
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 23, no. 1, pp. 3–27 (2019)
DOI: 10.7155/jgaa.00482

Sequentially Swapping Colored Tokens on Graphs

Katsuhisa Yamanaka[1] *Erik D. Demaine*[2] *Takashi Horiyama*[3] *Akitoshi Kawamura*[4] *Shin-ichi Nakano*[5] *Yoshio Okamoto*[6] *Toshiki Saitoh*[7] *Akira Suzuki*[8] *Ryuhei Uehara*[9] *Takeaki Uno*[10]

[1]Iwate University, Japan
[2] Massachusetts Institute of Technology, USA
[3]Saitama University, Japan.
[4]Kyushu University, Japan.
[5] Gunma University, Japan.
[6] The University of Electro-Communications, and RIKEN Center for Advanced Intelligence Project, Japan.
[7]Kyushu Institute of Technology, Japan.
[8] Tohoku University, Japan.
[9] Japan Advanced Institute of Science and Technology, Japan.
[10] National Institute of Informatics, Japan.

Abstract

We consider a puzzle consisting of colored tokens on an n-vertex graph, where each token has a distinct starting vertex and a set of allowable target vertices for it to reach, and the only allowed transformation is to "sequentially" move the chosen token along a path of the graph by swapping it with other tokens on the path.

Submitted:	Reviewed:	Revised:	Reviewed:
September 2017	May 2018	June 2018	October 2018
Revised:	Accepted:	Final:	Published:
October 2018	October 2018	November 2018	January 2019
Article type:		Communicated by:	
Regular Paper		M.S. Rahman, H.-C. Yen and S.-H. Poon	

A preliminary version of this paper was presented to the The 11th International Conference and Workshops on Algorithms and Computation (WALCOM 2017), Hsinchu, Taiwan, 2017.
This work is partially supported by MEXT/JSPS KAKENHI Grant Numbers JP15H03389, JP15H05711, JP15K00008, JP15K00009, JP15KT0020, JP16H01743, JP16K00002, JP16K12539, JP16K16006, JP17H06287, JP17K00003, JP17K12636, JP17K19960, and JP18H04091, the Asahi Glass Foundation, Kayamori Foundation of Informational Science Advancement, and JST CREST Grant Numbers JPMJCR1401 and JPMJCR1402.

E-mail addresses: yamanaka@cis.iwate-u.ac.jp (Katsuhisa Yamanaka) edemaine@mit.edu (Erik D. Demaine) horiyama@al.ics.saitama-u.ac.jp (Takashi Horiyama) kawamura@inf.kyushu-u.ac.jp (Akitoshi Kawamura) nakano@cs.gunma-u.ac.jp (Shin-ichi Nakano) okamotoy@uec.ac.jp (Yoshio Okamoto) toshikis@ces.kyutech.ac.jp (Toshiki Saitoh) a.suzuki@ecei.tohoku.ac.jp (Akira Suzuki) uehara@jaist.ac.jp (Ryuhei Uehara) uno@nii.jp (Takeaki Uno)

1 Introduction

In this paper, we consider the following puzzle on graphs. Let $G = (V, E)$ be an undirected unweighted graph with vertex set V and edge set E. Suppose that each vertex in G has a color in $C = \{1, 2, \ldots, |C|\}$, $|C| \leq |V|$, and has a token of a color in C. Then, we wish to transform the current token-placement into the one such that a token of color i is placed on a vertex of color i for all vertices by a "sequential" token swapping. For a walk $W = \langle w_1, w_2, \ldots, w_k \rangle$[1] of G, a *sequential swapping* is to swap the two tokens on w_i and w_{i+1} in the order of $i = 1, 2, \ldots, k-1$. Intuitively, the token on w_1 is moved to w_k along W and for each $i = 2, 3, \ldots, k$, the token on w_i is shifted to w_{i-1}. Figure 1 shows an example of a sequential swapping. If there exists a color i such that the number of vertices of color i is not equal to the number of tokens of color i in a current token-placement, then we cannot transform the token-placement into the target one. Thus, without loss of generality, we assume that the number of vertices of color i for each $i = 1, 2, \ldots, |C|$ is equal to the number of tokens of the same color.

Our problem is regarded as a variation of the Fifteen Puzzle or 15 Puzzle [1]. If we assume that (1) vertices and tokens are labeled, (2) the moving token is designated, and (3) an input graph is a grid graph, our problem is same as the Fifteen Puzzle on $N \times N$ board. The vertex having the designated moving token corresponds to a "blank vertex", which has no token (pebble), in Fifteen Puzzle. Swapping the moving token with an adjacent token in Sequential Token Swapping problem corresponds to moving a token adjacent to the blank vertex into the blank vertex in the Fifteen Puzzle. As for generalizations of the Fifteen Puzzle, there are the following important results. Ratner and Warmuth [10] considered the Fifteen Puzzle on a $N \times N$ board. They demonstrated that the problem of finding the shortest solution of the Fifteen Puzzle on $N \times N$ board is NP-complete. Goldreich [6] generalized the problem to a game on graphs. He demonstrated that the problem of finding the shortest solution of the Fifteen Puzzle on graphs is NP-complete. Kornhauser et al. [8] and Wilson [12] also considered the problem of the Fifteen Puzzle on graphs. The problem setting in [8] includes a special case of Sequential Token Swapping problem. Suppose that the number of colors is equal to the number of vertices of an input graph and the moving token is designated in Sequential Token Swapping problem. This case of Sequential Token Swapping problem is included in the problem setting in [8] (the detail is explained in Section 2.2). See Demaine and Hearn's survey [4] on the Fifteen Puzzle and its related puzzles for further details.

Recently, Yamanaka et al. [13, 14] considered the same problem with the different swapping rule which is to swap any two tokens on adjacent vertices. Yamanaka et al. [13] dealt with the case where the number of colors is equal to the number of vertices, and showed a polynomial-time 2-approximation algorithm for trees and a polynomial-time exact algorithm for complete bipartite graphs. However, the complexity of the problem was not proved in the paper.[2]

[1] In this paper, we denote a walk of a graph by a sequence of vertices.

[2] Quite recently, hardness results were shown [2, 7, 9].

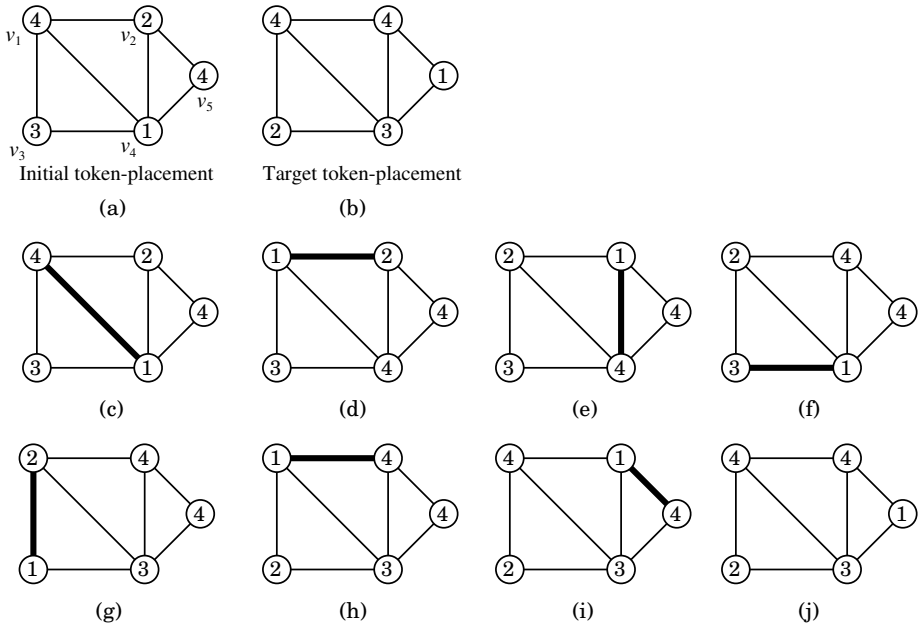

Figure 1: An example of Sequential Token Swapping. (a) An input graph and its initial token-placement. (b) The target token-placement. (c)–(j) A swapping sequence between (a) and (b). Token colors on vertices are written inside circles. We sequentially swap the token on v_4 along the walk $\langle v_4, v_1, v_2, v_4, v_3, v_1, v_2, v_5 \rangle$. For each solid line, the two tokens on its endpoints are swapped.

On the other hands, Yamanaka et al. [14] considered the more general case in which the number of colors is equal to or smaller than the number of vertices. They demonstrated that the problem is NP-complete when the number of colors is 3 or more, and otherwise the problem is polynomially solvable.

In this paper, we consider the sequential token swapping problem which asks to find the shortest walk W such that the sequential swapping along W gives the target token-placement. We first demonstrate an inapproximability of our problem even if the number of colors is 2. This result shows a difference on complexity between the problem in [14] and our problem in the sense that, when the number of colors is 2, the former is polynomially solvable, however the latter is computationally hard. Then, we present some positive results for restricted graph classes: trees, complete graphs, and cycles.

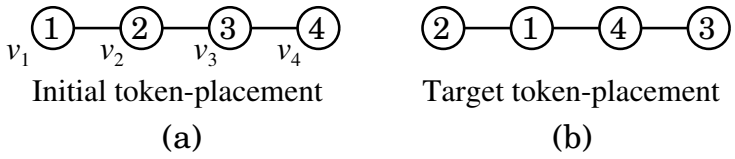

Figure 2: An example of Sequential Token Swapping. There is no swapping sequence between the two token-placements (a) and (b).

