Swapping Labeled Tokens on Graphs

Katsuhisa Yamanaka[1], Erik D. Demaine[2], Takehiro Ito[3], Jun Kawahara[4], Masashi Kiyomi[5], Yoshio Okamoto[6], Toshiki Saitoh[7], Akira Suzuki[3], Kei Uchizawa[8], and Takeaki Uno[9]

[1] Iwate University, Japan.
yamanaka@cis.iwate-u.ac.jp
[2] Massachusetts Institute of Technology, USA.
edemaine@mit.edu
[3] Tohoku University, Japan.
{takehiro, a.suzuki}@ecei.tohoku.ac.jp
[4] Nara Institute of Science and Technology, Japan.
jkawahara@is.naist.jp
[5] Yokohama City University, Japan.
masashi@yokohama-cu.ac.jp
[6] University of Electro-Communications, Japan.
okamotoy@uec.ac.jp
[7] Kobe University, Japan.
saitoh@eedept.kobe-u.ac.jp
[8] Yamagata University, Japan.
uchizawa@yz.yamagata-u.ac.jp
[9] National Institute of Informatics, Japan.
uno@nii.ac.jp

Abstract. Consider a puzzle consisting of n tokens on an n-vertex graph, where each token has a distinct starting vertex and a distinct target vertex it wants to reach, and the only allowed transformation is to swap the tokens on adjacent vertices. We prove that every such puzzle is solvable in $O(n^2)$ token swaps, and thus focus on the problem of minimizing the number of token swaps to reach the target token placement. We give a polynomial-time 2-approximation algorithm for trees, and using this, obtain a polynomial-time 2α-approximation algorithm for graphs whose tree α-spanners can be computed in polynomial time. Finally, we show that the problem can be solved exactly in polynomial time on complete bipartite graphs.

1 Introduction

A *ladder lottery*, known as "Amidakuji" in Japan, is one of the most popular lotteries. It is often used to assign roles to children in a group, as in the following example. Imagine a teacher of an elementary school wants to assign cleaning duties to two students among four students A, B, C and D. Then, the teacher draws four vertical lines and several horizontal lines between two consecutive vertical lines. (See Fig. 1(a).) The teacher randomly chooses two vertical lines,

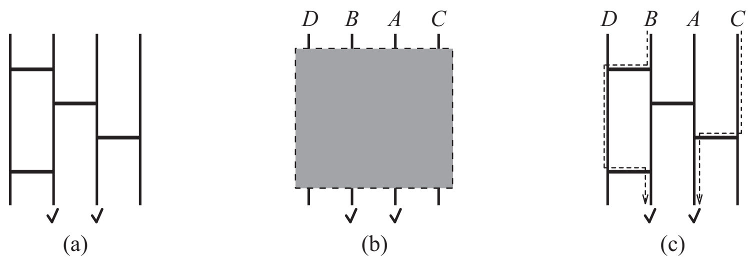


Fig. 1. How to use ladder lottery (Amidakuji) in Japan.

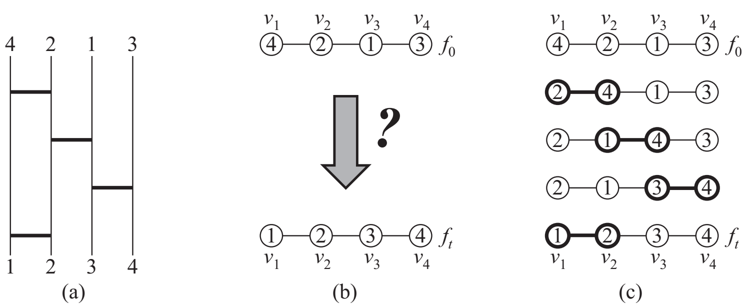


Fig. 2. (a) Ladder lottery of the permutation $(4, 2, 1, 3)$ with the minimum number of bars, (b) its corresponding instance of TOKEN SWAPPING for a path, and (c) a transformation from f_0 to f_t with the minimum number of token swaps.

and draws check marks at their bottom ends. The ladder lottery is hidden, and each student chooses one of the top ends of the vertical lines, as illustrated in Fig. 1(b). Then, the ladder lottery assigns two students to cleaning duties (check marks) by the top-to-bottom route from each student which always turns right or left at each junction of vertical and horizontal lines. (In Fig. 1(c), such a route is drawn as a dotted line.) Therefore, in this example, cleaning duties are assigned to students B and C.

