Swapping Mixed-Up Beers to Keep Them Cool

Davide Bilò
Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Italy

Maurizio Fiusco
Department of Enterprise Engineering, University of Rome "Tor Vergata", Italy

Luciano Gualà
Department of Enterprise Engineering, University of Rome "Tor Vergata", Italy

Stefano Leucci
Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Italy

Abstract

There was a mix-up in Escher's bar and n customers sitting at the same table have each received a beer ordered by somebody else in the party. The drinks can be rearranged by swapping them in pairs, but the eccentric table shape only allows drinks to be exchanged between people sitting on opposite sides of the table. We study the problem of finding the minimum number of swaps needed so that each customer receives its desired beer before it gets warm.

Formally, we consider the Colored Token Swapping problem on complete bipartite graphs. This problem is known to be solvable in polynomial time when all ordered drinks are different [Yamanaka et al., FUN 2014], but no results are known for the more general case in which multiple people in the party can order the same beer. We prove that Colored Token Swapping on complete bipartite graphs is NP-hard and that it is fixed-parameter tractable when parameterized by the number of distinct types of beer served by the bar.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Mathematics of computing → Combinatoric problems; Theory of computation → Problems, reductions and completeness

Keywords and phrases Colored Token Swapping, Complete Bipartite Graphs, Labeled Token Swapping, FPT Algorithms, NP-Hardness

Digital Object Identifier 10.4230/LIPIcs.FUN.2024.5

1 Introduction

A party of n theoretical computer scientists walks into Escher's bar, which is renowned for its tasty beers and its tables with eccentric geometric shapes. One of their papers just got accepted to an important conference[1] and they want to celebrate with a toast. They sit at a table shaped like the one in Figure 1 (a), and each of them orders one of the k beers from the bar's selection. The waiter brings the order to the table but, unfortunately, it delivers the drinks to the patrons in a mixed order. The scientists decide to rectify the situation by swapping pairs of drinks, so that everybody eventually ends up with their beer of choice. However the table's shape prevents some of these swaps: two people sitting on the same side of the table cannot easily swap their drinks, while people sitting on opposite sides of the table can do that by sliding their beers across. To avoid the disastrous waste of beer that would

[1] The reader might have already guessed the name of the conference. It suffices to say that it is held in a beautiful island, and that it is known for its entertaining talks.

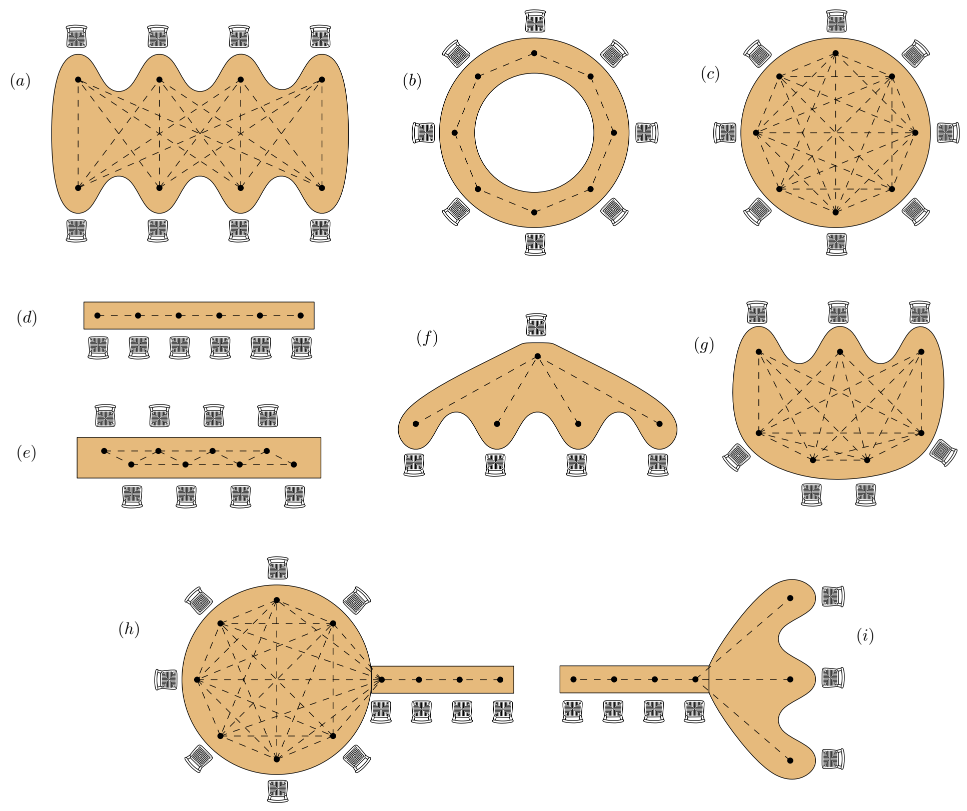

Figure 1 The interior of Escher's bar with its eccentric tables. (a) a complete bipartite table; (b) a cycle table; (c) a clique table; (d) a path table; (e) a square of a path table; (f) a star table; (g) a complete split table; (h) a lollipop table; (i) a broom table. Notice that (h) and (i) are actually obtained by joining two different tables, as it might happen when large parties need to be accommodated.

result if two pints were to crash, they adopt the safe strategy of only performing one such swap at a time. As nobody likes their beer warm (see, e.g., [5]), the above rearrangement process should be completed as quickly as possible.

This can be formalized as an instance of the Colored Token Swapping (CTS) problem on complete bipartite graphs. In this problem we are given an integer $k \in \{1, \ldots, n\}$, and an n-vertex graph $G = (V, E)$ (whose vertices represent scientists, and whose edges represent the swaps allowed by the table shape), in which each vertex v has an associated *color* $c(v) \in \{1, \ldots, k\}$, and hosts a token of color $t(v) \in \{1, \ldots, k\}$ (the colors represent the beers in the bar's selection). A move (or swap) consists in selecting an edge $(u, v) \in E$ and swapping the tokens placed on u and v. The goal is that of finding a shortest sequence of swaps needed to place each token on a vertex of the same color.

The CTS problem is known to be NP-Hard for any $k \geq 3$ even for planar (non-complete) bipartite graphs with maximum degree 3, while it is solvable in polynomial time when $k = 2$ [15]. If one considers special classes of graphs, the problem has been shown to be polynomial time solvable on stars and paths [3]. On cliques, CTS remains NP-hard and, assuming the exponential time hypothesis (ETH) [7], it does not admit any $2^{o(n)}$-time algorithm [3]. On the positive side, it is fixed-parameter tractable when parameterized by k [15].

The CTS problem is a generalization of the Labeled Token Swapping (TS) problem, which corresponds to the case in which $k = n$ and there is exactly one vertex and one token of each color (i.e., scientists order distinct drinks). This special case has received extensive

Table 1 State of the art of the CTS problem and of the TS problem (which corresponds to the case $k = n$ when there is exactly one vertex and one token of each color). References to results in this paper are in bold. Δ denotes the maximum degree of the input graph while results marked with "(ETH)" hold unless the exponential time hypothesis fails.