2 Preliminaries

2.1 Notations

In this paper, we assume without loss of generality that graphs are simple and connected. Let $G = (V, E)$ be an undirected unweighted graph with vertex set V and edge set E. We sometimes denote by $V(G)$ and $E(G)$ the vertex set and the edge set of G, respectively. We always denote $|V|$ by n. For a vertex v in G, let $N_G(v)$ be the set of all neighbors of v, that is, $N_G(v) = \{w \in V(G) \mid (v, w) \in E(G)\}$. Each vertex of a graph G has a color in $C = \{1, 2, \ldots, |C|\}$. We denote by $c(v) \in C$ the color of a vertex $v \in V$. A token is placed on each vertex in G. Each token also has a color in C. For a vertex v, we denote by $f(v)$ the color of the token placed on v. Then, we call the surjective function $f : V \to C$ a *token-placement* of G. Note that, since c is also a function from V to C, it can be regarded as a token-placement of G. Let f and f' be two token-placements of G. For a walk $W = \langle w_1, w_2, \ldots, w_h \rangle$ of G, a sequence $\mathcal{S} = \langle f_1, f_2, \ldots, f_h \rangle$ of token-placements is a *swapping sequence* of W between f and f' if the following three conditions (1)–(3) hold:

(1) $f_1 = f$ and $f_h = f'$;
(2) f_k is a token-placement of G for each $k = 1, 2, \ldots, h$; and
(3) f_k is obtained from f_{k-1} by swapping the two tokens on w_{k-1} and w_k for each $k = 2, 3, \ldots, h$.

Intuitively, a swapping sequence of W represents to move the token on w_1 to w_h along W. We call the token on w_1 in f the *moving token* of $\mathcal{S}$.

Let S be a sequence. Then, the length of S, denoted by $\mathrm{len}(S)$, is defined to be the number of elements in S minus one. The length of a swapping sequence $\mathcal{S}$, $\mathrm{len}(\mathcal{S})$, indicates the number of token-swaps in $\mathcal{S}$. For two token-placements f and c of G, we denote by $\mathrm{OPT}_{\mathrm{STS}}(G, f, c)$ the minimum length of a swapping sequence between f and c. Given two token-placements f and c of a graph G and a nonnegative integer ℓ, the Sequential Token Swapping problem is to determine whether or not $\mathrm{OPT}_{\mathrm{STS}}(G, f, c) \leq \ell$ holds. We call f and c the *initial* and *target* token-placements of G, respectively. We define that $\mathrm{OPT}_{\mathrm{STS}}(G, f, c) = \infty$ if there is no swapping sequence between f and c. Figure 2 is an example for which there is no swapping sequence.

2.2 Polynomial-length upper bound

We prove that, if there exists a swapping sequence between two token-placements, then the length of the sequence is polynomial.

To prove it, let us give some notations. A token-placement f is *colorful* if f consists of tokens each having a different color, that is the color set is $C = \{1, 2, \ldots, n\}$. We first define a restricted version of Sequential Token Swapping problem. Let f and c be colorful initial and target token-placements of a graph G. Suppose that we are given the vertex v having the moving token. Then, we denote by $\mathrm{OPT}_{\mathrm{CMSTS}}(G, f, c, v)$ the minimum length of a swapping sequence between f and c when the token on v is used as the moving token.

On the other hands, the problem setting in [8] by Kornhauser *et al.* can be regarded as a generalization of this restricted version of Sequential Token Swapping problem. In their problem setting, we are given a graph in which either each vertex has a labeled token or has no token. Then, a token can be moved to an adjacent blank vertex. Note that an input graph has one or more blank vertices. If the number of blank vertices is equal to 1, their problem is equivalent to (the restricted version of) our problem, since the blank vertex can be regarded as the vertex having the designated moving token. We have the following lemma from Theorem 2 in [8].[3]

Lemma 1 ([8]) *Let G be a graph, and let f, c be colorful initial and target token-placements. Suppose that the token on a vertex v of G is designated as the moving token. Then, if there exists a swapping sequence between the two token-placements, $OPT_{CMSTS}(G, f, c, v)$ is bounded from above by $O(n^3)$.*

Next, we define another restricted version of Sequential Token Swapping problem. Let f and c be colorful initial and target token-placements of a graph G. Then, we denote by $\mathrm{OPT}_{\mathrm{CSTS}}(G, f, c)$ the minimum length of a swapping sequence between f and c.[4] We have the following lemma.

Lemma 2 *Let G be a graph, and let f, c be colorful initial and target token-placements. Then, if there exists a swapping sequence between the two token-placements, $OPT_{CSTS}(G, f, c)$ is bounded from above by $O(n^3)$.*

Proof: Let $\mathcal{S}$ be a shortest swapping sequence between f and c. Let v be the vertex having the moving token for $\mathcal{S}$. Then, $\mathcal{S}$ is also a swapping sequence for $\mathrm{OPT}_{\mathrm{CMSTS}}(G, f, c, v)$. Thus, from Lemma 1, $\mathrm{OPT}_{\mathrm{CSTS}}(G, f, c)$ is bounded from above by $\mathrm{O}(n^3)$. □

Now, we are ready to prove the following theorem.

Theorem 1 *Let G be a graph, and let f, c be initial and target token-placements. Then, if there exists a swapping sequence between the two token-placements, $OPT_{STS}(G, f, c)$ is bounded from above by $O(n^3)$.*

[3] Actually, Theorem 2 in [8] is more general. The theorem includes the claim in our lemma.
[4] The moving token is not designated in this problem setting.

Proof: Let $\mathcal{S}$ be a shortest swapping sequence between f and c. Let u be a vertex, and let u' be the vertex such that, after the sequential swapping using $\mathcal{S}$, $f(u)$ is on u'. We reassign a unique color to $f(u)$ and $c(u')$. We do the same reassignment for each vertex. Then, we have the two colorful token-placements, denoted by f' and c', from f and c, respectively. The swapping sequence $\mathcal{S}$ is also a swapping sequence for $\mathrm{OPT}_{\mathrm{CSTS}}(G, f', c')$. Thus, from Lemma 2, $\mathrm{OPT}_{\mathrm{STS}}(G, f, c)$ is bounded from above by $\mathrm{O}(n^3)$. □

3 Inapproximability

In this section, we demonstrate the inapproximability of Sequential Token Swapping problem. To show the hardness result, we give a gap-preserving reduction from the following problem:

Problem: Maximum Vertex-Disjoint Path Cover on Undirected Bipartite Graphs
Instance: An undirected bipartite graph $G = (V, E)$ with vertex bipartition (X, Y) such that $|X| = |Y|$.
Question: Find a set of vertex-disjoint paths that cover all the vertices in G such that the paths contain the maximum number of edges.

If an input graph G is Hamiltonian, then a Hamiltonian path in the graph is an optimal solution and the number of edges in the path is $n - 1$, where n is the number of vertices in G. We can show that, for some constant ε, $(1-\varepsilon,1)$-gap Maximum Vertex-Disjoint Path Cover on Undirected Bipartite Graphs problem is NP-hard (This is proved by a reduction from Maximum Vertex-Disjoint Path Cover on Directed Graphs problem [3, 5, 11]. The definition of the problem and the proof are described in Appendix A). Thus, we give a gap-preserving reduction from the problem to Sequential Token Swapping problem with only 2 colors. Let $\mathrm{OPT}_{\text{U-MVDPC}}(G)$ denote the optimal value, which is the number of edges in paths in an optimal path cover, of Maximum Vertex-Disjoint Path Cover on Undirected Bipartite Graphs problem for an input graph G.

Theorem 2 *Let $G = (V, E)$ be an undirected bipartite graph with vertex bipartition (X, Y) such that $|X| = |Y|$. Then, there is a gap preserving reduction from* Maximum Vertex-Disjoint Path Cover on Undirected Bipartite Graphs *problem to* Sequential Token Swapping *problem that transforms G to a graph $H = (V_H, E_H)$ and its two token-placements f, c with 2 colors such that*

(1) if $OPT_{U\text{-}MVDPC}(G) = n - 1$, then $OPT_{STS}(H, f, c) = n - 1$ and
(2) if $OPT_{U\text{-}MVDPC}(G) < (1-\varepsilon)(n-1)$, then $OPT_{STS}(H, f, c) > (1+\varepsilon)(n-1)$,

where $n = |V|$.

Proof: Let G be an instance of Maximum Vertex-Disjoint Path Cover on Undirected Bipartite Graphs problem. Now we construct an instance

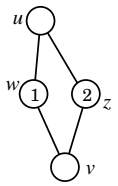

Figure 3: A connection gadget.

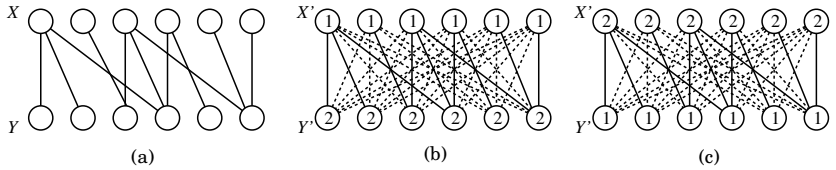

Figure 4: An example of reduction. (a) A bipartite graph $G = (V, E)$ with vertex bipartition (X, Y) such that $|X| = |Y|$. (b) The reduction graph H and its initial token-placement f. Connection gadgets are represented as dotted lines. (c) The target token-placement.

of Sequential Token Swapping problem, that is a graph, an initial token-placement, and a target one. We first construct a copy $G' = (V', E')$ of G, and denote its bipartition by (X', Y'). We set $f(u) = 1$ and $c(u) = 2$ for every vertex $u \in X'$, and set $f(v) = 2$ and $c(v) = 1$ for every vertex $v \in Y'$. We then insert "connection gadgets" for non-adjacent vertex pairs between X' and Y', as follows. Let $A = \{(u, v) \mid u \in X', v \in Y', \text{and } (u, v) \notin E'\}$. The *connection gadget* for $(u, v) \in A$ consists of two paths of length 2 connecting $u \in X'$ and $v \in Y'$ (see Figure 3). For the two intermediate vertices w and z in the paths, we set $f(w) = 1$ and $c(w) = 1$, and $f(z) = 2$ and $c(z) = 2$. We denote the obtained graph and its initial and target token-placements by H, f, and c, respectively. Figure 4 shows an example of the reduction graph. In the figure, connection gadgets are represented as dotted lines for convenience. The reduction graph, its initial token-placement, and its target token-placement can be constructed in polynomial time.

