k)$.

We also consider the following weighted variant of Token Swapping:

▶ **Definition 6** (Weighted Token Swapping)**.** *An instance of* Weighted Token Swapping $W = (G, T, w, f_1, f_2)$ *consists of a graph* $G = (V, E)$*, together with a set of tokens* T *of size* $|V|$*. The weight function* $w : T \to \mathbb{R}^{\geq 0}$ *assigns to each token a non-negative real weight. As in standard* Token Swapping*, the one-to-one functions* $f_1, f_2 : T \to V$ *map each token to a unique vertex, giving the starting and target configurations of* W*. The weight of an edge swap is the sum of the weights of the tokens being swapped. The weight of a swap sequence is the sum of the weights of the constitutive edge swaps. The output to* W *is the weight of the lowest-weight swap sequence from the starting to the target configuration.*

3 Hardness for standard token swapping

In this section, we prove Theorem 1.

▶ **Theorem 1.** *For any constant* $\epsilon > 0$*, it is NP-hard to approximate token swapping on graphs within a factor of* $14/13 - \epsilon$*.*

This improves on the lower bound presented in [18], which is roughly 1001/1000. We obtain our lower bound via a gap-preserving reduction from Label-Cover, defined below.

▶ **Definition 7** (Label Cover)**.** *An instance of* Label-Cover $\Phi = (X, Y, E, \Sigma, \Pi)$ *consists of a bipartite graph with vertex set* $X \cup Y$ *and edge set* E*, together with a finite alphabet* Σ*, and a set of constraints* $\Pi = \{\Pi_e : e \in E\}$*. Each constraint is a function* $\Pi_e : \Sigma \to \Sigma$*.*

A labelling of $X \cup Y$ *is a function* $\lambda : X \cup Y \to \Sigma$ *which assigns a label to each vertex. For* $x \in X$*,* $y \in Y$*, a constraint* $\Pi_{(x,y)}$ *is satisfied if* $\Pi_{(x,y)}(\lambda(x)) = \lambda(y)$*.*

We denote by $OPT(\Phi)$ the maximum fraction of constraints in Φ which can be satisfied by any labelling. An instance of the promise problem GapLabel-Cover$_{1,\gamma}(\Sigma)$ consists of a label-cover instance Φ with alphabet Σ, together with the guarantee that either $OPT(\Phi) = 1$ or $OPT(\Phi) < \gamma$. Here, γ is a positive constant close to 0. The following is a seminal result in the theory of hardness of approximation.

▶ **Theorem 8.** *For any constant* $\gamma > 0$*, there exists a sufficiently large constant* $|\Sigma|$ *(dependent only on* γ*) such that* GapLabel-Cover$_{1,\gamma}(\Sigma)$ *is NP-hard.*

Arora and Lund [4] describe a reduction from 3-SAT which, together with Raz's Parallel Repetition Theorem [22], implies that Theorem 8 remains true even on instances where the underlying graph is regular. Moreover, we may take the degree to be at least any arbitrarily

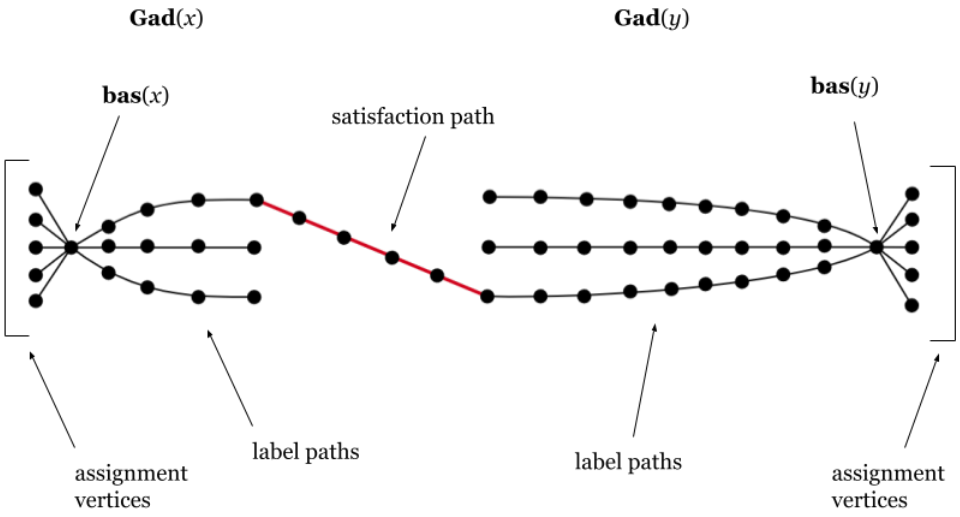

Figure 1 A left and right gadget, joined by a satisfaction path; $d = 5$, $|\Sigma| = 3$.

large constant by creating many copies of our graph $X \cup Y$ and adding a copy of the constraint $\Pi_{(x,y)}$ between every copy of x and every copy of y. Note that in such an instance $|X| = |Y|$. We use a gap-preserving reduction from label-cover to prove Theorem 1.

3.1 Construction

Our reduction maps a label cover instance $\Phi = (X, Y, E, \Sigma, \Pi)$, where the base graph has degree d, to a token swapping instance $K(\Phi) = (G, T, f_1, f_2)$.

We construct G by building a gadget $\mathbf{Gad}(v)$ for each $v \in X \cup Y$ as follows. Our construction for $\mathbf{Gad}(v)$ starts with a base vertex $\mathbf{bas}(v)$. We add d additional vertices adjacent to $\mathbf{bas}(v)$, which we call *assignment vertices* (recall that d is the degree in the label-cover instance). To each assignment vertex in $\mathbf{Gad}(v)$ we identify a unique edge $(v, w) \in E$ (where E is the label cover edge set) incident to v; we call this assignment vertex $\mathbf{asg}(v, w)$. We then create $|\Sigma|$ paths, each with $\mathbf{bas}(v)$ as an endpoint. We call these paths *label paths*. If $v \in X$, then each label path in $\mathbf{Gad}(v)$ contains $d - 1$ edges; if $v \in Y$, then each label path in $\mathbf{Gad}(v)$ contains $2d - 1$ edges. To each label path in $\mathbf{Gad}(v)$ we associate a unique $\sigma \in \Sigma$, and denote this label path by $\mathbf{lab}(v, \sigma)$.