Graph class	**# of colors (k)**	**Status**	**Ref.**
General	$k = 2$	Solvable in polynomial time	[15]
General	Part of the input	Solvable in time $2^{O(n \log n)}$	[11]
General	Part of the input	4-approximable	[11]
Trees	Part of the input	2-approximable	[11]
Stars	Part of the input	Solvable in polynomial time	[3]
Paths	Part of the input	Solvable in polynomial time	[3]
Planar bipartite, $\Delta = 3$	Any fixed $k \geq 3$	NP-Hard	[15]
Cliques	Part of the input	Solvable in time $O(f(k) \cdot \text{poly}(n))$	[15]
Cliques	Part of the input	NP-Hard	[3]
Cliques	Part of the input	No $2^{o(n)}$-time algorithm (ETH)	[3]
Complete bipartite	Part of the input	Solvable in time $O(f(k) + n)$	**Thm. 14**
Complete bipartite	Part of the input	NP-Hard	**Thm. 17**
Complete bipartite	Part of the input	No $2^{o(n)}$-time algorithm (ETH)	**Thm. 18**
$\Delta = O(1)$	$k = n$ (TS)	APX-Hard	[11]
Trees	$k = n$ (TS)	NP-Hard	[1]
Cliques	$k = n$ (TS)	Solvable in polynomial time	[4]
Cycles	$k = n$ (TS)	Solvable in polynomial time	[8]
Brooms	$k = n$ (TS)	Solvable in polynomial time	[13, 10]
Lollipops	$k = n$ (TS)	Solvable in polynomial time	[10]
Square of paths	$k = n$ (TS)	Solvable in polynomial time	[6]
Complete split	$k = n$ (TS)	Solvable in polynomial time	[17]
Complete bipartite	$k = n$ (TS)	Solvable in polynomial time	[16]

attention in the literature and it is known to be APX-Hard on bounded-degree graphs [11] and NP-Hard on trees [1]. Similarly to the colored case, TS has been considered on special classes of graphs and, besides those mentioned above, polynomial-time algorithms are also known for cliques [4], cycles [8], brooms [13, 10], lollipop graphs [10], squares of paths [6], and complete split graphs [17] (see Figure 1 for the corresponding table shapes, and Table 1 for a summary).

TS is also known to be polynomial-time solvable on complete bipartite graphs [16], where the complexity status of the more general CTS is still unknown. This is exactly the focus of this work, where we show that CTS is NP-Hard for general k, while it can be solved in time $O(\varphi(k) + n)$ for a suitable function $\varphi(\cdot)$ depending only on k, i.e., it is fixed-parameter tractable w.r.t. k.[2] Moreover, we show that no $2^{o(n)}$-time algorithm exists unless the ETH fails [7].

Other related work. The approximation of the CTS problem has been also studied, and a 4- and 2-approximation algorithms have been designed for general graphs and trees, respectively [11]. The same paper also shows that the problem can be solved in time $2^{O(n \log n)}$. Stars

[2] We assume that the algorithm does not have to check the validity of the input instance which, for CTS instances, can be done in time linear in the size of G.

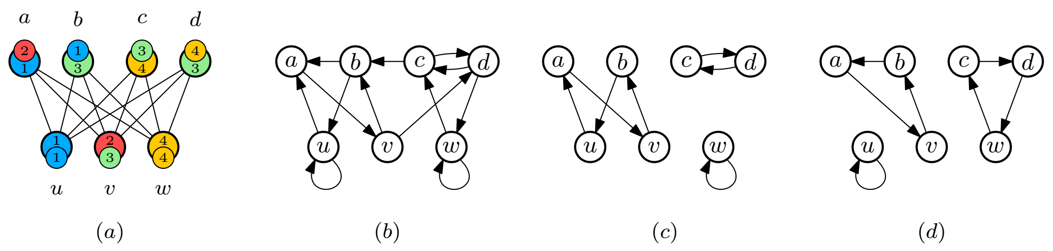

Figure 2 An instance of CTS on a complete bipartite graph with $X = \{a, b, c, d\}$ and $Y = \{u, v, w\}$ is shown in (a). The corresponding moves graph is shown in (b) while (c) shows a possible swapping plan $\mathcal{P}$ with one self-loop, one X-cycle, no Y-cycles, and one XY-cycle, hence $f(\mathcal{P}) = 1$. A better swapping plan $\mathcal{P}'$ with $f(\mathcal{P}') = 3$ is shown in (d). $\mathcal{P}'$ consists of one self-loop, no X- or Y-cycles, and two XY-cycles.

and paths can be solved in polynomial time even for the weighted version of CTS, where each color has an associated weight and a the cost of a swap is given by the sum of the color-weights of the two tokens involved [2]. A generalization of the CTS problem, called *subset* token swapping, where each token has a subset of destination vertices it can be placed on, has been studied in [3]. Finally, a *parallel* version of both CTS and TS, where tokens can be simultaneously swapped over a matching in a single round, and the objective is to minimize the number of rounds (see for example [1] and the references therein).

Structure of the paper. In Section 2 we discuss a useful connection between solutions of the CTS problem on complete bipartite graphs and cycle-covers of a suitable auxiliary graph. As a warm-up, Section 3 is devoted to the special case $k = 3$, where we show how to solve the problem in polynomial time. We do not concern ourselves with finer grained complexity considerations since a $O(n)$-time algorithm for this case follows from our more general $O(\varphi(k) + n)$-time algorithm for the general case, which is described in Section 4. Finally, in Section 5 we establish the NP-hardness of CTS on complete bipartite graphs and show that no $2^{o(n)}$-time algorithm exists unless the ETH fails.

2 Swapping plans and optimal solutions

In this section we argue that the problem of finding an optimal sequence of swaps can be rethought as the problem of finding a suitable (vertex-disjoint) cycle cover of an auxiliary moves graph.

The *moves graph* associated with an instance of swapping colored tokens on bipartite graphs is a directed graph M with vertex set V that contains edge (u, v) iff $t(u) = c(v)$, where u and v are not necessarily distinct (see Figure 2 (a) and Figure 2 (b) for an example).