Now we show that, an optimal solution of the reduced instance of Sequential Token Swapping can be obtained from an optimal solution of an instance of Maximum Vertex-Disjoint Path Cover on Undirected Bipartite Graphs, as follows. Let $\mathcal{P}$ be a maximum vertex-disjoint path cover of G and let $cost(\mathcal{P}) = \sum_{P \in \mathcal{P}} \text{len}(P)$, where $\text{len}(P)$ is the number of edges in the path $P \in \mathcal{P}$. We construct a swapping sequence between f and c from $\mathcal{P}$. Let P_1 and P_2 be two paths in $\mathcal{P}$ such that P_1 contains an endpoint v_1 in X' and P_2 contains an endpoint v_2 in Y'. Then, we connect $v_1 \in X'$ and $v_2 \in Y'$ with a path of the connection gadget between v_1 and v_2. Note that v_1 and v_2 are not adjacent from optimality of $\mathcal{P}$. When a token is sequentially swapped from $v_1 \in X'$ to $v_2 \in Y'$, we choose the path with the intermediate vertex of color 2. Otherwise,

we choose the path with the intermediate vertex of color 1. Since $|X| = |Y|$, by repeating the above process, we can find a path spanning all vertices in $X' \cup Y'$. Note that, since $|X| = |Y|$ holds again, the number of paths whose endpoints are both in X' are equal to the number of paths whose endpoints are both in Y'. Then, by swapping the token on an endpoint to the other endpoint along the obtained path, we have the target token-placement.

Let $\mathcal{S}$ be the swapping sequence obtained from $\mathcal{P}$ by the above process. Now, we show that $\mathcal{S}$ is optimal. We assume for a contradiction that $\mathcal{S}'$ is a better solution than $\mathcal{S}$. Let W be the walk corresponding to $\mathcal{S}$, and let m be the the number of edges in W in connection gadgets. Similarly, let W' be the walk corresponding to $\mathcal{S}'$, and let m' be the number of edges in W' in connection gadgets. Then, W' is a path spanning vertices in $X' \cup Y'$. More precisely, a vertex in $X' \cup Y'$ appears once in W' and an intermediate vertex in a connection gadget appears at most once in W'. Note that every vertex v in H does not appear twice or more, since to visit v twice or more produces redundant token-swaps. Hence, we can construct a path cover from W', as follows. First, we split W' into subsequences by regarding intermediate vertices as boundaries and removing the intermediate vertices. Then we obtain the set of subsequences. Since W' spans the vertices in $X' \cup Y'$ and visits each vertex at most once, the set is a path cover of G. Let $\mathcal{P}'$ denote the path cover obtained from W'.

Now, to derive a contradiction, we first focus on the two path covers $\mathcal{P}$ and $\mathcal{P}'$, and then we derive an inequality between $|\mathcal{P}|$ and $|\mathcal{P}'|$, more specifically the number of paths in $\mathcal{P}$ is smaller than or equal to the number of paths in $\mathcal{P}'$. Since each of $\mathcal{P}$ and $\mathcal{P}'$ visits every vertex in $X' \cup Y'$ exactly once, we have $\mathit{cost}(\mathcal{P}) = (n-1) - |\mathcal{P}| + 1$ and $\mathit{cost}(\mathcal{P}') = (n-1) - |\mathcal{P}'| + 1$. Then, we also have $\mathit{cost}(\mathcal{P}) \geq \mathit{cost}(\mathcal{P}')$, since $\mathcal{P}$ is an optimal path cover. Thus, we obtain $|\mathcal{P}| \leq |\mathcal{P}'|$.

Next we focus on the two walks W and W' and we also derive an inequality between $|\mathcal{P}|$ and $|\mathcal{P}'|$. Since $\mathcal{S}'$ is a shorter swapping sequence than $\mathcal{S}$, $\mathrm{len}(W) > \mathrm{len}(W')$ holds. Each of W and W' visits every vertex in $X' \cup Y'$ exactly once. Therefore, the number of edges in connection gadgets in W is greater than the number of edges in connection gadgets in W', that is $m > m'$. Since W and W' include exactly two edges in each connection gadget in W and W', we have $|\mathcal{P}| = \frac{m}{2} + 1$ and $|\mathcal{P}'| = \frac{m'}{2} + 1$, respectively. Therefore, $|\mathcal{P}| > |\mathcal{P}'|$ holds, which contradicts to $|\mathcal{P}| \leq |\mathcal{P}'|$. Therefore, $\mathcal{S}$ is an optimal swapping sequence of H, f, and c.

Now, we demonstrate the correctness of claims (1) and (2).

If $\mathrm{OPT}_{\text{U-MVDPC}}(G) = n - 1$, then G has a Hamiltonian path P. Note that an endpoint of P is in X and the other is in Y since $|X| = |Y|$. By sequentially swapping the token on an endpoint of P in H to the other endpoint, we obtain the target token-placement. Hence, $\mathrm{OPT}_{\mathrm{STS}}(H, f, c) \leq n - 1$. Since we must visit every vertex v in H such that $f(v) \neq c(v)$, the number of such vertices minus one is a lower bound for $\mathrm{OPT}_{\mathrm{STS}}(H, f, c)$. Therefore $\mathrm{OPT}_{\mathrm{STS}}(H, f, c) = n - 1$.

Let $\mathcal{S}$ be an optimal swapping sequence obtained from an optimal path cover

$\mathcal{P}$ of G by the transformation above. The length of $\mathcal{S}$ is equal to the sum of the number of edges in $\mathcal{P}$ and the number of edges in the connection gadgets which used in $\mathcal{S}$. (Recall that at most two edges in each connection gadget are used in $\mathcal{S}$.) Therefore we have the following equation:

$$\begin{aligned} \mathrm{len}(\mathcal{S}) &= cost(\mathcal{P}) + 2(|\mathcal{P}| - 1) \\ &= (n-1) + (|\mathcal{P}| - 1) \end{aligned} \tag{1}$$

If $\mathrm{OPT}_{\text{U-MVDPC}}(G) < (1-\varepsilon)(n-1)$, then $|\mathcal{P}|$ is bounded from below:

$$\begin{aligned} |\mathcal{P}| &= (n-1) - cost(\mathcal{P}) + 1 \\ &> (n-1) - (1-\varepsilon)(n-1) + 1 \\ &= \varepsilon(n-1) + 1. \end{aligned} \tag{2}$$

From Equality (1) and Inequality (2), we have $\mathrm{OPT}_{\text{STS}}(H, f, c) > (1+\varepsilon)(n-1)$. □

From Theorem 2, even if the number of colors is 2, there is no polynomial-time $(1+\varepsilon)$-approximation algorithm, unless $P = NP$.

4 Polynomial-time algorithms

In the previous section, we showed the inapproximability of Sequential Token Swapping problem even if the number of colors is 2. On the other hand, if graph classes are restricted, the problem can be solved in polynomial time for any number of colors. In this section, we consider the problem of computing the minimum numbers of token-swaps for trees, complete graphs, and cycles.

4.1 Trees

Let $G = (V, E)$ be a tree, and let f and c be initial and target token-placements of G. We show below that Sequential Token Swapping problem on trees can be solved in linear time.

Let v be a leaf of G with $f(v) = c(v)$. We can observe that the token on v is never swapped in any shortest swapping sequence. Now, we can reduce the instance, as follows. We repeatedly remove a leaf v with $f(v) = c(v)$ one by one, until the tree has no such leaf. Let G' be the obtained tree, and let f' and c' be the initial and target token-placements on the vertices of G'. See Figure 5 for an example. It is easy to see that a shortest swapping sequence of the instance (G', f', c') is also a shortest swapping of the instance (G, f, c).

Now, let us consider to solve the reduced instance. First, we have the following observation.

Lemma 3 *Let G' be a reduced tree, and let f' and c' be initial and target token-placements. Then, any shortest swapping sequence on G' forms a simple path of G'.*

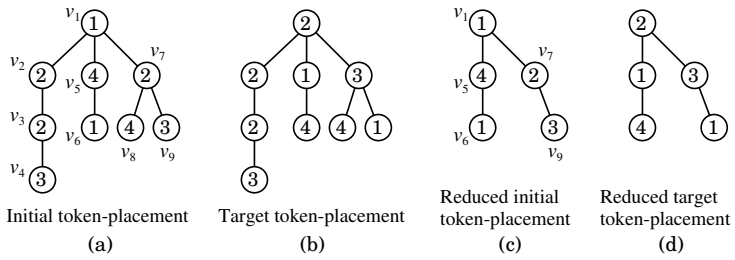

Figure 5: An example of the reduction. (a) An initial token-placement. (b) A target token-placement. (c) The reduced initial token-placement from (a). (d) The reduced target token-placement from (b).

Proof: Let $\mathcal{S} = \langle f_1, f_2, \ldots, f_h \rangle$ be a shortest swapping sequence between f' and c' with the moving token t. Let us assume that for a contradiction t visits the same vertex two or more times. Suppose that t is on a vertex v in both f_i and f_j, $i < j$, and t is not on v in f_k, $i < k < j$. Then, we can observe that $f_i = f_j$ holds, since G' is a tree. Hence, $\mathcal{S}' = \langle f_1, f_2, \ldots, f_i, f_{j+1}, f_{j+2}, \ldots, f_h \rangle$ is also a swapping sequence on G' and is shorter than $\mathcal{S}$, which is a contradiction. □

We have the following cases.

Case 1: G' has exactly two leaves.

In this case, G' is a path and G' has two leaves u and v with $f'(u) \neq c'(u)$ and $f'(v) \neq c'(v)$. From Lemma 3, any shortest swapping sequence forms a simple path. Any simple path that passes the two leaves in G' has the two leaves as its endpoints. The unique path between the two leaves is the only possible shortest swapping sequence. We can check whether or not such path is a swapping sequence in $\mathrm{O}(n)$ time.

Case 2: G' has three or more leaves.

Since there is no simple path that passes three or more leaves of G', in this case, (G, f, c) is no-instance.