More formally, a ladder lottery can be seen as a model of sorting a particular permutation. Let $\pi = (p_1, p_2, \ldots, p_n)$ be a permutation of integers $1, 2, \ldots, n$. Then, a *ladder lottery* of π is a network with n vertical lines (*lines* for short) and zero or more horizontal lines (*bars* for short) each of which connects two consecutive vertical lines and has a different height from the others. (See Fig. 2(a) as an example.) The top ends of the lines correspond to π, and the bottom ends of the lines correspond to the target permutation $(1, 2, \ldots, n)$. Then, each bar connecting two consecutive lines corresponds to a modification of the current permutation by *swapping* the two numbers on the lines. The sequence of such modifications in a ladder lottery must result in the target permutation $(1, 2, \ldots, n)$.

There are many ladder lotteries that transform the same permutation $\pi = (p_1, p_2, \ldots, p_n)$ into the target one. Thus, one interesting research topic is minimizing the number of bars in a ladder lottery for a given permutation π. This

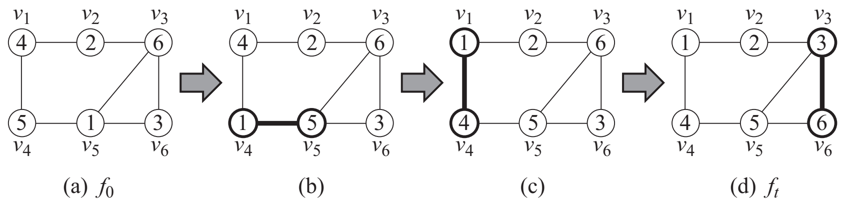


Fig. 3. A sequence of token placements of the same graph.

minimization problem on ladder lottery can be solved by counting the number of "inversions" in π [8, 10], where a pair (p_i, p_j) in π is called an *inversion* in π if $p_i > p_j$ and $i < j$; for example, there are four inversions in the permutation $(4, 2, 1, 3)$, that is, $(4, 2)$, $(4, 1)$, $(4, 3)$ and $(2, 1)$, and hence the ladder lottery in Fig. 2(a) has the minimum number of bars. The bubble sort algorithm sorts π using a number of adjacent swaps equal to the number of inversions in π, and hence gives an optimal ladder lottery of π. In this paper, we study a generalization of this minimization problem from one dimension to general graphs.

1.1 Our problem

Suppose that we are given a connected graph $G = (V, E)$ with $n = |V|$ vertices, with n tokens $1, 2, \ldots, n$ already placed on distinct vertices of G. (Refer to Fig. 3, where the number i written inside each vertex represents the token i.) We wish to transform this initial token placement f_0 into another given target token placement f_t. The transformation must consist of a sequence of token swaps, each defined by an edge of the graph and consisting of swapping the two tokens on the two adjacent vertices of the edge. (See Fig. 3 as an example.) Notice that we need the graph to be connected for there to be a solution.

We will show that such a transformation exists for any two token placements f_0 and f_t. Therefore, we consider the TOKEN SWAPPING problem of minimizing the number of token swaps to transform a given token placement f_0 into another given token placement f_t. Figure 3 illustrates an optimal solution for transforming the token placement f_0 in Fig. 3(a) into the token placement f_t in Fig. 3(d) using a sequence of three token swaps.

As illustrated in Fig. 2, TOKEN SWAPPING on a path is identical to minimizing the number of bars in a ladder lottery. The permutation $\pi = (p_1, p_2, \ldots, p_n)$ in the ladder lottery corresponds to the initial token placement f_0, and the target identity permutation $(1, 2, \ldots, n)$ corresponds to the target token placement f_t where each token i, $1 \leq i \leq n$, is placed on the vertex v_i. Then, the number of bars is identical to the number of token swaps.

1.2 Related work and known results

A ladder lottery appears in a variety of areas in different forms. First, it is strongly related to primitive sorting networks, which are deeply investigated by Knuth [9]. (More precise discussion will be given in Section 2.3.) Second, in algebraic combinatorics, a "reduced decomposition" of a permutation π [11] corresponds to a ladder lottery of π with the minimum number of bars. Third, a ladder lottery of the reverse permutation $(n, n-1, \ldots, 1)$ corresponds to a pseudoline arrangement in discrete geometry [13].

The computational hardness of TOKEN SWAPPING is unknown even for general graphs. However, the problem of minimizing the number of bars in a ladder lottery, and hence TOKEN SWAPPING for paths, can be solved in time $O(n^2)$ by counting the number of inversions in a given permutation [8, 10], or by the application of the bubble sort algorithm. Furthermore, TOKEN SWAPPING can be solved in time $O(n^2)$ for cycles [8] and for complete graphs [3, 8]. Heath and Vergara [7] proposed a polynomial-time 2-approximation algorithm for the square of a path P, where the *square* of P is the graph obtained from P by adding a new edge between two vertices with distance exactly two in P. Therefore, TOKEN SWAPPING has been studied for very limited classes of graphs.