A *swapping plan* is a feasible assignment of tokens to vertices of the same color. Formally, a swapping plan is a collection $\mathcal{P} = \{C_1, \ldots, C_h\}$ of vertex-disjoint cycles in M such that each vertex is part of exactly one cycle. A *partial swapping plan* is a swapping plan for a vertex-induced subgraph of M. We can show the following useful lemma:

▶ **Lemma 1.** *Let $\mathcal{P}'$ be a partial swapping plan that spans a subset of vertices U of the moves graph M. Then, the subgraph of M induced by all the vertices that are not in U admits a swapping plan $\mathcal{P}''$. Furthermore, $\mathcal{P} = \mathcal{P}' \cup \mathcal{P}''$ is a swapping plan for M.*

Proof. Both M and every cycle in $\mathcal{P}'$ contain as many vertices as tokens of the same color, for every color. Hence, this also holds for the subgraph M' of M induced by the vertices that are not in U. We arbitrarily match each token in M' with a vertex of the same color. If

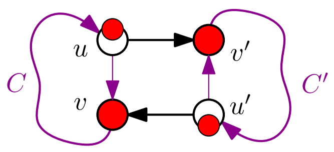

Figure 3 An example showing the merge operation of Lemma 3. The cycle C contains the edge (u, v) while the cycle C' contains the edge (u', v'). The two vertices v and v' have the same color. We can merge C and C' into a single cycle by substituting the edge (u, v) with the edge (u, v') and the edge (u', v') with the edge (u', v).

a token on vertex u is matched with vertex v then $t(u) = c(v)$ and (u, v) is an edge of M'. Since each vertex of M' has exactly one incoming and one outgoing edge in the matching, such a matching induces a swapping plan $\mathcal{P}''$ per M'. Clearly, $\mathcal{P} = \mathcal{P}' \cup \mathcal{P}''$ is a swapping plan for M. ◀

The CTS problem on bipartite graph is essentially that of finding a good swapping plan. Indeed, once a swapping plan $\mathcal{P}$ is fixed, the problem becomes an instance of labelled TS, where a generic token on vertex u needs to be placed on the unique vertex v such that (u, v) is an edge of some cycle in $\mathcal{P}$. The labelled version on complete bipartite graphs can be solved optimally in polynomial time, and it has been shown that the optimal number of swaps is $n - f(\mathcal{P})$, where $f(\mathcal{P})$ is a function that depends on the topologies of the cycles in $\mathcal{P}$ [14]. In the rest of the paper we use X and Y to denote the two sides of the bipartition of G, and we classify each cycle in $\mathcal{P}$ as either a *self-loop*, as an *X-cycle* (resp. a *Y-cycle*) which has length at least 2 and contains only vertices in X (resp. Y), or as an *XY-cycle* which contains at least one vertex in X and one vertex in Y. Defining $\eta_0(\mathcal{P})$, $\eta_X(\mathcal{P})$, $\eta_Y(\mathcal{P})$, $\eta_{XY}(\mathcal{P})$ as the number of self-loops, X-cycles, Y-cycles, and XY-cycles in $\mathcal{P}$, respectively, we have:

$$\begin{aligned} f(\mathcal{P}) &= \eta_0(\mathcal{P}) + \eta_{XY}(\mathcal{P}) + \eta_X(\mathcal{P}) + \eta_Y(\mathcal{P}) - 2\max\{\eta_X(\mathcal{P}), \eta_Y(\mathcal{P})\} \\ &= \eta_0(\mathcal{P}) + \eta_{XY}(\mathcal{P}) - |\eta_X(\mathcal{P}) - \eta_Y(\mathcal{P})|. \end{aligned} \tag{1}$$

As a consequence, the CTS problem on complete bipartite graphs can be equivalently thought of as the problem of finding a swapping plan maximizing $f(\cdot)$. Figure 2 (c)and Figure 2 (d) show two possible swapping plans with different values of $f(\cdot)$ for the instance in Figure 2 (a). We say that a cycle that is either a self-loop or an XY-cycle is *happy*, while X-cycles and Y-cycles are *unhappy*. Roughly speaking, one seeks to maximize the number of happy cycles while keeping the number of unhappy X- and Y-cycles as balanced as possible.

It turns out that, once an optimal swapping plan for the problem has been computed, the corresponding optimal sequence of swaps can be computed in $O(n)$ time, as stated in the following lemma, whose proof is given in Appendix A.

▶ **Lemma 2.** *A swapping plan $\mathcal{P}$ for an instance $\mathcal{I}$ of CTS on complete bipartite graphs can be converted, in time $O(n)$, into a solution for $\mathcal{I}$ consisting of $n - f(\mathcal{P})$ swaps.*

In the following we provide two lemmas that allow us to rearrange cycles of a swapping plan. We start with a *merge* operation which combines two cycles of a swapping plan that share some color into a single cycle. We say that a color c appears in a cycle C if there exists at least one vertex with color c in C. Equivalently, we can say that c appears in C if and only if C contains some token of color c.

▶ **Lemma 3** (Merge operation). *Let $\mathcal{P}$ be a (partial) swapping plan, let C, C' be two distinct cycles in C and let c be a color that appears in both C and C'. Then, there exists a cycle C^* spanning all and only the vertices in C and C' such that $(\mathcal{P} \setminus \{C, C'\}) \cup \{C^*\}$ is a (partial) swapping plan.*

Proof. Let v and v' be two vertices of color c belonging to $\mathcal{C}$ and $\mathcal{C}'$, respectively. And let u (resp. u') be the vertex that immediately precedes v (resp. v') in $\mathcal{C}$ (resp. $\mathcal{C}'$). Notice that it might be $u' = u$ and/or $v' = v$. The cycle $\mathcal{C}^*$ is obtained from $\mathcal{C}$ and $\mathcal{C}'$ by removing the edges (u, v) and (u', v') and by adding the edges (u, v') and (u', v) (see Figure 3). ◀

Next lemma shows that the converse also holds: a cycle contains two vertices/tokens of the same color, it can be split into two shorter cycles.

▶ **Lemma 4** (Split operation). *Let C be a cycle and let u, v be two distinct vertices of C such that $c(u) = c(v)$ or $t(u) = t(v)$. There exist two cycles C_u, C_v whose vertex-sets partition the set of vertices in C and such that u is a vertex of C_u and v is a vertex of C_v.*

Proof. Let $C = \langle u = w_1, w_2, \ldots, w_{j-1}, v = w_j, w_{j+1}, \ldots, w_\ell, u\rangle$.

In the case $c(u) = c(v)$, we must have $t(w_{j-1}) = c(v) = c(u)$ and $t(w_\ell) = c(u) = c(v)$. Then, we choose $C_u = \langle u = w_1, \ldots, w_{j-1}, u\rangle$ and $C_v = \langle v = w_j, w_{j+1}, \ldots, w_\ell, v\rangle$. Notice that it might be $j = 1$, in which case C_u is a self-loop and/or $\ell = j$, in which case C_v is a self loop.

In the case $t(u) = t(v)$, we must have $c(w_2) = t(u) = t(v)$ and $c(w_{j+1}) = t(v) = t(u)$, hence we can choose $C_u = \langle u, w_{j+1}, \ldots, w_\ell, u\rangle$ and $C_v = \langle v, w_2, \ldots, w_{j+1}, v\rangle$. ◀

3 Swapping tokens of 3 colors

In this section, as a warm-up, we focus on the special case of `CTS` on complete bipartite graphs with $k = 3$ possible colors for the tokens/vertices. We introduce the main ideas that will be used also in the next section to solve the more general case of unbounded number of colors.

An instance is *lopsided* if there is some side $Z \in \{X, Y\}$ and some color c such that all the vertices in Z are *monochromatic*, i.e., they induce self-loops in the moves graph, and have all the same color c. The following lemma shows that lopsided instances can be easily solved, hence in the rest of this section we only consider instances that are not lopsided.