If $v \in X$, then we call $\mathbf{Gad}(v)$ a *left gadget*; otherwise, we call $\mathbf{Gad}(v)$ a *right gadget*. We call an assignment vertex a left (resp. right) assignment vertex if it is in a left (resp. right) gadget. We denote the set of left gadgets by $\mathbf{L}$ and the set of right gadgets by $\mathbf{R}$.

Whenever it is the case that the label pair (σ_x, σ_y) satisfies the constraint $\Pi_{(x,y)}$, we add a path of d edges connecting the endpoint of $\mathbf{lab}(x, \sigma_x)$ opposite $\mathbf{bas}(x)$ with the endpoint of $\mathbf{lab}(y, \sigma_y)$ opposite $\mathbf{bas}(y)$. We call such a path a *satisfaction path*, and denote it by $\mathbf{sat}(x, \sigma_x, y, \sigma_y)$.

This gives our construction for G. An illustration of a left and right gadget connected by a satisfaction path given in Figure 1. We construct f_1 and f_2 as follows. For each $(x, y) \in E$, the tokens on $\mathbf{asg}(x, y)$ and $\mathbf{asg}(y, x)$ wish to swap positions with each other. That is, $f_2(f_1^{-1}(\mathbf{asg}(x, y))) = \mathbf{asg}(y, x)$, and $f_2(f_1^{-1}(\mathbf{asg}(y, x))) = \mathbf{asg}(x, y)$.

We will refer to such a token as an *assignment token*. We will call an assignment token t where $f_1(t)$ is a left (resp. right) assignment vertex a left (resp. right) assignment token. For $t \in T$ that are not assignment tokens, $f_1(t) = f_2(t)$. That is, t does not wish to move from its starting position.

The intuition behind the reduction is as follows: if there is a labelling λ satisfying every constraint in Π, then all the assignment tokens that start in $\mathbf{Gad}(v)$ can move to their destination along the single label path associated with $\lambda(v)$. Then, any assignment token seeking to *enter* $\mathbf{Gad}(v)$ can travel along this path, and will in the process swap with many of the assignment tokens leaving $\mathbf{Gad}(v)$, bringing them closer to their destination. If, on the other hand, no good labelling exists, then having so many efficient swaps becomes impossible.

We prove the following lemma.

▶ **Lemma 9.** *Let $\Phi = (X, Y, \Sigma, \Pi)$ be a label-cover instance whose graph is d-regular. The following two claims hold:*

- *If $OPT(\Phi) = 1$, then $OPT(K(\Phi)) \leq \left(\frac{13}{4}d^2 + d\right)|X \cup Y|$.*
- *If $OPT(\Phi) < \gamma$, then $OPT(K(\Phi)) > \left(\frac{7-\gamma}{2}d^2 - 5d\right)|X \cup Y|$.*

Given a label-cover instance Φ, the token swapping instance $K(\Phi)$ can be computed in polynomial time. We set d to be a large constant and γ a small constant, and the hardness result of Theorem 1 follows from Lemma 9. In Section 3.2, we briefly discuss the proof of the first half of Lemma 9 (completeness), with a proof given in the full version. In Section 3.3, we prove the second half (soundness).

3.2 Completeness

Let Φ be a label-cover instance as defined in Lemma 9 such that $OPT(\Phi) = 1$. We prove the first half of Lemma 9 by providing a swap sequence for $K(\Phi)$ using $(\frac{13}{4}d^2 + d)|X \cup Y|$ swaps. We describe the procedure in detail, and prove the first half of Lemma 9 in the full version.

3.3 Soundness

In this section, we will prove the second half of Lemma 9: if $OPT(\Phi) < \gamma$, then $OPT(K(\Phi)) > \left(\frac{7-\gamma}{2}d^2 - 5d\right)|X \cup Y|$.

Let Φ be a label-cover instance such that $OPT(\Phi) < \gamma$, and let S_{OPT} denote the optimal sequence of swaps on $K(\Phi)$. For a token $t \in T$, consider the subsequence of S_{OPT} in which t is one of the tokens being swapped. This subsequence traces out a walk in G. Consider the path from $f_1(t)$ to $f_2(t)$ obtained by removing all the closed sub-walks from this walk. We denote this path $\mathbf{Swap}(t)$.

We now partition the assignment tokens into two types: *detour tokens* and *non-detour* (assignment) tokens.

▶ **Definition 10** (Detour token)**.** *A token t is a* detour token *if:*

- *$f_1(t)$ is an assignment vertex.*
- ***Swap**(t) has more than $4d$ edges.*

We will refer to all other assignment tokens as *non-detour tokens*. This allows us to introduce the following notation: det_L (resp. det_R) is the number of detour tokens t such that $f_1(t)$ is a left (resp. right) assignment vertex. We will denote by Det the set of detour tokens, and $Ndet$ the set of non-detour tokens.

We obtain lower bounds on two quantities: B_{sat}, the number of swaps occurring within satisfaction paths, and B_{Gad}, the number of swaps occurring within gadgets.

$$B_{sat} \geq (d^2 - 3d)|X \cup Y| + \frac{d}{2}(det_L + det_R) \tag{1}$$

$$B_{Gad} > \left(\frac{5-\gamma}{2}d^2 - 2d\right)|X \cup Y| - \frac{d}{2}(det_L + det_R) \tag{2}$$

Combining these inequalities proves the second half of Lemma 9.