1.3 Our contribution

In this paper, we study the TOKEN SWAPPING problem for some larger classes of graphs, and mainly design three algorithms. We first give a polynomial-time 2-approximation algorithm for trees. Based on the algorithm for trees, we then present a 2α-approximation algorithm for graphs having tree α-spanners. (The definition of tree α-spanners will be given in Section 3.2.) We finally show that the problem is exactly solvable in polynomial time for complete bipartite graphs.

In addition, we give several results and observations which are related to the three main results above. In Section 2.2, we prove that any token placement for a (general) graph G can be transformed into any target token placement by $O(n^2)$ token swaps, where n is the number of vertices in G. We also show that there are instances on paths which require $\Omega(n^2)$ token swaps. In Section 2.3, we discuss the relationship between our problem and sorting networks. We finally note that our lower bound (in Lemma 1) on the minimum number of token swaps holds not only for trees but also for general graphs.

Due to the page limitation, several proofs are omitted.

2 Preliminaries

In this paper, we assume that all graphs are simple and connected. Let $G = (V, E)$ be an undirected and unweighted graph with vertex set V and edge set E. We sometimes denote by $V(G)$ and $E(G)$ the vertex set and edge set of G, respectively. We always denote $n = |V|$.

2.1 Definitions for TOKEN SWAPPING

Suppose that the vertices in a graph $G = (V, E)$ are assigned distinct labels $v_1, v_2, \ldots, v_n$. Let $L = \{1, 2, \ldots, n\}$ be a set of n labeled tokens. Then, a *token* *placement* f of G is a mapping $f\colon V \to L$ such that $f(v_i) \neq f(v_j)$ holds for every two distinct vertices $v_i, v_j \in V$; imagine that tokens are placed on the vertices of G. Since f is a one-to-one correspondence, we can obtain its inverse mapping $f^{-1}\colon L \to V$.

Two token placements f and f' of a graph $G = (V, E)$ are said to be *adjacent* if the following two conditions (a) and (b) hold:

(a) there exists exactly one edge $(v_i, v_j) \in E$ such that $f'(v_i) = f(v_j)$ and $f'(v_j) = f(v_i)$; and

(b) $f'(v_k) = f(v_k)$ for all vertices $v_k \in V \setminus \{v_i, v_j\}$.

In other words, the token placement f' is obtained from f by *swapping* the tokens on two vertices v_i and v_j such that $(v_i, v_j) \in E$. For two token placements f and f' of G, a sequence $\mathcal{S} = \langle f_1, f_2, \ldots, f_h \rangle$ of token placements is called a *swapping sequence* 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 $k = 2, 3, \ldots, h-1$; and

(3) f_{k-1} and f_k are adjacent for every $k = 2, 3, \ldots, h$.

The *length* $\mathsf{len}(\mathcal{S})$ of a swapping sequence $\mathcal{S}$ is defined to be the number of token placements in $\mathcal{S}$ minus one, that is, $\mathsf{len}(\mathcal{S})$ indicates the number of token swaps in $\mathcal{S}$. For example, $\mathsf{len}(\mathcal{S}) = 3$ for the swapping sequence $\mathcal{S}$ in Fig. 3.

Without loss of generality, we always denote by f_t the target token placement of a graph G such that $f_t(v_i) = i$ for all vertices $v_i \in V(G)$. For a token placement f_0 of G, let $\mathsf{OPT}(f_0)$ be the minimum length of a swapping sequence between f_0 and f_t, that is, $\mathsf{OPT}(f_0) = \min\{\mathsf{len}(\mathcal{S}) : \mathcal{S}$ is a swapping sequence between f_0 and $f_t\}$. As we will prove in Theorem 1, there always exists a swapping sequence from any token placement f_0 to the target one f_t, and hence $\mathsf{OPT}(f_0)$ is well-defined. Given a token placement f_0 of a graph G, the TOKEN SWAPPING problem is to compute $\mathsf{OPT}(f_0)$. We denote always by f_0 the *initial* token placement of G.

2.2 Polynomial upper bound on the minimum length

We show the following upper bound for any graph.

Theorem 1. *For any token placement f_0 of a graph G, $\mathsf{OPT}(f_0) = O(n^2)$.*

It is remarkable that there exists an infinite family of instances on paths such that $\mathsf{OPT}(f_0) = \Omega(n^2)$. Recall that TOKEN SWAPPING for paths is equivalent to minimizing the number of bars in a ladder lottery of a given permutation $\pi = (p_1, p_2, \ldots, p_n)$. As we have mentioned in Introduction, the minimum number of bars is equal to the number of inversions in π [8, 10]. Consider the reverse permutation $\pi_r = (n, n-1, \ldots, 1)$. The number of inversions in π_r is $\Omega(n^2)$, and hence $\mathsf{OPT}(f_0) = \Omega(n^2)$ for the corresponding instance of TOKEN SWAPPING.