▶ **Lemma 5.** *A lopsided instance can be solved in polynomial time.*

Proof. W.l.o.g., we can assume that at least one token is misplaced, which means that not all vertices in M induce self-loops.

Let U be the set of vertices in M that form self-loops. Let $\mathcal{P}' = \{\langle u, u\rangle \mid u \in U\}$ be a partial swapping plan that contains all self-loops of M. As Z is monochromatic, we have $Z \subseteq U$. Let $\mathcal{P}'' = \{C_1, \ldots, C_h\}$ be a swapping plan for the subgraph of M induced by all the vertices of M but those of U. The existence of this swapping plan $\mathcal{P}''$ is guarantee by Lemma 1. Moreover, $\mathcal{P}''$ can be computed in polynomial time as it is a cycle cover of the vertex-induced subgraph of M.

We argue that all cycles in $\mathcal{P}''$ can be merged into a single cycle. By construction, no cycle in $\mathcal{P}''$ can be a self-loop. As a consequence, given any two cycles $C, C' \in \mathcal{P}''$, by the pigeonhole principle there is color that appears in both C and C'. This implies that we can merge the two cycles into a single cycle C^* as proved in Lemma 3. Therefore, we can assume that $\mathcal{P}''$ contains a single cycle.

We divide the proof into two cases according to whether the color c of vertices in Z also appears in the unique cycle of $\mathcal{P}''$ or not.

The first case is when c also appears in the unique cycle of $\mathcal{P}''$, say C'. Let C be a self-loop in $\mathcal{P}'$ such that c appears in C. We define $\mathcal{P}$ as the swapping plan obtained by the union of $\mathcal{P}' \setminus \{C\}$ plus the cycle C^* obtained by merging C with C' (Lemma 3). We argue that $\mathcal{P}$ maximizes the function f defined in Equation (1). By construction, $f(\mathcal{P}) = |U|$. Let $\mathcal{P}^*$ be an optimal swapping plan. Observe that $\eta_0(\mathcal{P}^*) + \eta_{XY}(\mathcal{P}^*) \leq |U|$ as each XY-cycle must contain at least one vertex of Z and thus of U. Therefore, $f(\mathcal{P}^*) \leq |U| - |\eta_X(\mathcal{P}^*) - \eta_Y(\mathcal{P}^*)| \leq |U| = f(\mathcal{P})$.

The second case is when c does not appear in the unique cycle of $\mathcal{P}''$. W.l.o.g., we assume that $Z = X$, as the proof for the case $Z = Y$ is similar. We argue that $\mathcal{P} = \mathcal{P}' \cup \mathcal{P}''$ maximizes the function f defined in Equation (1). By construction $f(\mathcal{P}) = |U| - 1$ as the unique cycle in $\mathcal{P}''$ is unhappy because it spans a subset of vertices in Y. Let $\mathcal{P}^*$ be an optimal swapping plan. As each XY-cycle must contain at least one vertex of Z and thus of U, we have that $\eta_0(\mathcal{P}^*) + \eta_{XY}(\mathcal{P}^*) + \eta_X(\mathcal{P}^*) \leq |U|$, from which we derive $\eta_0(\mathcal{P}^*) + \eta_{XY}(\mathcal{P}^*) \leq |U| - \eta_X(\mathcal{P}^*)$. Furthermore, as no vertex spanned by the cycle in $\mathcal{P}''$ has the same color c of the vertices of Z, we have that $\mathcal{P}^*$ contains unhappy Y-cycles (i.e., those that span vertices of colors different from c). This implies that $\eta_Y(\mathcal{P}^*) \geq 1$. Therefore, $f(\mathcal{P}^*) \leq \eta_0^* - \eta_X(\mathcal{P}^*) - |\eta_X(\mathcal{P}^*) - \eta_Y(\mathcal{P}^*)| \leq \eta_0^* - 1 = f(\mathcal{P})$. ◀

Section 3 allows us to rule out the case in which an instance is lopsided. As the following lemma shows, all the remaining instances have the nice properties that optimal swapping plans consist of happy cycles only.

▶ **Lemma 6.** *Let $\mathcal{I}$ be an instance that is not lopsided. All optimal swapping plans for $\mathcal{I}$ contain only happy cycles.*

Proof. Assume towards a contradiction that some optimal swapping plan $\mathcal{P}^*$ for $\mathcal{I}$ contains an unhappy cycle, and assume further that such a cycle is an X-cycle (w.l.o.g.).

Observe that no unhappy cycle C in $\mathcal{P}^*$ contains any monochromatic vertex v, since otherwise we could increase the number of happy cycles without affecting the number of unhappy (thus increasing $f(\cdot)$) by replacing C with two cycles consisting of (i) a self-loop on v, and (ii) the cycle obtained from C by adding a directed edge (u, w) from the vertex immediately before v to the vertex immediately after v in C, and then deleting v (notice that $t(u) = c(v) = t(v) = c(w)$ and that u and v might coincide).

Then, every unhappy cycle involves at least two colors and any two distinct unhappy cycles can always be merged into a single cycle. This implies that it is impossible for $\mathcal{P}^*$ to contain both an unhappy X-cycle and an unhappy Y-cycle, since they could be merged (see Lemma 3) into a single happy XY-cycle, increasing $f(\cdot)$. Therefore $\mathcal{P}$ contains no unhappy Y-cycle and exactly one unhappy X-cycle C (otherwise all unhappy X-cycles could be merged into a single unhappy cycle, increasing $f(\cdot)$).

We now argue that C can be merged with some happy cycle containing a vertex in Y, thus decreasing the number of unhappy cycles to 0 without affecting the number of happy cycles, which is a contradiction. Since the vertices in C have at least two distinct colors and $\mathcal{I}$ is not lopsided, we can always find some vertex $v \in Y$ such that at least one of $c(v)$ and $t(v)$ coincides with the color c of a vertex u in C. Then color c appears both in C and in the happy cycle C' of $\mathcal{P}^*$ that contains v, which implies that C and C' can be merged (see Lemma 3). ◀

As a consequence of Lemma 6, we can restrict our attention to finding swapping plans with only happy cycles. However, such cycles could potentially be too long. However, it turns out that one can instead consider a relaxed version of the problem where the goal is that of finding a partial swapping plan that maximizes the number of happy cycles, and such version results in short happy cycles. Clearly, the value of an optimal partial swapping plan, i.e., number of its happy cycles, is an upper bound to the value $f(\mathcal{P}^*)$ of an optimal swapping plan $\mathcal{P}^*$. We prove that the converse also holds, and we show how to convert an optimal partial swapping plan into an optimal swapping plan.

We now formalize the above relation between the two problems. With a slight abuse of notation we let $f(\widetilde{\mathcal{P}})$ be the number of happy cycles in a partial swapping plan $\widetilde{\mathcal{P}}$.