3.3.1 Swaps on satisfaction paths

First, we prove inequality 1.

▶ **Observation 11.** *If t is a detour token, then **Swap**(t) contains at least three satisfaction paths as subpaths.*

Observation 11 follows from the construction of G; any path of length greater than $4d$ with a left assignment vertex and a right assignment vertex as endpoints traverses at least three satisfaction paths. Therefore, we can associate to each detour token t a sequence of at least three satisfaction paths. We refer to the path(s) in this sequence which are not the first or last path as the *intermediate path(s)* of t. Each detour token has at least one intermediate path.

▷ Claim 12. S_{OPT} contains at least $\frac{d}{2}(det_L + det_R)$ swaps where at least one token is a detour token visiting one of its intermediate paths.

Proof. There are $det_L + det_R$ detour tokens, each of which participates in at least d swaps on intermediate paths. Each swap involves at most 2 tokens, hence the bound of $\frac{d}{2}(det_L + det_R)$. ◁

Furthermore, let $ext(t)$ denote the number of swaps that t participates in on satisfaction paths that are not intermediate paths. For $t \in Det$, $ext(t) \geq 2d$. For $t \in Ndet$, $ext(t) \geq d$. Unfortunately, we cannot simply add these quantities together to obtain a lower bound on the number of swaps on satisfaction paths. This is because a swap may be between two different assignment tokens, so adding these quantities would risk double-counting. However, we can obtain an upper-bound on the amount of double-counting that can take place, using the following definition.

▶ **Definition 13** (Efficient satisfaction swap)**.** *A swap between assignment tokens t_1 and t_2 is an* efficient satisfaction swap *if one of the following is the case:*

1. *neither t_1 nor t_2 is a detour token,*
2. *one of t_1 or t_2 is not a detour token, the other is a detour token not on an intermediate path, and the swap is contained in **Swap**(t), where t is the detour token, or*
3. *t_1 and t_2 are detour tokens, neither is on an intermediate path, and the swap is contained in **Swap**(t), where t is t_1 or t_2.*

Let k_{sat} denote the number of efficient satisfaction swaps in S_{OPT}. Combining Definition 13 with Claim 12, we obtain the following inequality:

$$B_{sat} \geq \frac{d}{2}(det_L + det_R) + \sum_{t \in Det} ext(t) + \sum_{t \in Ndet} ext(t) - k_{sat}$$

which implies that

$$B_{sat} \geq \frac{d}{2}\left(det_L + det_R\right) + 2d(det_L + det_R) + d\Big(d|X \cup Y| - (det_L + det_R)\Big) - k_{sat}. \quad (3)$$

We give an upper bound on k_{sat}. We do so by proving individual upper bounds on each of the three types of efficient satisfaction swaps outlined in Definition 13.

Swaps of type 1: *neither t_1 not t_2 is a detour token.* Because the label-cover graph is simple, each edge $(x, y) \in E$ is mapped by our reduction to a unique pair of assignment vertices, those being $\mathbf{asg}(x, y)$ and $\mathbf{asg}(y, x)$. Therefore, any satisfaction path $\mathbf{sat}(x, \sigma_x, y, \sigma_y)$ is traversed by at most two non-detour assignment tokens, those being $f_1^{-1}(\mathbf{asg}(x, y))$ and $f_1^{-1}(\mathbf{asg}(y, x))$. It follows that any non-detour token can swap on a satisfaction path with at most one other non-detour token Because there are $d|X \cup Y| - (det_L + det_R)$ non-detour assignment tokens, we obtain the following upper bound on swaps of type 1:

$$\frac{1}{2}\Big(d|X \cup Y| - (det_L + det_R)\Big).$$

Swaps of type 2: *one of t_1 or t_2 is not a detour token, and the other is a detour token not on an intermediate path.* Again, any satisfaction path $\mathbf{sat}(x, \sigma_x, y, \sigma_y)$ is traversed by at most two non-detour assignment tokens, those being $f_1^{-1}(\mathbf{asg}(x, y))$ and $f_1^{-1}(\mathbf{asg}(y, x))$. A detour token moving across $\mathbf{sat}(x, \sigma_x, y, \sigma_y)$ can swap with only one, depending on the direction in which it is moving. Because each detour token traverses exactly two non-intermediate satisfaction paths, the number of swaps of type 2 is at most:

$$2(det_L + det_R)$$

Swaps of type 3: *t_1 and t_2 are detour tokens, but neither is on an intermediate path.* Each detour token participates in at most a combined $2d$ swaps on the first and last satisfaction paths that it visits. Each of these swaps involves at most 2 such tokens, implying that the number of swaps of type 3 is at most:

$$d(det_L + det_R).$$

Combining these bounds with inequality 3, we are left with the following lower bound on B_{sat}:

$$\begin{aligned}
B_{sat} &\geq \frac{d}{2}\left(det_L + det_R\right) + d\Big(d|X \cup Y| - (det_L + det_R)\Big) + 2d(det_L + det_R) \\
&\quad - \frac{1}{2}\Big(d|X \cup Y| - (det_L + det_R)\Big) - 2(det_L + det_R) - d(det_L + det_R) \\
&= d^2|X \cup Y| + \frac{3d}{2}(det_L + det_R) - \frac{d}{2}|X \cup Y| - \left(d + \frac{5}{2}\right)(det_L + det_R) \\
&= \left(d^2 - \frac{d}{2}\right)|X \cup Y| + \left(\frac{d-5}{2}\right)(det_L + det_R) \\
&\geq \left(d^2 - 3d\right)|X \cup Y| + \frac{d}{2}(det_L + det_R)
\end{aligned}$$

as desired.