2.3 Relations to sorting networks

In this subsection, we explain that TOKEN SWAPPING has a relationship to sorting networks in the sense that we can obtain an upper bound on $\mathsf{OPT}(f_0)$ for a given token placement f_0 from a sorting network which sorts f_0.

We first explain that a primitive sorting network [9] gives an upper bound on $\mathsf{OPT}(f_0)$ for TOKEN SWAPPING on paths (*i.e.*, ladder lotteries). A primitive sorting network transforms *any* given permutation into the permutation $(1, 2, \ldots, n)$ by comparators each of which replaces two consecutive elements p_i and p_{i+1} with $\min(p_i, p_{i+1})$ and $\max(p_i, p_{i+1})$, respectively. Therefore, in TOKEN SWAPPING for paths, we can obtain a swapping sequence for a given token placement f_0 by swapping two tokens whose corresponding elements are swapped in the primitive sorting network when f_0 is input as a particular permutation.

We generalize this argument to parallel sorting algorithms on an SIMD machine consisting of several processors with local memory which are connected by a network [1]. For our purpose, an interconnection network is modeled as an undirected graph G with n labeled vertices $v_1, v_2, \ldots, v_n$. Then, a (serial) sorting on G can be seen as a problem to transform a given token placement f_0 of G into the target one f_t by swapping two tokens on the adjacent vertices. In a parallel sorting algorithm for G, we can swap more than one pair of tokens at the same time along a matching M of G; note that each pair of two adjacent tokens in M can be swapped independently. More formally, a parallel sorting algorithm for G with r rounds consists of r prescribed matchings $M_1, M_2, \ldots, M_r$ of G and r prescribed swapping rules $R_1, R_2, \ldots, R_r$; each swapping rule R_i, $1 \le i \le r$, determines whether each pair of two adjacent tokens in M_i is swapped or not by the outcome of comparison of adjacent tokens in M_i. It should be noted that the parallel sorting algorithm must sort *any* given token placement f_0 of G by the prescribed r matchings and their swapping rules. Then, since each matching contains at most $n/2$ edges, the argument similar to primitive sorting networks establishes the following theorem.

Theorem 2. *Suppose that there is a parallel sorting algorithm with r rounds for an interconnection network G. Then, in* TOKEN SWAPPING, $\mathsf{OPT}(f_0) = O(rn)$ *for any token placement f_0 of the graph G.*

For example, it is known that there is a parallel sorting algorithm with $O(\sqrt{n})$ rounds for a $\sqrt{n} \times \sqrt{n}$ mesh [12]. Thus, we have $\mathsf{OPT}(f_0) = O(n^{3/2})$ for TOKEN SWAPPING on such meshes. Similarly, from an $O(\log n (\log \log n)^2)$-round algorithm on hypercubes [4], we obtain $\mathsf{OPT}(f_0) = O(n \log n (\log \log n)^2)$ for TOKEN SWAPPING on hypercubes.

3 Approximation

In this section, we give approximation results.

We first give a lower bound on $\mathsf{OPT}(f_0)$ which holds for any graph. For a graph G and two vertices $v, w \in V(G)$, we denote by $\mathsf{sp}_G(v, w)$ the number of edges in a shortest path on G between v and w. For a token placement f of G, we introduce a *potential function* $\mathsf{p}_G(f)$, as follows:

$$\mathsf{p}_G(f) = \sum_{1 \le i \le n} \mathsf{sp}_G(f^{-1}(i), v_i),$$

(a) f (b) D

Fig. 4. (a) token placement f of a graph, and (b) its conflict graph D.

that is, the sum of shortest path lengths of all tokens from their current positions to the target positions. Notice that $f_t^{-1}(i) = v_i$ for all tokens i, $1 \le i \le n$, and hence $\mathsf{p}_G(f_t) = 0$. Then, we have the following lemma.

Lemma 1. $\mathsf{OPT}(f_0) \ge \frac{1}{2}\mathsf{p}_G(f_0)$ *for any token placement* f_0 *of a graph* G.

3.1 Trees

The main result of this subsection is the following theorem.