▶ **Lemma 7.** *Let $\widetilde{\mathcal{P}}$ be an optimal partial swapping plan for a non-lopsided instance $\mathcal{I}$. We can compute an optimal swapping plan $\mathcal{P}$ for $\mathcal{I}$ with $f(\mathcal{P}) = f(\widetilde{\mathcal{P}})$ in polynomial time.*

Proof. We say that a vertex is uncovered if it does not belong to any of the cycles in $\widetilde{\mathcal{P}}$. We consider the case in which there exists at least one uncovered vertex, since otherwise the claim is trivial. We assume w.l.o.g., that such uncovered vertex is in X.

Observe that $\widetilde{\mathcal{P}}$ contains no monochromatic uncovered vertex, since otherwise we could add a self-loop to $\widetilde{\mathcal{P}}$ increasing $f(\cdot)$. Complete $\widetilde{\mathcal{P}}$ into a swapping plan $\mathcal{P}'$ by partitioning the uncovered vertices in an arbitrary set of additional cycles (Lemma 1 ensures that this is always possible), and notice that the optimality of $\widetilde{\mathcal{P}}$ implies that each such cycle is unhappy.

Since each unhappy cycle contains vertices with at least two different colors any two such cycles can always be merged (see Lemma 3). In particular, any unhappy X-cycle can always be merged with any unhappy Y-cycle in $\mathcal{P}'$ resulting in an happy XY-cycle. However, any such merge would contradict the optimality of $\widetilde{\mathcal{P}}$, which implies that $\mathcal{P}'$ contains no unhappy Y-cycles.

We can then merge all unhappy X-cycles in $\mathcal{P}'$ into a single cycle C. Since $\mathcal{I}$ is not lopsided, C can be further merged with some (happy) cycle containing a suitable vertex from Y by using analogous arguments to the ones employed in the proof of Lemma 6. This results in a swapping plan $\mathcal{P}$ with $f(\mathcal{P}) \geq f(\widetilde{\mathcal{P}})$, which implies $f(\mathcal{P}) = f(\widetilde{\mathcal{P}})$. ◀

The following lemma provides another important key ingredient that allows us to find optimal solutions of our problem instance. In particular, the lemma states that we can focus on partial swapping plans containing only very short happy cycles.

▶ **Lemma 8.** *There exists an optimal partial swapping plan containing only happy cycles having a length of at most* 4.

Proof. In the rest of this proof we name the three distinct colors `red`, `green`, and `blue`. W.l.o.g., we can assume that the optimal partial swapping plan contains only happy cycles as unhappy cycles can be discarded.

We argue that, given any happy cycle C of length 5 or more, there exists another happy cycle C' of length at most 4 that contains only a subset of the vertices of C. The existence of an optimal partial swapping plan with cycles of length at most 4 follows by starting from any optimal partial swapping plan and iteratively replacing any long happy cycle C with a short happy cycle C' that spans only (a subset of) vertices of C, while discarding the vertices that are in C but not in C'.

Given two vertices u, v, we say that they are complementary if $c(u) = t(v)$ and $t(u) = c(v)$. In the rest of the proof we assume that C contains no monochromatic vertices, since otherwise we can choose C' as the self-loop consisting of a single monochromatic vertex from C. We can also assume that there are no two complementary vertices u, v of C on different sides of the bipartition, otherwise we choose $C' = \langle u, v, u \rangle$.

Since C has length of at least 5, there exists one side of the bipartition, say X, that contains 3 vertices of C. If any two of these three vertices have the same color, or host tokens of the same color then, by Lemma 4, we can split C into two cycles such that at least one the two cycles must include a vertex on the opposite side of the bipartition, i.e., it is happy.

As a consequence, we consider the case in which all vertices in X have different colors and host tokens of different colors as well. Let $u \in X$ and $z \in Y$ such that $c(z) = c(u)$, and let v, w two other vertices in X. W.l.o.g. (i.e., up to renaming of the colors), $c(u) = c(z)$ is `red`, $t(u) = c(v)$ is `blue`, and $c(w)$ is `green`. Notice that $t(v)$ cannot be `blue` (since v would be monochromatic) and cannot be `red` (since w would be monochromatic), hence $t(v)$ is `green`, and $t(w)$ is `red`. Then $\bar{C} = \langle u, v, w, u \rangle$ is a cycle. We have that $t(z)$ cannot be `red` (since z would be monochromatic) and it cannot be `green` (since z and w would be complementary). Then $t(z)$ is `blue`, and we can choose $C' = \langle z, v, w, z \rangle$, which is happy. ◀

We now show how to find an optimal partial swapping plan fulfilling the conditions of Lemma 8 in polynomial time. The idea is that of encoding the problem as an integer linear program (ILP) with a constant number of variables and a constant number of constraints, and then using Lenstra's algorithm [9] (whose running time is super-exponential in the number of variables and polynomial in the number of constraints) to solve such ILP.[3]

We define the type τ_v of a vertex v as the tuple $(c(v), t(v), \mathbb{1}_{x \in X})$ where $\mathbb{1}_{x \in X} = 1$ if $x \in X$ and 0 otherwise, and we let T_{vertex} be the set of all possible vertex types. Notice that two vertices of the same type are copies of one-another, hence we can describe any happy cycle by counting the number of nodes of each type. Given a cycle C, we define the type γ_C of C as as tuple that has one entry $\gamma_C(\tau)$ for each vertex type $\tau \in T_{\text{vertex}}$. The value of $\gamma_C(\tau)$ is the number of occurrences of vertices of type τ in C. Notice that there are only a constant number of distinct types γ that can be associated to happy cycles of length at most 4 (see Lemma 8), and we denote the set of all such types with T_{happy}.

Given a cycle type γ we denote by $\gamma(\tau)$ and $n_\tau(\mathcal{I})$ the number of nodes of type τ in γ and in the input instance $\mathcal{I}$, respectively. Our ILP has one variable $h_\gamma \in \mathbb{N}$ (where $\mathbb{N}$ denotes the set of all non-negative integers) for each type $\gamma \in T_{\text{happy}}$ that represents the number of occurrences of happy cycles of type γ that are part of a partial swapping plan. The constraint associated to a type $\tau \in T_{\text{vertex}}$ ensures that the partial swapping plan spans at most $n_\tau(\mathcal{I})$ vertices of type τ.

$$\begin{aligned}
\max \quad & \sum_{\gamma \in T_{\text{happy}}} h_\gamma \\
\text{s.t.} \quad & \sum_{\gamma \in T_{\text{happy}}} \gamma(\tau) h_\gamma \leq n_\tau(\mathcal{I}) && \forall \tau \in T_{\text{vertex}}, \\
& \qquad\qquad h_\gamma \in \mathbb{N} && \forall \gamma \in T_{\text{happy}}.
\end{aligned}$$

Once an optimal solution for the above ILP has been found, we can convert it into an optimal partial swapping plan $\widetilde{\mathcal{P}}$ in polynomial time, which, using Lemma 7 can be further converted into an optimal $\mathcal{P}$ for $\mathcal{I}$, and then into optimal sequence of swaps (see Lemma 2). We have therefore shown the following:

▶ **Theorem 9.** *The* CTS *problem on complete bipartite graphs and $k = 3$ colors can be solved in polynomial time.*

4 Arbitrary number of colors

We extend the ideas used in Section 3 for instances with $k = 3$ colors to solve CTS problem on complete bipartite graphs with k colors in time $O(\varphi(k) + n)$, for some function φ that depends only on k.[4]

[3] See [12] for a recent improvement over Lenstra's algorithm.