3.3.2 Swaps on gadgets

Next, we prove inequality 2:

$$B_{Gad} > \left(\frac{5-\gamma}{2}d^2 - 2d\right)|X \cup Y| - \frac{d}{2}(det_L + det_R).$$

Any assignment token t participates in at least $3d-2$ swaps within label paths: $d-1$ swaps on a left label path and $2d-1$ swaps on a right label path. Moreover, we can identify the two label paths, one in a left gadget and one in a right gadget, which come at either end of $\mathbf{Swap}(t)$. We refer to these paths as the *first* and *last legs* of t. Note that the first leg of t comes in the gadget containing t's start vertex, and the last leg comes in the gadget containing t's target vertex.

A swap which corresponds to an edge in the first leg of one token t_1 and the last leg of another token t_2 we will call an *efficient gadget swap*. Note that such a swap appears in both $\mathbf{Swap}(t_1)$ and $\mathbf{Swap}(t_2)$. We denote the number of efficient gadget swaps in S_{OPT} by k_{Gad}, and we denote the number of efficient gadget swaps occurring within the gadget $\mathbf{Gad}(v)$ by $k_{Gad}(v)$. We observe that

$$B_{Gad} \geq (3d^2 - 2d)|X \cup Y| - k_{Gad} \tag{4}$$

and we seek an upper bound on k_{Gad} accordingly.

To this end, we define the *outflow* of a label path p to be the number of tokens whose first leg is p, and the *inflow* of p to be the number of tokens whose last leg is p. For $\mathbf{Gad}(v) \in \mathbf{L} \cup \mathbf{R}$, we call the label path in $\mathbf{Gad}(v)$ with the maximum outflow (resp. inflow) the *out-dominant* (resp. *in-dominant*) label path of $\mathbf{Gad}(v)$. We denote by $out(\mathbf{Gad}(v))$ (resp. $in(\mathbf{Gad}(v))$) the outflow (resp. inflow) of the out-dominant (resp. in-dominant) label path in $\mathbf{Gad}(v)$. We obtain two inequalities.

▷ Claim 14. For any gadget $\mathbf{Gad}(v)$,
- $k_{Gad}(v) \leq d(out(\mathbf{Gad}(v)))$
- $k_{Gad}(v) \leq d(in(\mathbf{Gad}(v)))$.

Proof. We prove the first inequality. Let $t \in T$ be an assignment token. Suppose $f_2(t)$ is in $\mathbf{Gad}(v)$. Then t participates in at most $out(\mathbf{Gad}(v))$ efficient gadget swaps on $\mathbf{Gad}(v)$, as at most $out(\mathbf{Gad}(v))$ such swaps can occur on the label path taken by t. There are d tokens t where $f_2(t)$ is an assignment vertex in $\mathbf{Gad}(v)$, hence the bound of $d(out(\mathbf{Gad}(v)))$. The proof of the second inequality is symmetric. ◁

▷ Claim 15. If $OPT(\Phi) < \gamma$, then

$$\sum_{x \in X} out(\mathbf{Gad}(x)) + \sum_{y \in Y} in(\mathbf{Gad}(y)) < \left(\frac{1+\gamma}{2}\right) d|X \cup Y| + det_L$$

$$\sum_{x \in X} in(\mathbf{Gad}(x)) + \sum_{y \in Y} out(\mathbf{Gad}(y)) < \left(\frac{1+\gamma}{2}\right) d|X \cup Y| + det_R$$

Proof. We prove the first half of the claim; the proof of the second half is symmetric. Suppose for contradiction the first inequality does not hold. The left side of the inequality counts two quantities: (a) the total outflow from the out-dominant label path of each left gadget, plus (b) the total inflow to the in-dominant label path of each right gadget. Each left assignment token can contribute at most 1 to (a) and at most 1 to (b). Because $d|X \cup Y|$ is the total number of assignment tokens and the graph is regular (so $|X| = |Y|$), there are exactly $(d/2)|X \cup Y|$ left assignment tokens. As a result, even if the maximum number of left assignment tokens are contributing to only one of (a) or (b), at least $\frac{\gamma}{2}(d|X \cup Y|) + det_L$ contribute to both. Therefore, there are at least $\frac{\gamma}{2}(d|X \cup Y|)$ non-detour tokens starting in left gadgets which have an out-dominant label path as a first leg and an in-dominant label path as a last leg. Let t be such a token. Because t is a non-detour token, there is a

satisfaction path between t's first and last legs. By our construction, the constraint in the label cover instance associated with t is satisfied by the labelling corresponding to t's first and last legs. Consider a labelling of Φ which assigns to $x \in X$ the label corresponding to the out-dominant label path of $\mathbf{Gad}(x)$, and to $y \in Y$ the label corresponding to the in-dominant label path of $\mathbf{Gad}(y)$. There are at least $\gamma|E|$ constraints satisfied by this labelling, which is a contradiction. ◁

We average the inequalities in Claim 14 and rearrange some terms to obtain the new bound for any $\mathbf{Gad}(v)$:

$$\frac{k_{Gad}(v)}{d} \leq \frac{out(\mathbf{Gad}(v)) + in(\mathbf{Gad}(v))}{2}$$

and we average the two inequalities in Claim 15 to obtain

$$\sum_{v \in X \cup Y} \frac{out(\mathbf{Gad}(v)) + in(\mathbf{Gad}(v))}{2} < \left(\frac{1+\gamma}{2}\right) d|X \cup Y| + \frac{det_L + det_R}{2}.$$

Combining these bounds, we arrive at the following inequality:

$$\sum_{v \in X \cup Y} \frac{k_{Gad}(v)}{d} < \frac{1+\gamma}{2} d|X \cup Y| + \frac{det_L + det_R}{2}$$

which implies

$$k_{Gad} < \frac{1+\gamma}{2} d^2|X \cup Y| + \frac{d}{2}(det_L + det_R)$$

which, combined with inequality 4:

$$B_{Gad} \geq (3d^2 - 2d)|X \cup Y| - k_{Gad}$$

implies inequality 2:

$$B_{Gad} > \left(\frac{5-\gamma}{2} d^2 - 2d\right)|X \cup Y| - \frac{d}{2}(det_L + det_R).$$

4 Hardness for 0/1-Weighted Token Swapping

We prove the following theorem:

▶ **Theorem 2.** *For any constant $\epsilon > 0$, it is NP-hard to approximate weighted token swapping with $\{0, 1\}$ weights on n vertices within a factor of $(1 - \epsilon) \cdot \ln n$.*

We proceed via a reduction from the Set Cover problem, defined below.