Therefore, we have the following theorem.

Theorem 3 *For a tree G, an initial token-placement f, and a target token-placement c, one can compute $OPT_{STS}(G, f, c)$ in $O(n)$ time.*

4.2 Complete graphs

Let $G = (V, E)$, f, and c be a complete graph, an initial token-placement and a target token-placement, respectively. Let $C = \{1, 2, \ldots, |C|\}$, $|C| \leq n$, be the set of colors. For a token-placement f, let $V'(f) \subseteq V$ be the set of vertices v such that $f(v) \neq c(v)$.

We first introduce a multiple digraph $D(f) = (V_D(f), E_D(f))$ called the *conflict graph* as follows:

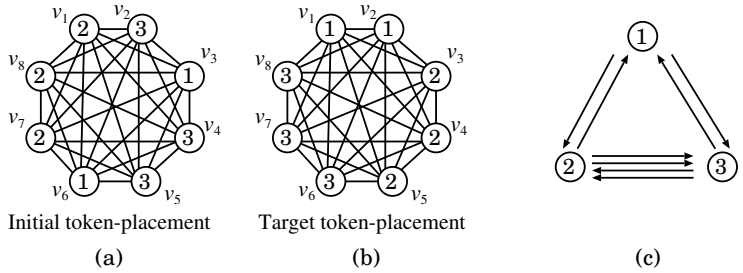

Figure 6: An example of a conflict graph. (a) An initial token-placement. (b) A target token-placement. (c) The conflict graph for (a) and (b).

- $V_D(f) = \{i \mid i \in C \text{ and } i = f(v) \text{ for some } v \in V'(f)\}$.
- $E_D(f) = \{(f(v), c(v)) \mid v \in V'(f)\}$

Note that $V_D(f)$ corresponds to the set of the colors each of which is a color of a token placed in some vertex in $V'(f)$. That is, a color c is not included in $V_D(f)$ if c does not appear as the color of the token on any vertex in $V'(f)$. Each arc $(f(v), c(v)) \in E_D(f)$ corresponds to a vertex $v \in V'(f)$. Since, for each color in C, the number of vertices in G of the color is equal to the number of tokens of the same color, each node in conflict graph $D(f)$ has the same numbers of incoming edges and outgoing edges. Thus, each connected component of $D(f)$ has a directed Euler cycle. Therefore, $D(f)$ consists of only strongly connected components. Let $s(f)$ be the number of strongly connected components in $D(f)$. Then we claim the following:

Claim 4 *For a complete graph, an initial token-placement f, and a target token-placement c, $OPT_{STS}(G, f, c) = |V'(f)| + s(f) - 2$.*

The claim above immediately implies the following theorem, since we can calculate the values of $|V'(f)|$ and $s(f)$ in $\mathrm{O}(n)$ time.

Theorem 5 *For a complete graph G, an initial token-placement f, and a target token-placement c one can compute $OPT_{STS}(G, f, c)$ in $O(n)$ time.*

In the rest of this section, we prove the above claim. First we show that $\mathrm{OPT}_{\mathrm{STS}}(G, f, c) \leq |V'(f)| + s(f) - 2$ by constructing a swapping sequence of length $|V'(f)|+s(f)-2$, then we show that $\mathrm{OPT}_{\mathrm{STS}}(G, f, c) \geq |V'(f)|+s(f)-2$ by using a potential function.

Upper bound

We present an algorithm that finds a swapping sequence of length $|V'(f)| + s(f) - 2$. Let C_i, $i = 1, 2, \ldots, s(f)$, be a strongly connected component of $D(f)$. Recall that C_i has a directed Euler cycle. We here denote a directed Euler cycle as a sequence of directed edges: For each C_i, $i = 1, 2, \ldots, s(f)$, let

$\langle e_{i,1}, e_{i,2}, \ldots, e_{i,t_i} \rangle$ denote a directed Euler cycle of C_i, where t_i is the number of edges in C_i. For each $e_{i,j}$, let $v_{i,j}$ be the corresponding vertex in $V'(f)$. Let

$$\begin{aligned} W = \ \langle \ & v_{1,1}, v_{1,2}, \ldots, v_{1,t_1}, \\ & v_{2,1}, v_{2,2}, \ldots, v_{2,t_2}, v_{1,t_1}, \\ & v_{3,1}, v_{3,2}, \ldots, v_{3,t_3}, v_{1,t_1}, \\ & \quad \ldots, \\ & v_{s(f),1}, v_{s(f),2}, \ldots, v_{s(f),t_{s(f)}}, v_{1,t_1} \rangle. \end{aligned}$$

be a walk in $V'(f)$. Now we show that the length of W is $|V'(f)| + s(f) - 2$ and the target token-placement c is obtained by swapping along W. This immediately implies that we have a swapping sequence between f and c of length $|V'(f)| + s(f) - 2$.

Since each vertex in $V'(f) \setminus \{v_{1,t_1}\}$ appears exactly once and v_{1,t_1} appears $s(f)$ times, $\mathrm{len}(W) = |V'(f)| + s(f) - 2$. Let f' be the token-placement obtained by sequentially swapping the token on $v_{1,1}$ along W, and let $v_{i,j}$, $1 \leq j \leq t_i - 1$, denote a vertex in $V'(f) \setminus \{v_{1,t_1}\}$. Since $v_{i,j}$ appears exactly once, $f'(v_{i,j}) = f(v_{i,j+1})$ holds. Recall that $v_{i,j}$ and $v_{i,j+1}$ correspond to $e_{i,j}$ and $e_{i,j+1}$ in the directed Euler cycle of C_i, respectively. Thus, $f(v_{i,j+1}) = c(v_{i,j})$ holds. Next, let us consider the vertex v_{i,t_i} in $V'(f) \setminus \{v_{1,t_1}\}$. It can be observed that, while we traverse from $v_{i,1}$ to v_{i,t_i}, v_{1,t_1} has $f(v_{i,1})$, since the sequence $\langle e_{i,1}, e_{i,2}, \ldots, e_{i,t_i} \rangle$ is an Euler cycle, $f(v_{i,1}) = c(v_{i,t_i})$ holds. Thus, v_{i,t_i} has its expected token after the token-swaps on vertices of C_i. Finally, we have $f'(v_{1,t_1}) = c(v_{1,t_1})$, since $f(v_{1,1}) = c(v_{1,t_1})$ holds.

Lower bound

Now we show that the length of any swapping sequence is at least $|V'(f)| + s(f) - 2$. Let $\mathcal{S} = \langle f_1, f_2, \ldots, f_h \rangle$ be an arbitrary swapping sequence between f and c for a walk $W = \langle w_1, w_2, \ldots, w_h \rangle$. Note that $f_1 = f$ and $f_h = c$. Let $D(f_i)$ be the conflict graph for each token-placement.

First, we define a potential function $p(f_i)$. Let $p_1(f_i)$ be the number of vertices in $V'(f_i)$ except the vertex with the moving token x, and let $p_2(f_i)$ be the number of strongly connected components in $D(f_i)$ that do not include x. We define the potential function as $p(f_i) = p_1(f_i) + p_2(f_i)$. Then,

$$p(f_i) \geq (|V'(f_i)| - 1) + (s(f_i) - 1)$$

and

$$p(c) = 0.$$

Note that for any token-placement $f_i \neq c$,

$$p(f_i) \geq 1.$$

Now we show that the potential function decreases by at most one for each token-swap. For each token-swap, we swap the moving token x on w_i with

another token on w_{i+1}. Note that, from the definition of conflict graphs, $D(f_{i+1})$ is obtained from $D(f_i)$ by removing the two edges $(f_i(w_i), c(w_i))$ and $(f_i(w_{i+1}), c(w_{i+1}))$ and inserting the two edges $(f_i(w_i), c(w_{i+1}))$ and $(f_i(\ w_{i+1}),$ $c(w_i))$. We have the following cases.

Case 1: The colors of $f_i(w_i)$, $c(w_i)$, $f_i(w_{i+1})$, and $c(w_{i+1})$ are different from each other.

In this case, $p_1(f_i) = p_1(f_{i+1})$ holds, since the colors of $f_i(w_i)$, $c(w_i)$, $f_i(w_{i+1})$, and $c(w_{i+1})$ are different. Therefore, we focus only on the value of $p_2(f_i)$ in the following subcases.

Case 1-1: $f_i(w_i)$ and $c(w_i)$ are in the same strongly connected component in $D(f_i)$.

Let C_a be the strongly connected component of $D(f_i)$ including $f_i(w_i)$ and $c(w_i)$. We have the following subcases.

Case 1-1-1: C_a includes neither $f_i(w_{i+1})$ nor $c(w_{i+1})$.

We first assume that $f_i(w_{i+1})$ and $c(w_{i+1})$ are included in the same strongly connected component of $D(f_i)$, denoted by C_b. Then, C_a and C_b are combined as a strongly connected component in $D(f_{i+1})$. Hence, the number of strongly connected components decreases by one. Otherwise, $f_i(w_{i+1})$ and $c(w_{i+1})$ are included in different components in $D(f_i)$. Then, the number of strongly connected components increases by one if C_a is divided into two components. Therefore, in this case, the value of $p_2(f_i)$ decreases by at most one.

Case 1-1-2: C_a includes $f_i(w_{i+1})$ or $c(w_{i+1})$.

The number of strongly connected components does not change, and also the value of $p_2(f_i)$ does not change.

Case 1-2: $f_i(w_i)$ and $c(w_i)$ are in different strongly connected components in $D(f_i)$.

Let C_a and C_b be the strongly connected components including $f_i(w_i)$ and $c(w_i)$, respectively. If C_a includes both $f_i(w_{i+1})$ and $c(w_{i+1})$, we can confirm that the number strongly connected components does not change. Similarly, for all possible cases, we can observe that the number strongly connected components does not change. Hence, in this case, the value of $p_2(f_i)$ does not change.

Case 2: Only a pair among $f_i(w_i)$, $c(w_i)$, $f_i(w_{i+1})$, and $c(w_{i+1})$ has the same color.