[4] We assume that the input graph G is a complete bipartite graph and that the sides X and Y of the bipartition can be found in time $O(n)$. This is the case, e.g., when G is represented using adjacency lists.

We will solve the problem instance via an ILP formulation with a number of variables and constraints that depends only on k. Unfortunately, some of the nice properties we have proved for the 3-color case, as the existence of optimal swapping plans containing only happy cycles (Lemma 6) and the existence of partial swapping plans maximizing the number of happy cycles and containing only short happy cycles (Lemma 8), are no longer true. Therefore, we need to find alternative structural properties of some optimal solutions for our problem instances.

We say that a cycle is *long* if its length is at least $2k+1$, and *short* otherwise. We first show the existence of optimal swapping plans with few long happy cycles.

▶ **Lemma 10.** *There exists an optimal swapping plan in which the number of happy cycles that are long is at most k.*

Proof. Given a generic swapping plan $\mathcal{P}$, we define $\Phi(\mathcal{P})$ as a vector, sorted in non-decreasing order, that contains one entry with value equal to the length $|C|$ of C for each long cycle C in $\mathcal{P}$.

We show that, if $\mathcal{P}$ has $\ell \geq k+1$ happy long cycles, then we can transform it into another swapping plan $\mathcal{P}'$ that has the same number of self-loops, X-cycles, Y-cycles, and XY-cycles and either has $\ell - 1$ happy long cycles, or is such that $\Phi(\mathcal{P}')$ is greater than $\Phi(\mathcal{P})$ in lexicographic order. Since there are only finitely many possible vectors $\Phi(\cdot)$, a repeated application of the above transformation eventually results in a swapping plan $\mathcal{P}^*$ with $f(\mathcal{P}^*) = f(\mathcal{P})$ and at most k happy long cycles.

Assume than that $\mathcal{P}$ has $\ell \geq k+1$ happy long cycles. For each such cycle C, the pigeonhole principle guarantees that we can find a pair $\{u, v\}$ of vertices in C that are both on the same side of the bipartition and such that $c(u) = c(v)$. We label C with the color of $c(u) = c(v)$ of these vertices.[5] Since there are $\ell \geq k+1$ cycles but at most k distinct labels, another invocation of the pigeonhole principle ensures that we can find two cycles C', C'' with the same label c. Assume w.l.o.g. that $|C'| \leq |C''|$ and let $\{u', v'\}$ and $\{u'', v''\}$ be the pairs of vertices chosen for C' and C'', respectively.

By Lemma 4 we can partition the vertices of C' into two cycles $C_{u'}, C_{v'}$, where $C_{u'}$ contains u' and $C_{v'}$ contains v'. Since u' and v' are on the same side and C is happy, at least one cycle $C^* \in \{C_u, C_v\}$ must also be happy. Let $\bar{C}$ the unique cycle in $\{C_u, C_v\} \setminus \{C^*\}$.

We choose $\mathcal{P}'$ as the swapping plan obtained from $\mathcal{P}$ by deleting C and C' and replacing them with C^* and the cycle obtained by merging $\bar{C}$ with C'' (see Lemma 3, and notice that vertex u'' in C'' has the same color c of some vertex in $\bar{C}$, which is either u' or v').

To relate $\Phi(\mathcal{P})$ to $\Phi(\mathcal{P}')$, we observe that the entry with value $|C''|$ corresponding to C in $\Phi(\mathcal{P})$ is replaced with an entry of value $|C''| + |\bar{C}| > |C''|$ in $\Phi(\mathcal{P}')$, the entry corresponding to C' decreases, and all other entries are unaffected. ◀

The following lemma considers optimal swapping plans with a few long happy cycles and shows the existence of such a swapping plan that additionally contains few unhappy cycles.

▶ **Lemma 11.** *There exists an optimal swapping plan $\mathcal{P}$ with at most k happy long cycles, $\eta_X(\mathcal{P}) \leq k/2$, and $\eta_Y(\mathcal{P}) \leq k/2$.*

This is needed since we obtain an algorithm with a running time of $O(n)$ for $k = O(1)$, but complete bipartiteness cannot be tested in time $o(n^2)$.

[5] Notice that there might be multiple such pairs of vertices for C, and that different choices can result in different labels for C. In this case, we arbitrarily choose one of the pairs.

Proof. Among all optimal swapping plans having most k happy long cycles, let $\mathcal{P}$ be one that minimizes $\max\{\eta_X(\mathcal{P}), \eta_Y(\mathcal{P})\}$ (the existence of $\mathcal{P}$ is guaranteed by Lemma 10) and assume towards a contradiction that $\max\{\eta_X(\mathcal{P}), \eta_Y(\mathcal{P})\} > k$.

The optimality of $\mathcal{P}$ ensures that no unhappy cycle $C \in \mathcal{P}$ contains any monochromatic vertex v, otherwise we could replace C with a self-loop on v (which is a short happy cycle), and an unhappy-cycle that spans all other vertices of C. This either increases the number of happy cycles by 1 without affecting the number of unhappy X- and Y-cycles (when $|C| > 2$) thus increasing $f(\cdot)$ by 1, or it increases the number of happy short cycles by 2 and removes exactly one unhappy cycle (when $|C| = 2$), thus increasing $f(\cdot)$ by at least 1.

In the rest of the proof we focus on the case $\eta_X(\mathcal{P}) \geq \eta_Y(\mathcal{P})$ since the complementary case is symmetric.

Since each X-cycle C in $\mathcal{P}$ contains vertices of at least two different colors, the pigeonhole principle ensures that we can find two vertices of the same color that belong to two distinct X-cycles C'_X and C''_X. Then C'_X and C''_X can be merged into a single cycle C^*_X (see Lemma 3).

If $\eta_X(\mathcal{P}) > \eta_Y(\mathcal{P})$, the swapping plan $(\mathcal{P} \setminus \{C'_X, C''_X\}) \cup \{C^*_X\}$ has the same number of self-loops, XY-cycles, and Y-cycles as $\mathcal{P}$ but one less X-cycle, which contradicts the optimality of $\mathcal{P}$.