▶ **Definition 16** (Set Cover). *An instance of Set-Cover $\Phi = (U, \{S_i\}_{1 \leq i \leq k})$ consists of a finite universe set U, together with subsets $S_1, ..., S_k \subseteq U$. We seek to find the size of the smallest collection of subsets in $\{S_1, ..., S_k\}$ so that each element of U is in at least one subset.*

Dinur and Steurer [13] prove the following theorem:

▶ **Theorem 17.** *For any constant $\delta > 0$, it is NP-hard to approximate Set Cover within a factor of $(1 - \delta) \cdot \ln(|U| + k)$.*

We give a gap-preserving reduction from Set Cover to Weighted Token Swapping. We describe the reduction in the full version, proving the desired lower-bound.

5 Additional barriers to improved algorithms

In this section, we discuss two barriers to obtaining a $(4-\epsilon)$-approximation for unweighted token swapping on graphs. The first barrier (Theorem 3 is against a broad class of algorithms that encompasses all known approximation algorithms for token swapping. The second barrier (Theorem 4) shows that the technique used to prove approximation factors for existing algorithms cannot be used to prove an approximation factor of $4-\epsilon$ for any $\epsilon > 0$. The proof of Theorem 4 is given in the full version of the paper.

5.1 A barrier against a general class of algorithms

We define the following property for swap sequences on a Token Swapping instance.

▶ **Definition 18** (Local Optimality). *A swap sequence is* locally optimal *if every swap between tokens t_1 and t_2 does not move both t_1 and t_2 further from their destinations.*

All known algorithms for token swapping on trees or on general graphs work by returning the length of a swap sequence that is locally optimal.

▶ **Theorem 3.** *For any $\delta > 0$, there exists a token swapping instance K so that for any locally optimal swap sequence of length k, $(4-\delta) \cdot OPT(K) \leq k$.*

We construct a token swapping instance K as follows. See Figure 2. Let p and q be parameters to be decided later. We can assume p is even. We begin with a cycle C^{out} of length $p \cdot q$. Starting with an arbitrary vertex, we label the vertices $v_0, ..., v_{p \cdot q-1}$ in one direction (say, clockwise) around the cycle. We partition the vertices of C^{out} into p consecutive "segments". For any $0 \leq i \leq p-1$, the vertices $\{v_{iq}, ..., v_{iq+(q-1)}\}$ form the i-th segment. If i is even, we call such a segment an *even segment*; else it is an *odd segment*. For any j so that v_j is in an even segment, the token starting on v_j has target vertex $v_{j+2q} \pmod{pq}$. If v_j is in an odd segment, the token starting on v_j has target vertex $v_{j-2q} \pmod{pq}$. That is, each token in an even segment wants to move to the corresponding vertex in the next even segment in the clockwise direction, while each token in an odd segment wants to move to the next odd segment over in the counter-clockwise direction. For each $0 \leq j \leq pq$, we add a path P_j of $2q-2$ edges between v_j and v_{j+2q}. All the vertices on this path (except the endpoints in C^{out}) wish to stay on their start vertices. We call these paths *inner paths*. We call the vertices in C^{out} *outer vertices*, and all other vertices *inner vertices*. Tokens that begin on outer vertices (which also have destinations on outer vertices) we call *outer tokens*; other tokens are *inner tokens*. Outer tokens with start and target vertices in even segments we will call *even tokens*; other outer tokens we will call *odd tokens*. Note that each outer token begins on a vertex connected to its target vertex by an inner cycle. The outer cycle, together with a single inner cycle, is depicted in Figure 2.

▷ Claim 19. $OPT(K) \leq pq^2$.

Proof. Here is a swap sequence bringing every token to its destination vertex:

1. For $j = 0, ..., pq-1$:
   a. If $f_1^{-1}(v_j)$ is an odd token, bubble it counter-clockwise around C^{out} across q edges.
2. Again, for $j = 0, ..., pq-1$:
   a. If $f_1^{-1}(v_j)$ is an odd token, bubble it counter-clockwise around C^{out} across q edges.

We check that the above swap sequence brings every token to its target vertex. No swap occurs on an edge in an inner cycle, so none of the inner tokens are displaced. Suppose $f^{-1}(v_j)$ (the token starting on v_j) is an odd outer token. It is not difficult to see that when

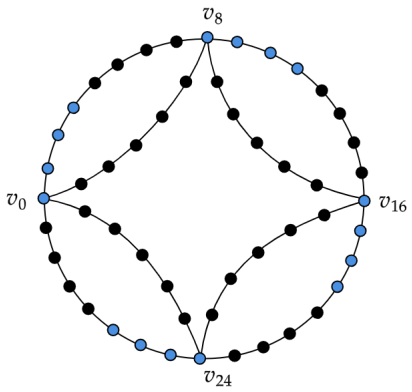

Figure 2 An (incomplete) graph for a token swapping instance where $p = 8$, $q = 4$. The diagram depicts the outer cycle and one inner cycle, but leaves out the remaining inner cycles for legibility. In the outer cycle, vertices in even segments are colored blue, while those in odd segments are colored black. The token starting on v_0 has target v_8, the token starting on v_8 has target v_{16}, the token starting on v_{16} has target v_{24}, and the token starting on v_{24} has target v_0.