Case 2-1: $f_i(w_i) = c(w_i)$.

The value of $p_1(f_i)$ increases by one after swapping the two tokens on w_i and w_{i+1}. The number of strongly connected components in $D(f_i)$ decreases by one if the strongly connected component including the node $f_i(w_i)$ and the strongly connected component including the edge $(f_i(w_{i+1}), c(w_{i+1}))$ are different. Otherwise the number of strongly connected components does not change. Therefore, $p(f_i)$ decreases by at most one.

Case 2-2: $f_i(w_i) = f_i(w_{i+1})$ or $c(w_i) = c(w_{i+1})$.

After swapping the two tokens on w_i and w_{i+1}, $D(f_i)$ does not change. Hence, $p(f_i)$ does not change.

Case 2-3: $f_i(w_i) = c(w_{i+1})$.

Since $f_i(w_i) = c(w_{i+1})$ holds, $p_1(f_i)$ decreases by one. The number of strongly connected components in $D(f_i)$ increases by one if the set of the two edges $(f_i(w_i), c(w_i))$ and $(f_i(w_{i+1}), c(w_{i+1}))$ is a cut of $D(f_i)$. Otherwise, the number of strongly connected components does not change. Therefore, $p(f_i)$ decreases by at most one.

Case 2-4: $c(w_i) = f_i(w_{i+1})$.

Symmetric to Case 2-3.

Case 2-5: $f_i(w_{i+1}) = c(w_{i+1})$.

Symmetric to Case 2-1.

Case 3: Only a triple among $f_i(w_i)$, $c(w_i)$, $f_i(w_{i+1})$, and $c(w_{i+1})$ has the same color.

The values of $p_1(f_i)$ and $p_2(f_i)$ do not change, since the conflict graph does not change. Hence, $p(f_i)$ does not change.

Case 4: $f_i(w_i) = c(w_i) = f_i(w_{i+1}) = c(w_{i+1})$.

Similar to Case 3, the values of $p_1(f_i)$ and $p_2(f_i)$ do not change. Hence, $p(f_i)$ does not change.

Therefore, $p(f_{i+1}) \geq p(f_i) - 1$ holds. Hence, the length of any swapping sequence f is at least $p(f)$. This completes the proof of Theorem 5.

4.3 Cycles

In this section, we present two algorithms for cycles. The first algorithm runs in $O(n^4)$ time, while the second one is faster and runs in $O(n^2)$ time. Let $G = (V, E)$ be a cycle with n vertices, and let f and c be initial and target token-placements of G. For cycles, the moving token goes clockwise or counterclockwise. In the shortest sequential swapping, the moving token does not turn back, since changing the direction produces redundant token-swaps.

Lemma 4 *Let G be a cycle, and let f and c be initial and target token-placements. In any shortest swapping sequence on G, the moving token always goes either clockwise or counterclockwise.*

Proof: Let $\mathcal{S} = \langle f_1, f_2, \ldots, f_h \rangle$ be a shortest swapping sequence between f and c with the moving token t. Let us assume that for a contradiction t turns back before and after f_i and f_i is the first token-placement where t turns back. Without loss of generality, we assume that t goes clockwise before f_i and counterclockwise after f_i. Then, we can observe that $f_{i-1} = f_{i+1}$ holds. Hence, $\mathcal{S}' = \langle f_1, f_2, \ldots, f_{i-1}, f_{i+2}, \ldots, f_h \rangle$ is also a swapping sequence between f and c on G and is shorter than $\mathcal{S}$, which is a contradiction. □

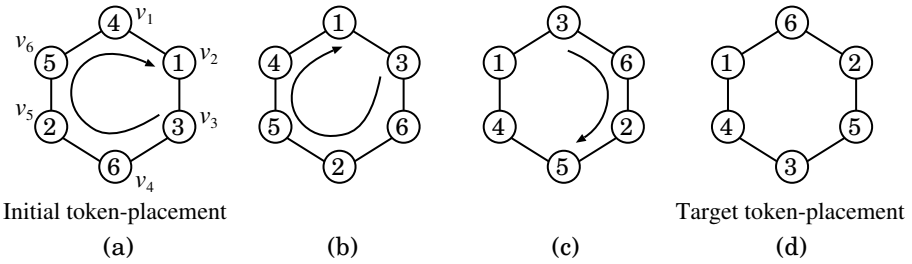

Figure 7: An example of a swapping sequence in a cycle. We choose the token 3 as the moving token and rotate it clockwise. (a) An initial token-placement. (b) The token-placement after 1 rotation of the token 3. (c) The token-placement after 2 rotations. (d) The token-placement after 2 rotations and 3 token-swaps.

From the lemma above, the moving token always goes either clockwise or counterclockwise. The optimal solution is the shortest sequential swapping among both directions. Thus, in this section, we suppose that the moving token always goes clockwise, since the same discussion can be applied to the other direction.

Naïve algorithm

We denote vertices in clockwise order on the cycle by $\langle v_1, v_2, \ldots, v_n \rangle$. First, we define the following $n \times n$ table $T[x][k]$:

$$T[x][k] = \begin{cases} 1 & f(v_x) = c(v_{(x-k) \bmod n}) \\ 0 & \text{otherwise.} \end{cases}$$

The value of $T[x][k]$ represents whether or not the token on v_x is placed on its expected vertex after the token goes counterclockwise by k token-swaps. Using this table, we make sure the token-placement after a sequential swapping is identical to the target one.

If we move the token on a vertex v_x clockwise with a sequential swapping of length $n-1$, then all other tokens are shifted once counterclockwise. Similarly, if we move the token on v_x clockwise with a sequential swapping of length $i(n-1)$, for $i = 1, 2, \ldots, n-1$, then all other tokens are shifted i times counterclockwise. Thus, a sequential token-swap of length $i(n-1)+j$ moves each token on v_w, $w = x+1, x+2, \ldots, x+j \pmod n$, $i+1$ times counterclockwise and each token on v_w, $w = x-1, x-2, \ldots, x+j+1 \pmod n$, i times counterclockwise. See Figure 7. Therefore, we have the following observation.

Observation 6 *The token-placement obtained by a sequential swapping of length $i(n-1)+j$ with the moving token on v_x is identical to the target one if and only if*

(1) $T[w][i+1] = 1$ *for each* $w = x+1, x+2, \ldots, x+j \pmod n$

(2) $T[w][i] = 1$ *for each* $w = x-1, x-2, \ldots, x+j+1 \pmod n$

(3) $T[x][i-j]=1$.

From this observation, we have a naïve algorithm. We denote a candidate of a solution by a triple (x,i,j) for $1\leq x\leq n$, $1\leq i\leq n-1$, and $1\leq j\leq n-1$. A triple (x,i,j) is *feasible* if it satisfies the above three conditions. The naïve algorithm simply investigates whether or not every triple (x,i,j) is feasible, then returns the triple that minimizes the value of $(n-1)i+j$ among all the feasible triples. This algorithm runs in $\mathrm{O}(n^4)$ time.

Theorem 7 *For a cycle G, an initial token-placement f, and a target token-placement c one can compute $OPT_{STS}(G,f,c)$ in $O(n^4)$ time.*

Improvement

In this subsection, we improve the running time of the naïve algorithm. We construct three other tables that store auxiliary information to efficiently check the conditions in Observation 6.

First we define the table T'. For a vertex v_x, $1\leq x\leq n$, and an integer k, $1\leq k\leq n-1$, $T'[x][k]$ stores the maximum index s such that $T[w][k]=1$ for all $w=x+1,x+2,\ldots,x+s \pmod n$. Intuitively, the entry $s=T'[x][k]$ means that, after k token-swaps, all consecutive tokens $f(w)$, $w=x+1,x+2,\ldots,x+s \pmod n$, are placed on their expected vertices, $f(x+s+1 \bmod n)$ is placed on an unexpected vertex. Similarly, we define the table T'', as follows. For a vertex v_x, $1\leq x\leq n$, and an integer k, $1\leq k\leq n-1$, $T''[x][k]$ stores the maximum index s such that $T[w][k]=1$ for any $w=x-1,x-2,\ldots,x-s \pmod n$. The table T'' focuses on the consecutive tokens from v_x in the opposite direction of T'. We also define the table T'''. The table $T'''[x][k]$ stores the maximum index s such that $T[x][\ell]=0$ for any $\ell=k,k-1,\ldots,k-s+1 \pmod n$. Intuitively, this entry means how many token-swaps we need to place the moving token, which is placed on v_x in f, on its expected vertex, after the token is swapped k times counterclockwise.

Our goal is to find the feasible triple (x,i,j) that minimizes the value of $(n-1)i+j$. To find such a triple, for every pair of (x,i), we find the smallest j such that the triple (x,i,j) is feasible. Among them, the triple that minimizes the value of $(n-1)i+j$ is a desired solution.

Now we describe the algorithm. Suppose we are given a pair of (x,i), $1\leq x\leq n$ and $1\leq i\leq n-1$. First, we investigate a range of j using the tables. Since a feasible triple needs to satisfy the first and second conditions in Observation 6, we have two ranges $j\leq T'[x][i+1]$ and $j\geq n-T''[x][i]-1$. Let $j_{min}\leq j\leq j_{max}$ be the range of j which satisfies the above two inequalities. Note that, if the range is empty, it implies that there is no feasible triple for the given pair (x,i) and thus the algorithm returns false. Then, we investigate whether there is j, $j_{min}\leq j\leq j_{max}$, with the third condition in Observation 6. This can be checked by $j_{min}+T'''[x][i-j_{min}]\leq j_{max}$. If the inequality is true, then the algorithm returns $j=j_{min}+T'''[x][i-j_{min}]$. Note that this value is the minimum j for the given pair (x,i) from the definition of T'''. Otherwise, the algorithm returns false. Therefore, we have the following theorem.