Consider then $\eta_X(\mathcal{P}) = \eta_Y(\mathcal{P})$. Using analogous counting arguments as the ones used for X-cycles, we can find two Y-cycles C'_Y, C''_Y that can be merged into a single cycle C^*_Y. Then $\mathcal{P}^* = (\mathcal{P} \setminus \{C'_X, C''_X, C'_Y, C''_Y\}) \cup \{C^*_X, C^*_Y\}$ has the same set of self-loops and XY-cycles as $\mathcal{P}$, one less X-cycle, and one less Y-cycle. Hence we simultaneously have that (i) $\mathcal{P}^*$ has at most k happy long cycles, (ii) $f(\mathcal{P}) = f(\mathcal{P}^*)$ which implies that $\mathcal{P}^*$ is optimal, and (iii) $\max\{\eta_X(\mathcal{P}^*), \eta_Y(\mathcal{P}^*)\} = \eta_X(\mathcal{P}^*) = \eta_X(\mathcal{P}) - 1 = \max\{\eta_X(\mathcal{P}), \eta_Y(\mathcal{P})\} - 1$. This contradicts our choice of $\mathcal{P}$. ◀

As we will see, Lemma 11 allows us to model long happy cycles and unhappy cycles of an optimal swapping plan using a few additional variables in our ILP. However, as some of these cycles might be long, it is not clear how to guess the types of such long cycles. To deal with this issue, we introduce the concept of base cycles that will allow us to re-think of these long cycles as if they were short.

We say that a cycle of C is a *base cycle* if it does not contain two distinct vertices u, v of same type, i.e., $\tau_u = \tau_v$. Given a collection B of base cycles, we say that B is connected if the directed graph $H(B)$ that has one vertex for each vertex type that appears in some cycle in B, and an edge (τ, τ') iff the token color of type τ is the same as the vertex color of type τ', is strongly connected. We now provide two useful lemmas that show the connection between cycles and base cycles of the moves graphs.

▶ **Lemma 12.** *Given a cycle C, there exists a connected collection B of base cycles spanning the vertices in C such that each vertex in C is part of exactly one cycle in B.*

Proof. We build B by starting with $B = \{C\}$ and then iteratively replacing any cycle $\bar{C}$ in B containing two distinct vertices with the same color, with two (shorter) cycles that together span all and only the vertices in $\bar{C}$ exactly once (see Lemma 4). Notice that $\{C\}$ is connected and that the above operation preserves the connectedness property. ◀

▶ **Lemma 13.** *Given a connected collection of base cycles B, there exists a cycle C that spans all and only the vertices in B.*

Proof. We maintain a collection of cycles B' which initially coincides with B and we prove the claim by iteratively merging pairs of cycles in B' until B' contains a single cycle C.

We now show that, as long as $|B'| \geq 2$, there always exist two distinct cycles $C', C'' \in B'$ that can be merged. Let C' be an arbitrary cycle in B' and call $T_{C'}$ and T_B the set of all types of the vertices in C' and in (the cycles of) B, respectively. If $T_{C'} = T_B$ then C' can be merged with any other cycle in B' (see Lemma 3) and we are done. Otherwise, let (τ, τ') be some edge of $H(B)$ such that $\tau \in T_B \setminus T_{C'}$ and $\tau' \in T_{C'}$. Such an edge always exist since $H(B)$ is strongly connected. Let $C'' \neq C'$ be any cycle in B' that contains a vertex u of type τ, let v be the vertex that immediately follows u in C'' (possibly $v = u$), and let w be a vertex of type τ' in C'. Our choice of vertices ensures that $t(u) = c(v)$ and that $t(u) = c(w)$, hence $c(v) = c(w)$, and Lemma 3 implies that C' and C'' can be merged. ◀

Let $T_{\text{s-happy}}$ to be the set of all possible types of short happy cycles, and let T_{base} be the set of all possible types of base cycles. The cardinalities of s-happy and T_{base} depend only on k since they contain only cycles of length at most $2k$ and $2k^2$, respectively.

Given a cycle type $\gamma \in T_{\text{base}}$, we denote by $\beta_\gamma(B)$ the number of base cycles of type γ in B. Given a collection B of base cycles, the *signature* $\sigma(B)$ of B is the set containing all types $\gamma \in T_{\text{base}}$ such that there is at least one cycle of type γ in B. Notice that since $|T_{\text{base}}|$ only depends on k, the same also holds for the number of the possible distinct signatures $\sigma(B)$.

We guess the values ℓ^*, η^*_X and η^*_Y corresponding to number of long happy cycles, X-cycles $\eta_X(\mathcal{P}^*)$ and Y-cycles $\eta_Y(\mathcal{P}^*)$ of an optimal swapping plan $\mathcal{P}^*$. Thanks to Lemmas 10 and 11, we can restrict ourselves to $\ell^* \in \{0, \dots, k\}$ and $\eta^*_X, \eta^*_Y \in \{0, \dots, \lfloor k/2 \rfloor\}$.

Then, we look for a swapping plan that, (i) has exactly ℓ^* happy long cycles $C_1, C_2, \dots$, (ii) has exactly η^*_X X-cycles $C^X_1, C^X_2 \dots$, (iii) has exactly η^*_Y Y-cycles $C^Y_1, C^Y_2, \dots$, and (iv) maximizes the number of short happy cycles.

Instead of searching for a generic long happy cycle C_i, X-cycle C^X_i, or Y-cycle C^Y_i, Lemmas 12 and 13 together allow us to look for connected collections B_i, B^X_i, or B^Y_i of base cycles, respectively.

To this aim we further guess the signatures σ_i, σ^X_i, and σ^Y_i of each collection B_i, B^X_i, B^Y_i, respectively. In particular, σ_i must be some connected signature that involves at least one vertex type for each side of the bipartition, while σ^X_i and σ^Y_i must be connected signatures in which all types are on side X and Y, respectively. We can now write an integer linear program that has:

- one variable $h_\gamma \in \mathbb{N}$ for each type $\gamma \in T_{\text{s-happy}}$;
- one variable $h_{i,\gamma} \in \mathbb{N}^+$ associated to each type $\gamma \in \sigma_i$ which counts the number of base cycles of type γ in B_i (here $\mathbb{N}^+$ denotes the set of all positive integers);
- one variable $x_{i,\gamma} \in \mathbb{N}^+$ associated to each type $\gamma \in \sigma^X_i$ which counts the number of base cycles of type γ in B^X_i; and
- one variable $y_{i,\gamma} \in \mathbb{N}^+$ for each type $\gamma \in \sigma^Y_i$ which counts the number of base cycles of type γ in B^Y_i.

$$
\begin{aligned}
\max \quad & \ell^* + |\eta_X^* - \eta_Y^*| + \sum_{\gamma \in T_{\text{happy}}} h_\gamma \\
\text{s.t.} \quad & \sum_{i=1}^{\ell^*} \sum_{\gamma \in \sigma_i} \gamma(\tau) h_{i,\gamma} + \sum_{i=1}^{\eta_X^*} \sum_{\gamma \in \sigma_i^X} \gamma(\tau) x_{i,\gamma} \\
& \quad + \sum_{i=1}^{\eta_Y^*} \sum_{\gamma \in \sigma_i^Y} \gamma(\tau) y_{i,\gamma} + \sum_{\gamma \in T_{\text{happy}}} \gamma(\tau) h_\gamma = n_\tau(\mathcal{I}) && \forall \tau \in T_{\text{vertex}}, \\
& h_\gamma \in \mathbb{N} && \forall \gamma \in T_{\text{s-happy}}, \\
& h_{i,\gamma} \in \mathbb{N}^+ && \forall i \in \{1, \dots, \ell^*\} \;\; \forall \gamma \in \sigma_i, \\
& x_{i,\gamma} \in \mathbb{N}^+ && \forall i \in \{1, \dots, \eta_X^*\} \;\; \forall \gamma \in \sigma_i^X, \\
& y_{i,\gamma} \in \mathbb{N}^+ && \forall i \in \{1, \dots, \eta_Y^*\} \;\; \forall \gamma \in \sigma_i^Y.
\end{aligned}
$$

Similarly to Section 3, we once again solve the above ILP using Lenstra's algorithm [9], whose running time is upper bounded by a function of the number and variables and of the number of constraints of the ILP, both of which depend only on k.