$f^{-1}(v_j)$ is bubbled q edges counter-clockwise, it only swaps with even tokens (this is because any odd token from the same segment was already bubbled q edges counter-clockwise during a previous iteration). Thus, each odd token is bubbled counter-clockwise $2q$ times, and each even token is bubbled clockwise $2q$ times. This brings every token to its target node. ◁

▷ Claim 20. Let S be the shortest locally optimal swap sequence bringing every token to its destination, and let k be the length of S. Then $4pq^2 - 5pq - 8q^2 \leq k$.

We observe that for $0 \leq j \leq 2q - 1$, the inner paths $P_j, P_{j+2q}, P_{j+4q}, ..., P_{j+q(p-1)}$ form a cycle, which we will call the j-th *inner cycle*, denoted C^{in}_j. Moreover, every token begins on a vertex in the same inner cycle as its target vertex. Claim 20 will follow from the next two claims.

▷ Claim 21. Let $a, b \in V$ be in the same inner cycle C^{in}_j, and let $P_{a,b}$ be a path from a to b containing at least one edge not in C^{in}_j. Then $\text{dist}(a, b) + 2 \leq |P_{a,b}|$, where $|P_{a,b}|$ is the number of edges in $P_{a,b}$.

Proof. We will prove the claim in two steps. First, we will modify $P_{a,b}$ to obtain a path $P'_{a,b}$ which is the same length as $P_{a,b}$ and which does not contain any edges from inner cycles other than C^{in}_j, but which contains at least one edge from C^{out}. Then, we will show that any path from a to b which contains edges from (and only from) both C^{in}_j and C^{out} has length at least $\text{dist}(a, b) + 2$.

Suppose $P_{a,b}$ contains edges outside of C^{in}_j. If $P_{a,b}$ does not contain edges from a different inner cycle, then we set $P'_{a,b}$ to $P_{a,b}$ and move on to the argument in the next paragraph. Otherwise, $P_{a,b}$ contains an edge e from a different inner cycle. Let ℓ be such that C^{in}_ℓ is the inner cycle containing e. Suppose e is, in particular, the first edge to appear in $P_{a,b}$ which is in an inner cycle other than C^{in}_j. Therefore, e is incident to a vertex $v_{\ell'}$ so that $\ell' \equiv \ell \pmod{2q}$, and e is contained in either $P_{\ell'}$ or $P_{\ell'-2q}$. We can assume by symmetry that e is contained in $P_{\ell'}$. Because $P_{a,b}$ is a shortest path, it is simple. By the construction of the graph then, $P_{a,b}$ contains the entirety of $P_{\ell'}$. Let $v_{j'}$ be the last vertex in C^{in}_j preceding $v_{\ell'}$ in $P_{a,b}$. Because e is incident to an outer vertex, the path between $v_{j'}$ and $v_{\ell'}$ only has edges in C^{out}. The length of the sub-path of $P_{a,b}$ from the appearance of $v_{j'}$ to $v_{\ell'+2q}$ is

then $|\ell' - j'| + 2q - 2$. We replace this sub-path with the following path: the shortest path in C^{in}_j from $v_{j'}$ to $v_{j'+2q}$, then the shortest path in C^{out} from $v_{j'+2q}$ to $v_{\ell'+2q}$. The length of this path is at most $|\ell' - j'| + 2q - 2$, so we have not increased the length of $P_{a,b}$, but we have decreased the number of edges from an inner cycle other than C^{in}_j. We repeat this procedure until we obtain a path $P'_{a,b}$ which does not contain any edges from inner cycles other than C^{in}_j.

We have a simple path $P'_{a,b}$ which contains edges only from C^{in}_j and C^{out}, and at least one edge from C^{out}. Then there exists a consecutive sub-path in $P'_{a,b}$ of edges from C^{out} of length $2q$, as this is the number of edges needed to go from one vertex in C^{in}_j to another while traversing C^{out}. However, this sub-path can be replaced by the sub-path of length $2q - 2$ between its endpoints using only edges from C^{in}_j. Therefore, $\text{dist}(a, b) + 2 \leq |P'_{a,b}| = |P_{a,b}|$. ◁

For a, b in C^{in}_j, and for c not in C^{in}_j which is a neighbor of a, it is the case that $\text{dist}(a, b) < \text{dist}(c, b)$. Otherwise, there would be a path from a to b containing the edge (a, c) of length at most $\text{dist}(a, b) + 1$, even though (a, c) is not in C^{in}_j, which contradicts Claim 21. Because each token starts in the same inner cycle as its target vertex, this implies that any swap across an edge in C^{out} is not locally optimal. We proceed by proving a lower bound on the number of swaps needed to bring all the tokens in C^{in}_j to their target vertex without leaving C^{in}_j.

▷ Claim 22. For any j, the number of swaps needed to bring all of the tokens on C^{in}_j to their target vertices while swapping only along edges in C^{in}_j is greater than $2pq - 5p/2 - 4q$.

Proof. Suppose without loss of generality that v_j is in an even segment; by our construction, all the other outer vertices in C^{in}_j are also in even segments. Moreover, each outer vertex in C^{in}_j begins with a token which wants to move to the closest outer vertex in C^{in}_j in the clockwise direction, of distance $2q - 2$ away. All inner tokens want to stay on their start vertices. Note that C^{in}_j has length $p(q - 1)$, and contains $p/2$ outer vertices.

In a given swap, one token in C^{in}_j is moved clockwise, and the other counter-clockwise. For a token t, let $\mathit{clock}(t)$ be the number of swaps in the optimal swap sequence on C^{in}_j in which t is moved clockwise, and $\mathit{counter}(t)$ the number of times its moved counter-clockwise.