Theorem 8 *For a cycle G, an initial token-placement f, and a target token-placement c, one can compute $OPT_{STS}(G, f, c)$ in $O(n^2)$ time.*

Proof: For each pair of (x, i), $1 \leq x \leq n$ and $1 \leq i \leq n-1$, the algorithm returns the minimum index j such that (x, i, j) is feasible. This can be done in constant time using the three tables. Hence, the total running time is $\mathrm{O}(n^2)$.

Now, we give algorithms and their running time to construct the four tables T, T', T'', and T'''. The table T can be constructed in $\mathrm{O}(n^2)$ time, since each entry is computed in constant time. From now on, we only describe how to construct T' in $\mathrm{O}(n^2)$ time from T, since the other two tables are constructed by the similar way in the same running time.

For each $k = 1, 2, \ldots, n-1$, we construct $T'[w][k]$ for $w = 1, 2, \ldots, n$. We regard $T[w][k]$ for $w = 1, 2, \ldots, n$ as a "cyclic" 0-1 binary string, that is the next element of $T[n][k]$ is $T[1][k]$. Let $L = \langle s_1, s_2, \ldots, s_z \rangle$ be the sequence of all the indices such that $1 \leq s_1 < s_2 < \cdots < s_z \leq n$ and $T[s_\ell][k] = 0$ for each $\ell = 1, 2, \ldots, z$. If there is no such index, then $T[w][k] = 1$ for all $w = 1, 2, \ldots, n$. In this case, we set $T'[w][k] = \infty$ as a special case. If L has only one index, denote by s_1, then $T[s_1][k] = 0$ and $T[w][k] = 1$ for all $w = 1, 2, \ldots, s_1 - 1, s_1 + 1, \ldots, n$. In this case, we set $T'[w][k] = s_1 - w - 1$ for each $w = 1, 2, \ldots, s_1 - 1$ and $T'[w][k] = n - w + s_1 - 1$ for each $w = s_1, s_1 + 1, \ldots, n$. Now, we suppose that L has at least two indices. Let $s_\ell (\neq s_z)$ be an index in L. Then, for each $w = s_\ell, s_\ell + 1, \ldots, s_{\ell+1} - 1$, $T'[w][k] = s_{\ell+1} - w - 1$ always holds. Next let us consider the case of $s_\ell = s_z$. In this case, we determine the entries of $T'[w][k]$ for elements from s_z to s_1. We have the following two cases.

Case 1: $s_1 > 1$.

For $w = 1, 2, \ldots, s_1 - 1$, we set $T'[w][k] = s_1 - w - 1$. For $w = s_z, s_z + 1, \ldots, n$, we need to count the number of the consecutive ones in T from w to $s_1 - 1$, and hence we set $T'[w][k] = n - w + T'[1][k] + 1$.

Case 2: $s_1 = 1$.

In this case, $T[1][k] = 0$ holds. For $w = s_z, s_z + 1, \ldots, n$, we set $T'[w][k] = n - w$.

Therefore, each entry of T' is computed in constant time. Thus, we can construct T' in $\mathrm{O}(n^2)$ time. □

References

[1] E. Berlekamp, J. Conway, and R. Guy. *Winning Ways for Your Mathematical Plays*. Taylor & Francis, 2 edition, 2003.

[2] É. Bonnet, T. Miltzow, and P. Rzążewski. Complexity of token swapping and its variants. *Algorithmica*, 80(9):2656–2682, 2018. doi:10.1007/s00453-017-0387-0.

[3] J. Bosboom, E. D. Demaine, M. L. Demaine, A. Hesterberg, P. Manurangsi, and A. Yodpinyanee. Even $1\times n$ edge-matching and jigsaw puzzles are really hard. *CoRR*, abs/1701.00146, 2017. URL: http://arxiv.org/abs/1701.00146.

[4] E. Demaine and R. Hearn. *Playing games with algorithms: Algorithmic combinatorial game theory*, volume 56, pages 3–56. Cambridge University Press, 2009.

[5] Engebretsen. An explicit lower bound for tsp with distances one and two. *Algorithmica*, 35(4):301–319, 2003. doi:10.1007/s00453-002-1001-6.

[6] O. Goldreich. *Finding the Shortest Move-Sequence in the Graph-Generalized 15-Puzzle Is NP-Hard*, volume 6650 of *LNCS*, pages 1–5. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-22670-0_1.

[7] J. Kawahara, T. Saitoh, and R. Yoshinaka. The time complexity of the token swapping problem and its parallel variants. In S. Poon, M. S. Rahman, and H. Yen, editors, *WALCOM: Algorithms and Computation, 11th International Conference and Workshops, WALCOM 2017*, volume 10167 of *Lecture Notes in Computer Science*, pages 448–459. Springer, 2017. doi:10.1007/978-3-319-53925-6_35.

[8] D. Kornhauser, G. Miller, and P. Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *25th Annual Symposium on Foundations of Computer Science, 1984.*, pages 241–250, 1984. doi:10.1109/SFCS.1984.715921.

[9] T. Miltzow, L. Narins, Y. Okamoto, G. Rote, A. Thomas, and T. Uno. Approximation and hardness of token swapping. In P. Sankowski and C. D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016*, volume 57 of *LIPIcs*, pages 66:1–66:15. Schloss Dagstuhl, 2016. doi:10.4230/LIPIcs.ESA.2016.66.

[10] D. Ratner and M. Warmuth. The (n^2-1)-puzzle and related relocation problems. *Journal of Symbolic Computation*, 10(2):111 – 137, 1990. doi:10.1016/S0747-7171(08)80001-6.

[11] S. Vishwanathan. An approximation algorithm for the asymmetric travelling salesman problem with distances one and two. *Information Processing Letters*, 44(6):297 – 302, 1992. doi:10.1016/0020-0190(92)90103-3.

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

[13] 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. *Theoretical Computer Science*, 586:81 – 94, 2015. Fun with Algorithms. doi:10.1016/j.tcs.2015.01.052.

[14] K. Yamanaka, T. Horiyama, D. G. Kirkpatrick, Y. Otachi, T. Saitoh, R. Uehara, and Y. Uno. Swapping colored tokens on graphs. In F. Dehne, J. Sack, and U. Stege, editors, *Algorithms and Data Structures - 14th International Symposium, WADS 2015*, volume 9214 of *Lecture Notes in Computer Science*, pages 619–628. Springer, 2015. doi:10.1007/978-3-319-21840-3_51.

A Inapproximability of Maximum Vertex-Disjoint Path Cover on Undirected Bipartite Graphs

In this section, we demonstrate inapproximability of Maximum Vertex-Disjoint Path Cover on Undirected Bipartite Graphs problem in Section 3. We give a gap-preserving reduction from a maximum vertex-disjoint path cover problem on directed graphs with a degree bound. A formal definition of the problem is described below.

We use the following notations. For a directed graph $D = (V_D, E_D)$, we denote in-degree and out-degree of a vertex v in V_D by $d^-_D(v)$ and $d^+_D(v)$. Also we denote the set of predecessors and successors of v by $N^-_D(v)$ and $N^+_D(v)$. Now we define Maximum Vertex-Disjoint Path Cover on Directed Graphs problem, as follows.

Problem: Maximum Vertex-Disjoint Path Cover on Directed Graphs [3, 5, 11]
Instance: A directed graph $D = (V_D, E_D)$ such that any v in V_D holds either (1) $d^-_D(v) = 1$ and $d^+_D(v) = 2$, (2) $d^-_D(v) = 2$ and $d^+_D(v) = 1$, or (3) $d^-_D(v) = 2$ and $d^+_D(v) = 2$.
Question: Find a set of vertex-disjoint (directed) paths that cover all the vertices in D such that the paths contain the maximum number of edges.

It is known that $(1, 1-\varepsilon)$-gap Maximum Vertex-Disjoint Path Cover on Directed Graphs problem is NP-hard [3, 5].

We denote the optimal value of Maximum Vertex-Disjoint Path Cover on Directed Graphs problem for an input graph D by $\mathrm{OPT}_{\text{D-MVDPC}}(D)$, and the optimal value of Maximum Vertex-Disjoint Path Cover on Undirected Bipartite Graphs problem for an input graph G by $\mathrm{OPT}_{\text{U-MVDPC}}(G)$.

Theorem 9 *There is a gap preserving reduction from* Maximum Vertex-Disjoint Path Cover on Directed Graphs *problem to* Maximum Vertex-Disjoint Path Cover on Undirected Bipartite Graphs *problem that transforms a directed graph $D = (V_D, E_D)$, where $|V_D| \geq 2$, to a bipartite graph $G = (V_G, E_G)$, where $V_G = X \cup Y$ and $|X| = |Y|$, such that*

(1) if $OPT_{D\text{-}MVDPC}(D) = |V_D| - 1$, then $OPT_{U\text{-}MVDPC}(G) = |V_G| - 1$ and
(2) if $OPT_{D\text{-}MVDPC}(D) < (1-\varepsilon)(|V_D| - 1)$, then $OPT_{U\text{-}MVDPC}(G) < (1-\varepsilon')(|V_G| - 1)$.