Theorem 3. *There is a polynomial-time* 2*-approximation algorithm for* TOKEN SWAPPING *on trees.*

As a proof of Theorem 3, we give a polynomial-time algorithm which actually finds a swapping sequence $\mathcal{S}$ between two token placements f_0 and f_t of a tree T such that

$$\mathsf{len}(\mathcal{S}) \le \sum_{1 \le i \le n} \mathsf{sp}_T(f_0^{-1}(i), v_i) = \mathsf{p}_T(f_0). \tag{1}$$

Then, Lemma 1 implies that $\mathsf{len}(\mathcal{S}) \le 2 \cdot \mathsf{OPT}(f_0)$, as required.

Conflict graph.

To give our algorithm, we introduce a digraph $D = (V_D, E_D)$ for a token placement f of a graph G (which is not necessarily a tree), called the *conflict graph* for f, as follows:

- $V_D = \{v_i \in V(G) : f(v_i) \ne f_t(v_i)\}$; and
- there is an arc (v_i, v_j) from v_i to v_j if and only if $f(v_i) = f_t(v_j) = j$.

Therefore, each token $f(v_i)$ on a vertex $v_i \in V_D$ needs to be moved to the vertex $v_j \in V_D$ such that $(v_i, v_j) \in E_D$. (See Fig. 4 as an example.)

Lemma 2. *Let* D *be the conflict graph for a token placement* f *of a graph* G*. Then, every component in* D *is a directed cycle.*

Algorithm for trees.

We now give our algorithm for trees. For two vertices u and v of a tree T, we denote by $P(u, v)$ a unique path in T between u and v. Let D be the conflict graph for an initial token placement f_0 of T, and let $C = (w_1, w_2, \ldots, w_q)$ be an arbitrary directed cycle in D where $w_q = w_1$. Let $\ell_k = f_0(w_k)$ for each k, $1 \le k \le q-1$; then $f_t(w_{k+1}) = \ell_k$. Our algorithm moves the tokens $\ell_1, \ell_2, \ldots, \ell_{q-1}$ on the vertices in C to their target positions along the unique paths. More formally,

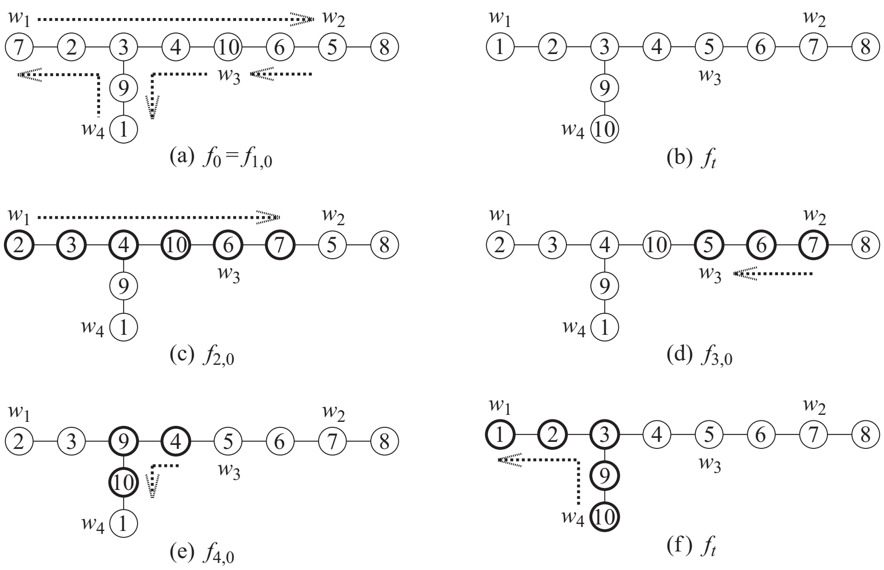


Fig. 5. (a) Initial token placement f_0 of a tree and (b) target one f_t, where a directed cycle $C = (w_1, w_2, w_3, w_4, w_1)$ in the conflict graph D for f_0 is depicted by dotted arrows. (c), (d) and (e) indicate the applications of Step (1) to the tokens $\ell_1 = 7$, $\ell_2 = 5$ and $\ell_3 = 10$, respectively. (f) indicates the application of Step (2) to the token $\ell_4 = 1$.

we construct a swapping sub-sequence $\mathcal{S}_C$ for C, as follows; let $f_{1,0} = f_0$ as the initialization. (See also Fig. 5 as an example.)

(1) At the k-th step of the algorithm, $1 \le k \le q-2$, we focus on the token ℓ_k ($= f_0(w_k)$) which is currently placed on the vertex $f_{k,0}^{-1}(\ell_k)$, and move it to the vertex in the path $P(f_{k,0}^{-1}(\ell_k), f_{k,0}^{-1}(\ell_{k+1}))$ which is adjacent to the vertex $f_{k,0}^{-1}(\ell_{k+1})$. Let $f_{k+1,0}$ be the resulting token placement of T.