Let t_{in} be an inner token in C^{in}_j. Because t_{in} has the same start and target, $\mathit{clock}(t_{in}) - \mathit{counter}(t_{in}) \equiv 0 \pmod{p(q-1)}$. If t_{out} is an outer vertex in C^{in}_j, then $\mathit{clock}(t_{out}) - \mathit{counter}(t_{out}) \equiv 2q - 2 \pmod{p(q-1)}$. Because each swap moves one token clockwise and one token counter-clockwise, $\sum_{t_{in} \in C^{in}_j} \mathit{clock}(t_{in}) - \mathit{counter}(t_{in}) = 0$. It follows there is a token $t_{counter} \in C^j_{in}$ where $\mathit{counter}(t_{counter}) > \mathit{clock}(t_{counter})$ (else, each token in C^j_{in} would end on its start vertex). By our construction, the distance from $t_{counter}$'s start vertex to its target in the counter-clockwise direction is at least $p(q-1) - 2q + 2$, so $\mathit{counter}(t_{counter}) - \mathit{clock}(t_{counter}) \geq p(q-1) - 2q + 2$.

If there is an additional token $t'_{counter}$ so that $p(q-1) - 2q + 2 \leq \mathit{counter}(t')$, then there are at least $2(p(q-1) - 2q) > 2pq - 5p/2 - 4q$ swaps in the optimal swap sequence, as desired. Otherwise, there are at least $p/2 - 1$ outer tokens in C^{in}_j which are not moved counter-clockwise at least $p(q-1) - 2q$ times. Each is involved in at least $2q - 2$ swaps where it is the token being moved clockwise. Moreover, at most 1 of these swaps is with $t_{counter}$, because if a pair of tokens swap at least twice, then both swaps can be removed to yield an equivalent swap sequence. This results in a total of at least

$$\begin{aligned}(p/2-1)(2q-3) + \mathit{counter}(t_{counter}) &\geq (p/2-1)(2q-3) + p(q-1) - 2q \\ &> 2pq - 5p/2 - 4q\end{aligned}$$

swaps, as desired. ◁

Because there are $2q$ inner cycles, the total number of swaps taken by a sequence with only locally optimal swaps is at least $2q(2pq - 5p/2 - 4q) = 4pq^2 - 5pq - 8q^2$, proving Claim 20. We set p, q to be large enough so that $4pq^2$ overtakes the smaller-order terms, so that for the parameter δ,

$$4 - \delta < (4pq^2 - 5pq - 8q^2)/pq^2 < 4.$$

This proves Theorem 3.

References

1 Oswin Aichholzer, Erik D. Demaine, Matias Korman, Anna Lubiw, Jayson Lynch, Zuzana Masárová, Mikhail Rudoy, Virginia Vassilevska Williams, and Nicole Wein. Hardness of token swapping on trees. In *30th annual European Symposium on Algorithms*, volume 244 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 3, 15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.3.

2 S.B. Akers and B. Krishnamurthy. A group-theoretic model for symmetric interconnection networks. *IEEE Transactions on Computers*, 38(4):555–566, 1989. doi:10.1109/12.21148.

3 Noga Alon, F. R. K. Chung, and R. L. Graham. Routing permutations on graphs via matchings. *SIAM J. Discrete Math.*, 7(3):513–530, 1994. doi:10.1137/S0895480192236628.

4 Sanjeev Arora and Carsten Lund. Hardness of approximation. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, chapter 10, pages 399–446. PWS Publishing, Boston, 2004.

5 Avah Banerjee, Xin Liang, and Rod Tohid. Locality-aware qubit routing for the grid architecture. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 607–613. IEEE, 2022.

6 Indranil Banerjee and Dana Richards. New results on routing via matchings on graphs. In *Fundamentals of computation theory*, volume 10472 of *Lecture Notes in Comput. Sci.*, pages 69–81. Springer, Berlin, 2017. doi:10.1007/978-3-662-55751-8_7.

7 Aniruddha Bapat, Andrew M Childs, Alexey V Gorshkov, and Eddie Schoute. Advantages and limitations of quantum routing. *PRX Quantum*, 4(1):010313, 2023.

8 Ahmad Biniaz, Kshitij Jain, Anna Lubiw, Zuzana Masárová, Tillmann Miltzow, Debajyoti Mondal, Anurag Murty Naredla, Josef Tkadlec, and Alexi Turcotte. Token swapping on trees. *Discrete Math. Theor. Comput. Sci.*, 24(2):Paper No. 9, 37, 2022. doi:10.46298/DMTCS.8383.

9 Édouard Bonnet, Tillmann Miltzow, and Paweł Rzążewski. Complexity of token swapping and its variants. *Algorithmica*, 80(9):2656–2682, 2018. doi:10.1007/s00453-017-0387-0.

10 Arthur Cayley. Lxxvii. note on the theory of permutations. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 34(232):527–529, 1849.

11 Andrew M. Childs, Eddie Schoute, and Cem M. Unsal. Circuit transformations for quantum architectures. In *14th Conference on the Theory of Quantum Computation, Communication and Cryptography*, volume 135 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 3, 24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

12 Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Henk Meijer, and Christian Scheffer. Coordinated motion planning: reconfiguring a swarm of labeled robots with bounded stretch. *SIAM J. Comput.*, 48(6):1727–1762, 2019. doi:10.1137/18M1194341.

13 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 624–633, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2591796.2591884.

14 Laurent Gourvès, Julien Lesca, and Anaëlle Wilczynski. Object allocation via swaps along a social network. In *26th International Joint Conference on Artificial Intelligence (IJCAI'17)*, pages 213–219, 2017. doi:10.24963/IJCAI.2017/31.