Proof: We first explain a reduction from a directed graph D to an undirected bipartite graph G. Let G' be the copy of D. Each vertex v in G' is replaced with four vertices v_{in}, $v_{m\text{-}in}$, $v_{m\text{-}out}$, and v_{out} such that $N_G(v_{in}) = N^-_{G'}(v) \cup \{v_{m\text{-}in}\}$, $N_G(v_{m\text{-}in}) = \{v_{in}, v_{m\text{-}out}\}$, $N_G(v_{m\text{-}out}) = \{v_{m\text{-}in}, v_{out}\}$, and $N_G(v_{out}) = N^+_{G'}(v) \cup \{v_{m\text{-}out}\}$, respectively. Then, we replace each directed edge with an undirected edge. We denote the obtained graph by G. Figure 8

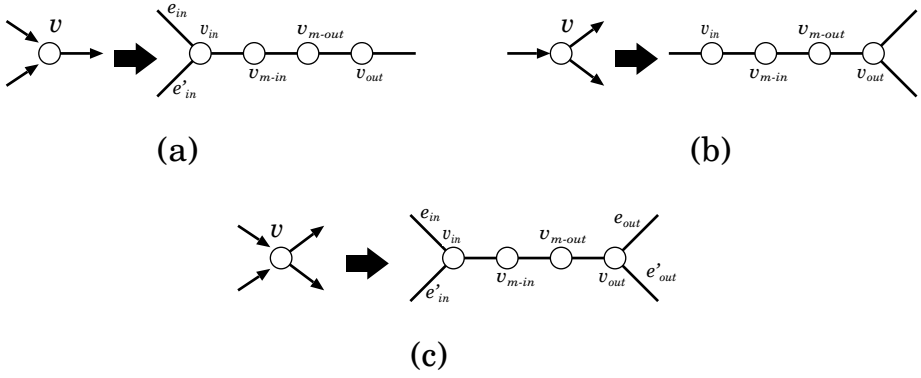

Figure 8: A replacement from a vertex v in D into v_{in}, $v_{m\text{-}in}$, $v_{m\text{-}out}$, and v_{out} in G. (a) A vertex of $d_D^-(v) = 2$ and $d_D^+(v) = 1$. (b) A vertex of $d_D^-(v) = 1$ and $d_D^+(v) = 2$. (c) A vertex of $d_D^-(v) = 2$ and $d_D^+(v) = 2$.

illustrates this transformation. We call the three edges between v_{in} and v_{out} *intermediate* edges and the path of length three the *vertex gadget* of v. From the definition of the transformation, we observe that $|V_G| = 4\,|V_D|$ and that G is a bipartite graph with the bipartition (X, Y), where $X = \{v_{in}, v_{m\text{-}out} \mid v \in V_D\}$ and $Y = \{v_{out}, v_{m\text{-}in} \mid v \in V_D\}$, and hence $|X| = |Y|$ holds. The graph G is constructed in polynomial time.

Now we assume that $\mathrm{OPT}_{\text{D-MVDPC}}(D) = |V_D| - 1$. Then D has a Hamiltonian path. For every vertex v on the Hamiltonian path, we replace v with the path consisting of v_{in}, $v_{m\text{-}in}$, $v_{m\text{-}out}$, and v_{out} in G, then the obtained path is also Hamiltonian in G. Hence, $\mathrm{OPT}_{\text{U-MVDPC}}(G) = |V_G| - 1$ holds.

Next we assume that $\mathrm{OPT}_{\text{D-MVDPC}}(D) < (1-\varepsilon)(|V_D| - 1)$. Let $\mathcal{P}_D$ be an optimal vertex-disjoint path cover of D. By replacing every vertex v on each path in $\mathcal{P}_D$ with the path consisting of v_{in}, $v_{m\text{-}in}$, $v_{m\text{-}out}$, and v_{out}, we obtain a path cover $\mathcal{P}_G$ of G. We prove that $\mathcal{P}_G$ is an optimal path cover of G by contradiction. We assume that there is a path cover $\mathcal{Q}_G$ of G with $\mathit{cost}(\mathcal{P}_G) < \mathit{cost}(\mathcal{Q}_G)$. Recall that, for a path cover $\mathcal{P}$, $\mathit{cost}(\mathcal{P}) = \sum_{P \in \mathcal{P}} \mathrm{len}(P)$. For a path cover of G, an edge e in G is *covered* if e is included in a path in the path cover. Otherwise, e is *uncovered*.

If every intermediate edge is covered in $\mathcal{Q}_G$, then we have a contradiction, as follows. First, from the construction of $\mathcal{P}_G$, it is observed that $\mathit{cost}(\mathcal{P}_D) = \mathit{cost}(\mathcal{P}_G) - 3\,|V_D|$. Next, we transform $\mathcal{Q}_G$ into a path cover of D by transforming the three intermediate edges in each vertex gadget into a vertex of D. Since every intermediate edge is covered, we obtain a path cover $\mathcal{Q}_D$ of D from $\mathcal{Q}_G$. Note that $\mathit{cost}(\mathcal{Q}_D) = \mathit{cost}(\mathcal{Q}_G) - 3\,|V_D|$ holds, since there exist $3\,|V_D|$ covered

intermediate edges in $\mathcal{Q}_G$. Now let us compare the costs of $\mathcal{P}_D$ and $\mathcal{Q}_D$.

$$\begin{aligned} cost(\mathcal{P}_D) &= cost(\mathcal{P}_G) - 3\,|V_D| \\ &< cost(\mathcal{Q}_G) - 3\,|V_D| \\ &= cost(\mathcal{Q}_D), \end{aligned}$$

which contradicts to the optimality of $\mathcal{P}_D$.

Now, we assume that one or more intermediate edges are uncovered in $\mathcal{Q}_G$. In this case, we show that $\mathcal{Q}_G$ can be transformed into a path cover with the same cost such that every intermediate edge is covered. Then, by applying the same discussion above to the path cover, we obtain a contradiction.

Let v be a vertex in D such that its vertex gadget in G includes at least one uncovered intermediate edge in $\mathcal{Q}_G$. Let $Q_{in} = \langle p_1, p_2, \ldots, p_x \rangle$ and $Q_{out} = \langle q_1, q_2, \ldots, q_y \rangle$ be the two paths in $\mathcal{Q}_G$ including v_{in} and v_{out}, respectively. We may have $Q_{in} = Q_{out}$. We first focus on the case of $d_D^-(v) = 2$ and $d_D^+(v) = 1$ in D. We denote the edges incident to v_{in} except the edge $(v_{in}, v_{m\text{-}in})$ by e_{in} and e'_{in} (see Figure 8(a)).

Case 1: The vertex v_{in} is an endpoint of Q_{in}.

If Q_{in} includes an edge $(v_{in}, v_{m\text{-}in})$, then the length of Q_{in} is at most two, since the vertex gadget of v includes at least one uncovered edge. We can obtain a new path by connecting Q_{in} and Q_{out}. The obtained path cover includes one more edge than $\mathcal{Q}_G$, which is a contradiction for the optimality of $\mathcal{Q}_G$. Note that an endpoint of Q_{in} and one of Q_{out} are adjacent with an uncovered intermediate edge from the optimality of $\mathcal{Q}_G$.

Now we assume otherwise, that is either e_{in} or e'_{in} is included in Q_{in}. From the optimality of $\mathcal{Q}_G$, it can be observed that $v_{m\text{-}in}$ is an endpoint of Q_{out}. If $Q_{in} \neq Q_{out}$ holds, by connecting the two paths Q_{in} and Q_{out}, we have the new path including $(v_{in}, v_{m\text{-}in})$. Thus we obtain a better path cover than $\mathcal{Q}_G$. This contradicts to the optimality of $\mathcal{Q}_G$. Now we suppose that $Q_{in} = Q_{out}$. Then we replace Q_{in} with $\langle p_2, p_3, \ldots, p_x, p_1 \rangle$, where $p_1 = v_{in}$. See Figure 9(a). Now we obtain a new path cover with the same cost such that all the intermediate edges in the vertex gadget of v are covered.

Case 2: The vertex v_{in} is an internal vertex of Q_{in}.

Case 2-1: The edge $(v_{in}, v_{m\text{-}in})$ is included in Q_{in}.

Since at least one edge in this vertex gadget is uncovered, either $(v_{m\text{-}in}, v_{m\text{-}out})$ or $(v_{m\text{-}out}, v_{out})$ is uncovered. Note that, if both $(v_{m\text{-}in}, v_{m\text{-}out})$ and $(v_{m\text{-}out}, v_{out})$ are uncovered, then it contradicts to the optimality of $\mathcal{Q}_G$. We first consider the case where $(v_{m\text{-}in}, v_{m\text{-}out})$ is uncovered. Then, Q_{out} includes $(v_{m\text{-}out}, v_{out})$ and $v_{m\text{-}out}$ as an endpoint. If $Q_{in} \neq Q_{out}$ holds, then we obtain a new path by connecting Q_{in} and Q_{out}. The obtained path cover includes one more edge than $\mathcal{Q}_G$, which is a contradiction for the optimality of $\mathcal{Q}_G$. Now suppose that $Q_{in} = Q_{out}$ holds. We replace Q_{in} with $\langle p_3, p_4, \ldots, p_x, p_1, p_2 \rangle$, where $p_1 = v_{m\text{-}in}$ and $p_2 = v_{in}$. We also have the same discussion when $(v_{m\text{-}out}, v_{out})$ is uncovered. Thus we have a path cover with

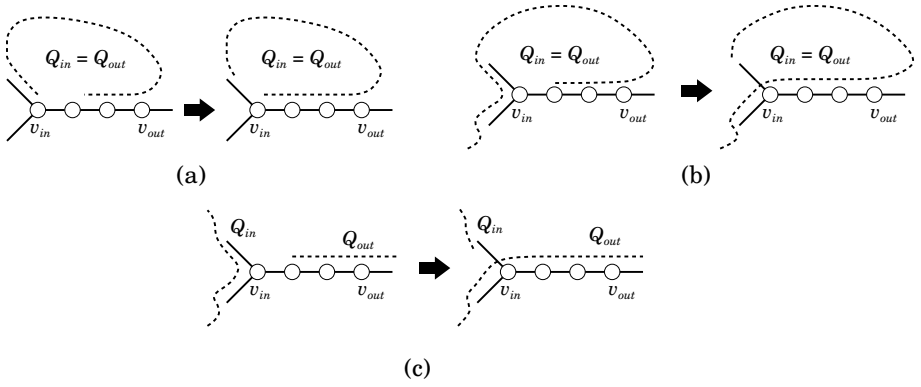

Figure 9: Transformations of paths for Cases 1 and 2. Each dashed line represents a path in a path cover.

the same cost such that all the intermediate edges in the vertex gadget are covered.

Case 2-2: The edge $(v_{in}, v_{m\text{-}in})$ is not included in Q_{in}.