(2) At the $(q-1)$-st step of the algorithm, we move the token ℓ_{q-1} ($= f_0(w_{q-1})$) from the vertex $f_{q-1,0}^{-1}(\ell_{q-1})$ to the vertex w_q ($= w_1$).

Then, we have the following lemma.

Lemma 3. *For the swapping sub-sequence $\mathcal{S}_C$, the following (a) and (b) hold:*

(*a*) $\mathsf{len}(\mathcal{S}_C) \le \sum_{1 \le k \le q-1} \mathsf{sp}_T(w_k, w_{k+1})$; *and*

(*b*) *the token placement f of T obtained by $\mathcal{S}_C$ satisfies*

$$f(v_i) = \begin{cases} f_t(v_i) & \text{if } v_i \text{ in } C; \\ f_0(v_i) & \text{otherwise,} \end{cases}$$

for each vertex $v_i \in V(T)$.

It should be noted that Lemma 3(b) ensures that we can choose directed cycles in D in an arbitrary order. Therefore, by repeatedly constructing swapping sub-sequences for all directed cycles in D (in an arbitrary order), we eventually obtain the target token placement f_t of T. Furthermore, notice that $f_0^{-1}(\ell_k) = w_k$ for each k, $1 \leq k \leq q-1$, and hence Lemma 3(a) implies that Eq. (1) holds.

This completes the proof of Theorem 3. $\square$

3.2 General graphs

We now give an approximation algorithm for general graphs by combining our algorithm in Section 3.1 with the notion of "tree spanner" of a graph.

A *tree α-spanner* T of an unweighted graph $G = (V, E)$ is a spanning tree of G such that $\mathsf{sp}_T(v, w) \leq \alpha \cdot \mathsf{sp}_G(v, w)$ for every pair of vertices $v, w \in V$ [2]. Then, we have the following theorem.

Theorem 4. *Suppose that a graph G has a tree α-spanner, and it can be computed in polynomial time. Then, there is a polynomial-time 2α-approximation algorithm for* TOKEN SWAPPING *on G.*

Theorem 4 requires to find a tree α-spanner of a graph G in polynomial time. However, Cai and Corneil [2] proved that deciding whether an unweighted graph G has a tree α-spanner is NP-complete for any fixed $\alpha \geq 4$, while it can be solved in polynomial time for $\alpha \leq 2$. Therefore, several approximation and FPT algorithms have been studied extensively. For example, Emek and Peleg [6] proposed a polynomial-time $O(\log n)$-approximation algorithm on any unweighted graph for the problem of finding the minimum value of α. Dragan and Köhler [5] gave approximation results for some graph classes. (For details, see their papers and the references therein.)

4 Complete Bipartite Graphs

The main result of this section is the following theorem.

Theorem 5. TOKEN SWAPPING *can be solved exactly in polynomial time for complete bipartite graphs.*

Let G be a complete bipartite graph, and let X and Y be the bipartition of the vertex set $V(G)$. We again construct the conflict graph $D = (V_D, E_D)$ for a token placement f of G. Then, we call a directed cycle in D an *XY-cycle* if it contains at least one vertex in X and at least one vertex in Y. Similarly, a directed cycle in D is called an *X-cycle* (or a *Y-cycle*) if it consists only of vertices in X (resp., only of vertices in Y). Let $c_{XY}(f)$, $c_X(f)$ and $c_Y(f)$ be the numbers of XY-cycles, X-cycles and Y-cycles in D, respectively. Let $c_0(f)$ be the number of vertices in $V(G)$ that are not in D, that is, $c_0(f) = |V(G) \setminus V_D|$. Then, we introduce the following value $s(f)$ for f:

$$s(f) = c_{XY}(f) + c_X(f) + c_Y(f) + c_0(f) - 2 \cdot \max\bigl\{c_X(f), c_Y(f)\bigr\}. \qquad (2)$$

For a token placement f of a complete bipartite graph G, let $\mathsf{q}(f) = n - s(f)$. Then, we have the following formula for $\mathsf{OPT}(f_0)$.

Lemma 4. $\mathsf{OPT}(f_0) = \mathsf{q}(f_0)$.

Lemma 4 implies that $\mathsf{OPT}(f_0)$ can be computed in polynomial time for a complete bipartite graph G. Therefore, in the remainder of this section, we prove Lemma 4 as a proof of Theorem 5.

4.1 Upper bound

We first prove $\mathsf{OPT}(f_0) \leq \mathsf{q}(f_0)$ by induction on $\mathsf{q}(f_0)$. Our proof yields an actual swapping sequence $\mathcal{S}$ between two token placements f_0 and f_t of a complete bipartite graph G such that $\mathsf{len}(\mathcal{S}) = \mathsf{q}(f_0)$.