15 Takehiro Ito, Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, and Yoshio Okamoto. Algorithmic theory of qubit routing. In *Algorithms and data structures*, volume 14079 of *Lecture Notes in Comput. Sci.*, pages 533–546. Springer, Cham, 2023. doi:10.1007/978-3-031-38906-1_35.

16 Mark R. Jerrum. The complexity of finding minimum-length generator sequences. *Theoret. Comput. Sci.*, 36(2-3):265–289, 1985. doi:10.1016/0304-3975(85)90047-7.

17 Jun Kawahara, Toshiki Saitoh, and Ryo Yoshinaka. The time complexity of permutation routing via matching, token swapping and a variant. *J. Graph Algorithms Appl.*, 23(1):29–70, 2019. doi:10.7155/jgaa.00483.

18 Tillmann Miltzow, Lothar Narins, Yoshio Okamoto, Günter Rote, Antonis Thomas, and Takeaki Uno. Approximation and hardness of token swapping. In *24th Annual European Symposium on Algorithms (ESA)*, volume 57 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 66, 15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.

19 Abtin Molavi, Amanda Xu, Martin Diges, Lauren Pick, Swamit Tannu, and Aws Albarghouthi. Qubit mapping and routing via maxsat. In *2022 55th IEEE/ACM international symposium on Microarchitecture (MICRO)*, pages 1078–1091. IEEE, 2022. doi:10.1109/MICRO56248.2022.00077.

20 Igor Pak. Reduced decompositions of permutations in terms of star transpositions, generalized Catalan numbers and k-ary trees. *Discrete Math.*, 204(1-3):329–335, 1999. doi:10.1016/S0012-365X(98)00377-X.

21 Frederick J Portier and Theresa P Vaughan. Whitney numbers of the second kind for the star poset. *European Journal of Combinatorics*, 11(3):277–288, 1990. doi:10.1016/S0195-6698(13)80127-8.

22 Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998. doi:10.1137/S0097539795280895.

23 Bruno Schmitt, Mathias Soeken, and Giovanni De Micheli. Symbolic algorithms for token swapping. In *2020 IEEE 50th International Symposium on Multiple-Valued Logic—ISMVL 2020*, pages 28–33. IEEE Computer Soc., Los Alamitos, CA, 2020. doi:10.1109/ISMVL49045.2020.00-34.

24 Asim Sharma and Avah Banerjee. Noise-aware token swapping for qubit routing. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 1, pages 82–88. IEEE, 2023. doi:10.1109/QCE57702.2023.00018.

25 Zhizhang Shen and Ke Qiu. On the Whitney numbers of the second kind for the star poset: comment on [European J. Combin. **11** (1990), no. 3, 277–288; mr1059558] by F. J. Portier and T. P. Vaughan. *European J. Combin.*, 29(7):1585–1586, 2008. doi:10.1016/j.ejc.2007.11.026.

26 Marcos Yukio Siraichi, Vinícius Fernandes dos Santos, Caroline Collange, and Fernando Magno Quintão Pereira. Qubit allocation as a combination of subgraph isomorphism and token swapping. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA):1–29, 2019. doi:10.1145/3360546.

27 Pavel Surynek. Finding optimal solutions to token swapping by conflict-based search and reduction to sat. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 592–599. IEEE, 2018. doi:10.1109/ICTAI.2018.00096.

28 Pavel Surynek. Multi-agent path finding with generalized conflicts: An experimental study. In *International Conference on Agents and Artificial Intelligence*, pages 118–142. Springer, 2019. doi:10.1007/978-3-030-37494-5_7.

29 Caio Henrique Segawa Tonetti, Vinicius Fernandes dos Santos, and Sebastián Urrutia. Polynomial time algorithms for the token swapping problem on cographs. *RAIRO Oper. Res.*, 58(1):441–455, 2024. doi:10.1051/ro/2023134.

30 Theresa P. Vaughan. Bounds for the rank of a permutation on a tree. *J. Combin. Math. Combin. Comput.*, 10:65–81, 1991.

31 Theresa P. Vaughan. Factoring a permutation on a broom. *J. Combin. Math. Combin. Comput.*, 30:129–148, 1999.

32 Theresa P. Vaughan and Frederick J. Portier. An algorithm for the factorization of permutations on a tree. *J. Combin. Math. Combin. Comput.*, 18:11–31, 1995.

33 Friedrich Wagner, Andreas Bärmann, Frauke Liers, and Markus Weissenbäck. Improving quantum computation by optimized qubit routing. *J. Optim. Theory Appl.*, 197(3):1161–1194, 2023. doi:10.1007/s10957-023-02229-w.

34 Katsuhisa Yamanaka, Erik D. Demaine, Takashi Horiyama, Akitoshi Kawamura, Shin-ichi Nakano, Yoshio Okamoto, Toshiki Saitoh, Akira Suzuki, Ryuhei Uehara, and Takeaki Uno. Sequentially swapping colored tokens on graphs. *J. Graph Algorithms Appl.*, 23(1):3–27, 2019. doi:10.7155/jgaa.00482.

35 Katsuhisa Yamanaka, Erik D. Demaine, Takehiro Ito, Jun Kawahara, Masashi Kiyomi, Yoshio Okamoto, Toshiki Saitoh, Akira Suzuki, Kei Uchizawa, and Takeaki Uno. Swapping labeled tokens on graphs. *Theoret. Comput. Sci.*, 586:81–94, 2015. doi:10.1016/J.TCS.2015.01.052.

36 Gaku Yasui, Kouta Abe, Katsuhisa Yamanaka, and Takashi Hirayama. Swapping labeled tokens on complete split graphs. *Inf. Process. Soc. Japan. SIG Tech. Rep*, 14:1–4, 2015.

37 Louxin Zhang. Optimal bounds for matching routing on trees. *SIAM J. Discrete Math.*, 12(1):64–77, 1999. doi:10.1137/S0895480197323159.