In this case, both e_{in} and e'_{in} are included in Q_{in}, and $v_{m\text{-}in}$ is an endpoint of Q_{out}. We first consider the case of $Q_{in} = Q_{out}$. Let $p_1 = v_{m\text{-}in}$ and $p_i = v_{in}$. Then we replace Q_{in} with the path $\langle p_{i-1}, p_{i-2}, \ldots, p_1, p_i, p_{i+1}, \ldots, p_x \rangle$. See Figure 9(b). For the case of $Q_{in} \neq Q_{out}$, the same replacement can be performed as illustrated in Figure 9(c). Thus we have a path cover with the same cost such that all the intermediate edges in the vertex gadget are covered.

The case for vertices of $d_D^-(v) = 1$ and $d_D^+(v) = 2$ is symmetric to the case above, and hence we omit the details. Now, we next focus on vertices of $d_D^-(v) = 2$ and $d_D^+(v) = 2$.

Let v be a vertex of $d_D^-(v) = 2$ and $d_D^+(v) = 2$ and its vertex gadget includes at least one uncovered intermediate edge for $\mathcal{Q}_G$. Similar to the case before, let e_{in} and e'_{in} be the edges incident to v_{in} except the edge $(v_{in}, v_{m\text{-}in})$, and let e_{out} and e'_{out} be the edges incident to v_{out} except the edge $(v_{m\text{-}out}, v_{out})$ (see Figure 8(c)).

Case A: The vertex v_{in} is an endpoint of Q_{in}.

We have the following two subcases according to v_{out}.

Case A-1: The vertex v_{out} is an endpoint of Q_{out}.

First, assume Q_{in} includes either e_{in} or e'_{in} and Q_{out} includes either e_{out} or e'_{out}. Then, there is a path Q consisting only $v_{m\text{-}in}$ and $v_{m\text{-}out}$ in $\mathcal{Q}_G$. If $Q_{in} \neq Q_{out}$ holds, we can obtain a better path cover than $\mathcal{Q}_G$ by connecting Q_{in}, Q, and Q_{out}, which is a contradiction. Otherwise, $Q_{in} = Q_{out}$, we can obtain a better path cover than $\mathcal{Q}_G$ by connecting Q_{in} and Q, which is also a contradiction. Second, assume Q_{in} includes either e_{in} or e'_{in} and Q_{out} includes $(v_{m\text{-}out}, v_{out})$. Then, $Q_{out} = \langle v_{out}, v_{m\text{-}out}, v_{m\text{-}in} \rangle$ holds. By connecting Q_{in} and

Q_{out}, we have a better path cover than $\mathcal{Q}_G$, which is a contradiction. Third, assume Q_{in} includes $(v_{in}, v_{m\text{-}in})$ and Q_{out} includes either e_{out} or e'_{out}. This case is a symmetry of the above case, and hence we have the same discussion. Finally, assume Q_{in} includes $(v_{in}, v_{m\text{-}in})$ and Q_{out} includes $(v_{m\text{-}out}, v_{out})$. Then, the edge $(v_{m\text{-}out}, v_{m\text{-}out})$ is uncovered. By connecting Q_{in} and Q_{out}, we have a better path cover than $\mathcal{Q}_G$, which is a contradiction.

Case A-2: The vertex v_{out} is an internal vertex of a path in $\mathcal{Q}_G$.

We first assume that Q_{out} includes both e_{out} and e'_{out}. If Q_{in} includes either e_{in} or e'_{in}, then there is a path consisting of only $v_{m\text{-}in}$ and $v_{m\text{-}out}$ in $\mathcal{Q}_G$. By connecting the path and Q_{in}, we have a better path cover than $\mathcal{Q}_G$, which is a contradiction. Otherwise, Q_{in} includes the edge $(v_{in}, v_{m\text{-}in})$, we also have a better path cover by connecting Q_{in} and the path which has a vertex in $N_G(v_{in}) \setminus \{v_{m\text{-}in}\}$ as an endpoint.

We next assume that Q_{out} includes the edge $(v_{m\text{-}out}, v_{out})$. Suppose Q_{in} includes either e_{in} or e'_{in}. Then Q_{out} includes $v_{m\text{-}in}$ as its endpoint. If $Q_{in} \neq Q_{out}$ holds, then we have a better path cover than $\mathcal{Q}_G$ by connecting Q_{in} and Q_{out}. Otherwise, $Q_{in} = Q_{out}$, we replace Q_{in} with $\langle p_2, p_3, \ldots, p_x, p_1 \rangle$, where $p_1 = v_{in}$. Now we obtain a new path cover with the same cost such that all the intermediate edges in the vertex gadget of v are covered. Now, suppose Q_{in} includes the edge $(v_{m\text{-}out}, v_{out})$. By connecting Q_{in} and Q_{out}, we have a better path cover than $\mathcal{Q}_G$, which is a contradiction.

Case B: The vertex v_{in} is an internal vertex of Q_{in}.

We have the following two subcases.

Case B-1: The vertex v_{out} is an endpoint of Q_{out}.

This case is a symmetry of Case A-2. We omit the details.

Case B-2: The vertex v_{out} is an internal vertex of Q_{out}.

First, we assume that Q_{in} includes both e_{in} and e'_{in} and Q_{out} also includes both e_{out} and e'_{out}. Then, there is a path consisting of the two vertices $v_{m\text{-}in}$ and $v_{m\text{-}out}$. Suppose that $Q_{in} = Q_{out}$ holds. Let $p_i = v_{in}$ and $p_j = v_{out}$, $i < j$. We replace Q_{in} with $\langle p_1, p_2, \ldots, p_i, v_{m\text{-}in}, v_{m\text{-}out}, p_j, p_{j-1}, \ldots, p_{i+1} \rangle$ and $\langle p_{j+1}, p_{j+2}, \ldots, p_x \rangle$ (see Figure 10(a)). For the case of $Q_{in} \neq Q_{out}$, we also replace with the same manner (see Figure 10(b)). Thus we have a path cover with the same cost such that all the intermediate edges in the vertex gadget are covered.

Second, we assume that Q_{in} includes both e_{in} and e'_{in} and Q_{out} includes $(v_{m\text{-}out}, v_{out})$. Then, it can be observed that $v_{m\text{-}in}$ is an endpoint of Q_{out}. Suppose $Q_{in} = Q_{out}$ holds, and let $p_i = v_{in}$ and $p_j = v_{out}$, $i < j$. Then we replace Q_{in} with $\langle p_1, p_2, \ldots, p_i, v_{m\text{-}in}, v_{m\text{-}out}, p_j, p_{j-1}, \ldots, p_{i+1} \rangle$ (see Figure 10(c)). For the case of $Q_{in} \neq Q_{out}$, we can replace with the same manner (see Figure 10(d)). Thus we have a path cover with the same cost.

Third, we assume that Q_{in} includes the edge $(v_{in}, v_{m\text{-}in})$ and Q_{out} includes both e_{out} and e'_{out}. This is a symmetry of the above case, and hence we omit the details.

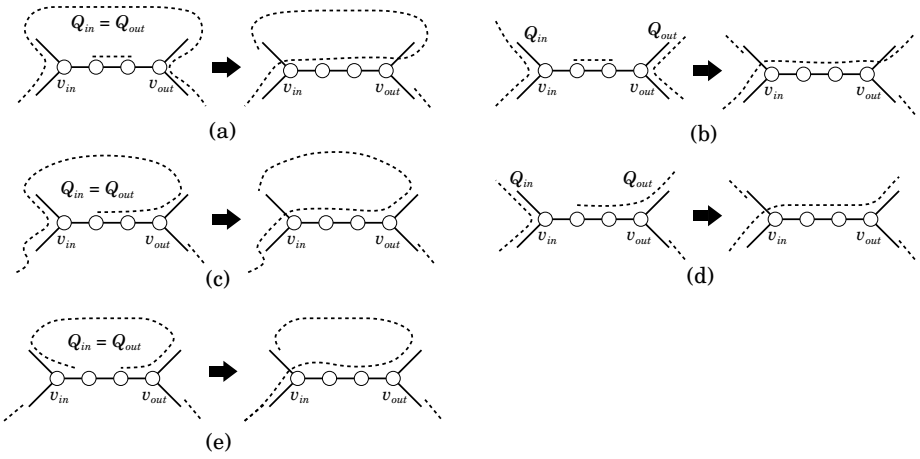

Figure 10: Transformations of paths for Case B.

Finally, we assume that Q_{in} includes the edge $(v_{in}, v_{m\text{-}in})$ and Q_{out} includes the edge $(v_{m\text{-}out}, v_{out})$. Then $v_{m\text{-}in}$ is an endpoint of Q_{in} and $v_{m\text{-}out}$ is an endpoint of Q_{out}. From the optimality of $\mathcal{Q}_G$, $Q_{in} = Q_{out}$ holds. Then we replace the path as illustrated in Figure 10(e). The obtained path cover by the replacement is better than $\mathcal{Q}_G$, which is a contradiction.

From the above case analysis, by transforming paths in $\mathcal{Q}_G$, we can obtain a path cover with the same cost as $\mathcal{Q}_G$ such that all the intermediate edges are included. Therefore, $\mathcal{P}_G$ is an optimal solution.

We have the following equations and inequations:

$$\begin{aligned}
cost(\mathcal{P}_G) &= cost(\mathcal{P}_D) + 3\,|V_D| \\
&< (1-\varepsilon)(|V_D|-1) + 3\,|V_D| \\
&= 4\,|V_D| - \varepsilon(|V_D|-1) - 1 \\
&= |V_G| - \frac{1}{4}\varepsilon(|V_G|-4) - 1 \\
&= (|V_G|-1) - \frac{1}{4}\varepsilon(|V_G|-1) + \frac{3}{4}\varepsilon \\
&= (1-\frac{1}{4}\varepsilon)(|V_G|-1) + \frac{3}{4}\varepsilon \\
&= (1-\frac{1}{8}\varepsilon)(|V_G|-1) - \frac{1}{8}\varepsilon(|V_G|-1) + \frac{3}{4}\varepsilon \\
&< (1-\frac{1}{8}\varepsilon)(|V_G|-1) \qquad (|V_D| \geq 2),
\end{aligned}$$

which completes the proof. □