Base case.

Let f_0 be an initial token placement of G such that $\mathsf{q}(f_0) = 0$. Then, we claim that $f_0 = f_t$. Recall that $c_{XY}(f_0)$, $c_X(f_0)$ and $c_Y(f_0)$ denote the numbers of directed *cycles* in D, while $c_0(f_0)$ denotes the number of *vertices* in G that are not contained in D. Since each directed cycle in D contains at least two vertices of G, we have $c_0(f_0) = |V(G) \setminus V_D| \leq n - 2 \cdot \big(c_{XY}(f_0) + c_X(f_0) + c_Y(f_0)\big)$. Therefore, by Eq. (2) we have

$$s(f_0) \leq n - \big(c_{XY}(f_0) + c_X(f_0) + c_Y(f_0)\big) - 2 \cdot \max\big\{c_X(f_0), c_Y(f_0)\big\}.$$

Since $c_{XY}(f_0)$, $c_X(f_0)$ and $c_Y(f_0)$ are all non-negative integers, we thus have $s(f_0) \leq n$. Furthermore, $s(f_0) = n$ holds if and only if $c_{XY}(f_0) = c_X(f_0) = c_Y(f_0) = 0$, that is, the conflict graph D has no vertex. Therefore, if $\mathsf{q}(f_0) = n - s(f_0) = 0$ and hence $s(f_0) = n$ holds, then we have $f_0 = f_t$ as claimed. We thus have $\mathsf{OPT}(f_0) = 0 = \mathsf{q}(f_0)$.

Inductive step.

Suppose that $\mathsf{OPT}(f'_0) \leq \mathsf{q}(f'_0)$ holds for any token placement f'_0 of G such that $\mathsf{q}(f'_0) = k$. Let f_0 be an initial token placement of G such that $\mathsf{q}(f_0) = k+1$. Then, we prove that $\mathsf{OPT}(f_0) \leq \mathsf{q}(f_0) = k + 1$ holds.

We may assume without loss of generality that $c_X(f_0) \geq c_Y(f_0)$. We first choose one directed cycle C from the conflict graph D for f_0 in the following manner:

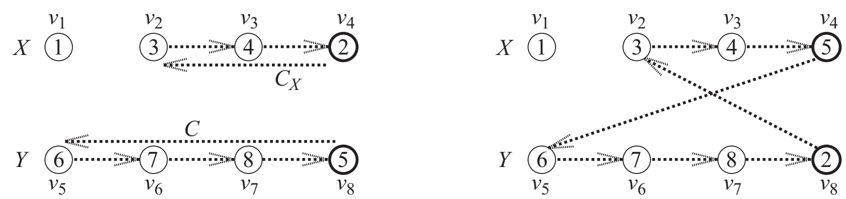
(A) if $c_{XY}(f_0) \geq 1$, then choose any XY-cycle C in D;
(B) if $c_{XY}(f_0) = 0$ and $c_Y(f_0) \geq 1$, then choose any Y-cycle C in D; and
(C) otherwise choose any X-cycle C in D.

It should be noted that at least one of $c_{XY}(f_0)$, $c_X(f_0)$ and $c_Y(f_0)$ is non-zero because $\mathsf{q}(f_0) = n - s(f_0) \neq 0$. Therefore, we can always choose one directed cycle C from D according to the three cases (A)–(C) above.

We then swap some particular pair of tokens according to the chosen directed cycle C. We will show that the resulting token placement f'_0 of G satisfies $\mathsf{q}(f'_0) = k$. Then, by applying the induction hypothesis to f'_0, we have

$$\mathsf{OPT}(f_0) \leq 1 + \mathsf{OPT}(f'_0) \leq 1 + \mathsf{q}(f'_0) = 1 + k = \mathsf{q}(f_0).$$

Due to the page limitation, we here prove only Case (b), that is, C is a Y-cycle; the remaining cases can be proved similarly.



(a) f_0 (b) f_0'

Fig. 6. Example of Case (B).

In this case, by the choice of directed cycles from D, we have $c_{XY}(f_0) = 0$. Furthermore, since $c_X(f_0) \geq c_Y(f_0)$, we have $c_X(f_0) \geq 1$ and hence the conflict graph D for f_0 contains at least one X-cycle C_X. Figure 6(a) illustrates an example; in the figure, for the sake of simplicity, we omit all the edges in $E(G)$ and depict the arcs in the conflict graph by dotted arrows.

We arbitrarily pick one vertex in C and one vertex in C_X, and swap the two tokens on them. (See Fig. 6(b).) Then, the resulting token placement f_0' of G satisfies $c_{XY}(f_0') = c_{XY}(f_0) + 1$ $(= 1)$; $c_X(f_0') = c_X(f_0) - 1$ (≥ 0); $c_Y(f_0') = c_Y(f_0) - 1$ (≥ 0); and $c_0(f_0') = c_0(f_0)$. Therefore, by Eq. (2) we have

$$s(f_0') = \big(c_{XY}(f_0) + 1\big) + \big(c_X(f_0) - 1\big) + \big(c_Y(f_0) - 1\big) \\ + c_0(f_0) - 2 \cdot \max\big\{c_X(f_0) - 1, c_Y(f_0) - 1\big\} = s(f_0) + 1.$$

We thus have $\mathsf{q}(f_0') = n - s(f_0') = n - \big(s(f_0) + 1\big) = \mathsf{q}(f_0) - 1 = k$ for Case (b).

In this way, we can verify that $\mathsf{OPT}(f_0) \leq \mathsf{q}(f_0)$ holds.

4.2 Lower bound

We then prove $\mathsf{OPT}(f_0) \geq \mathsf{q}(f_0)$. Since $\mathsf{q}(f_t) = 0$, it suffices to show that one token swap can decrease the value $\mathsf{q}(f_0)$ by at most one. More formally, we have the following lemma, which completes the proof of Lemma 4.

Lemma 5. $\big|\mathsf{q}(f') - \mathsf{q}(f)\big| \leq 1$ *holds for any two adjacent token placements* f *and* f' *of a complete bipartite graph* G.

5 Concluding Remark

In this paper, we investigated algorithms for the TOKEN SWAPPING problem on some non-trivial graph classes. We note that the algorithm for trees runs in $O(n^2)$ time, because each step moves the token ℓ_k along the unique path of $O(n)$ length in the tree. A swapping sequence $\mathcal{S}$ can be represented by outputting the edges used for the token swaps in $\mathcal{S}$. Therefore, the algorithm can return an actual swapping sequence for a given token placement f_0 in $O(n^2)$ time, while there are instances on paths such that $\mathsf{OPT}(f_0) = \Omega(n^2)$ as we have discussed in Section 2.2. Therefore, it seems difficult to improve the time complexity $O(n^2)$ of the algorithm if we wish to output an actual swapping sequence explicitly.

Acknowledgment

We are grateful to Takashi Horiyama, Shin-ichi Nakano and Ryuhei Uehara for their comments on related work and fruitful discussions with them. This work is partially supported by MEXT/JSPS KAKENHI, including the ELC project. (Grant Numbers 24.3660, 24106010, 24700130, 25106502, 25106504, 25330003.)

References

1. Bitton, D., DeWitt, D.J., Hsaio, D.K., Menon, J.: A taxonomy of parallel sorting. ACM Computing Surveys 16, pp. 287–318 (1984)
2. Cai, L., Corneil, D.G.: Tree spanners. SIAM J. Discrete Mathematics 8, pp. 359–387 (1995)
3. Cayley, A.: Note on the theory of permutations. Philosophical Magazine 34, pp. 527–529 (1849)
4. Cypher, R., Plaxton, C.G.: Deterministic sorting in nearly logarithmic time on the hypercube and related computers. J. Computer and System Sciences 47, pp. 501–548 (1993)
5. Dragan F.F., Köhler, E.: An approximation algorithm for the tree t-spanner problem on unweighted graphs via generalized chordal graphs. Proc. APPROX-RANDOM 2011, LNCS 6845, pp. 171–183 (2011)
6. Emek, Y., Peleg, D.: Approximating minimum max-stretch spanning trees on unweighted graphs. SIAM J. Computing 38, pp. 1761–1781 (2008)
7. Heath, L.S., Vergara, J.P.C.: Sorting by short swaps. J. Computational Biology 10, pp. 775–789 (2003)
8. Jerrum, M.R.: The complexity of finding minimum-length generator sequence. Theoretical Computer Science 36, pp. 265–289 (1985)
9. Knuth, D.E.: Axioms and Hulls. LNCS 606, Springer-Verlag (1992)
10. Knuth, D.E.: The Art of Computer Programming, vol. 3, 2nd edition. Addison-Wesley (1998)
11. Manivel, L.: Symmetric Functions, Schubert Polynomials and Degeneracy Loci. American Mathematical Society (2001)
12. Thompson, C.D., Kung, H.T.: Sorting on a mesh-connected parallel computer. Communications ACM 20, pp. 263–271 (1977)
13. Yamanaka, K., Nakano, S., Matsui, Y., Uehara, R., Nakada, K.: Efficient enumeration of all ladder lotteries and its application. Theoretical Computer Science 411, pp. 1714–1722 (2010)