After an optimal solution for the above ILP has been found, it needs to be converted into a corresponding swapping plan. We will now argue that this can be done in time $O(\varphi(k)+n)$, for some function $\varphi(\cdot)$ that depends only on k.

We first create a collection of buckets β_τ, with $\tau \in T_{\text{vertex}}$, where β_τ contains all vertices of G of type τ. We now assign the vertices of G to each of the $\sum_{\gamma \in T_{\text{happy}}} h_\gamma$ short happy cycles, and to each base cycle in the collections B_i, B_i^X, B_i^Y corresponding to the ℓ^* long happy cycles C_i, η_X^* unhappy X-cycles C_i^X, and η_Y^* unhappy Y-cycles C_i^Y. This can be done by drawing the vertices of the needed types from the corresponding bucket. Once this assignment is complete each of the above cycles can be built by brute-force from the assigned vertices in a time that depends only on k.

It remains to transform each (connected) collection of base cycles into a single cycle the spans the same vertices (whose existence is guaranteed by Lemma 13). It is not hard to come up with an algorithm that performs this task in time $O(k + n_B)$ for a collection spanning n_B vertices (see, e.g., Appendix B). Since there are $\ell^* + \eta_X^* + \eta_Y^* \le 2k$ such collections, and each vertex of G is spanned by at most one base cycle, this takes time $O(\varphi(k) + n)$. By Lemma 2, the resulting optimal swapping plan can be converted into an optimal sequence of swaps in time $O(n)$, thus we have:

▶ **Theorem 14.** *The CTS problem on complete bipartite graphs with k colors can be solved in time $O(\varphi(k) + n)$ for a suitable function $\varphi(\cdot)$ that depends only on k.*

5 NP-hardness of CTS on complete bipartite graphs

We show that CTS on bipartite graphs is NP-Hard by reducing from CTS on cliques, which has been shown to be NP-Hard in [3].

Given an instance $\mathcal{I}_H$ of CTS on a clique H with n_H vertices, we create an instance $\mathcal{I}_G$ of CTS on a complete bipartite graph $G = (V, E)$ with $n = 2n_H$ vertices, where V is partitioned into two sets X and Y which correspond to the two sides of the bipartition.

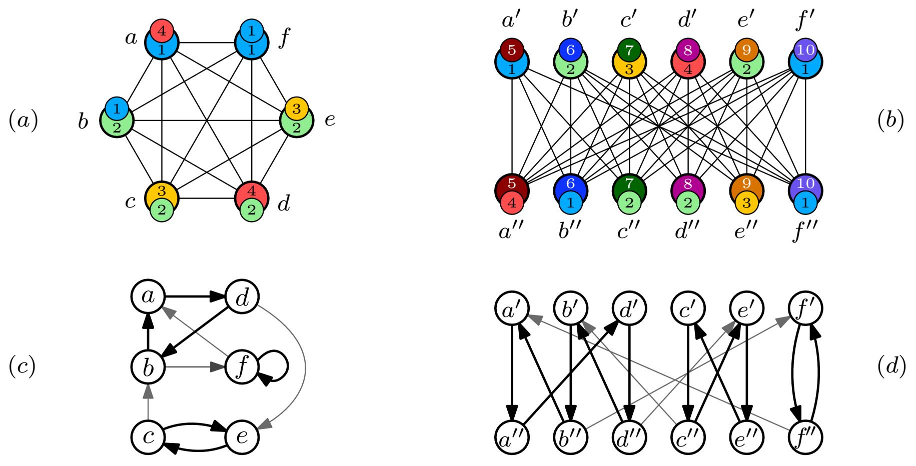

Figure 4 An example of the CTS instance $\mathcal{I}_H$ on a clique graph with $n_H = 6$ vertices is shown in (a). The corresponding instance $\mathcal{I}_G$ on a complete bipartite graph with $2n_H$ vertices defined by our reduction is shown in (b). In (c) and (d) we have the moves graphs of instances $\mathcal{I}_H$ and $\mathcal{I}_G$, respectively. Notice that any directed edge (u, v) in the moves graph of $\mathcal{I}_H$ is modelled by the path consisting of the two directed edges (u', u'') and (u'', v') in the moves graph of $\mathcal{I}_G$, the first one going from a vertex of X to a vertex of Y and the second one going from a vertex of Y to a vertex of X. Thus any swapping plan for $\mathcal{I}_H$ corresponds to a swapping plan for $\mathcal{I}_G$, and vice-versa. The bold edges depict an optimal swapping plan for $\mathcal{I}_H$ and the corresponding swapping plan for $\mathcal{I}_G$.

The graph G of $\mathcal{I}_G$ is the complete bipartite graph obtained by creating two copies v', v'' of each vertex v in H and placing v' in X and v'' in Y. The set of colors of $\mathcal{I}_G$ consists of all the colors in $\mathcal{I}_H$ plus one new color c_v for each vertex v of H. We set $c(v') = c(v)$, $t(v') = c(v'') = c_v$, and $t(v'') = t(v)$. Let M be the moves graph of $\mathcal{I}_G$.[6] See Figure 4 for an example.

▶ **Lemma 15.** *The sets X and Y are a bipartition of M. Moreover, each vertex $v' \in X$ has a single outgoing edge incident to it, which is the edge (v', v'').*

Proof. Let $v' \in X$. Our construction of $\mathcal{I}_G$ ensures that $t(v') = c_v$ and that the only vertex of color c_v is v''. This implies that (v', v'') is in M as is the only outgoing edge of v'.

We now show that the moves graph M is bipartite by proving that there cannot be any edge (u'', v'') between two vertices $u'', v'' \in Y$. Indeed, as each vertex $v' \in X$ has a single outgoing edge incident to it which enters a vertex of Y, it cannot be the case that M contains an edge between two vertices of X. Consider now a generic vertex $u'' \in V$. Since $t(u'')$ is one of the colors of $\mathcal{I}_H$, while each $v'' \in Y$ has color $c(v'') = c_v$, which is one of the colors that has been introduced in $\mathcal{I}_G$ but was not in $\mathcal{I}_H$, we conclude that (u'', v'') is not in M. ◀

▶ **Lemma 16.** *There exists a swapping plan $\mathcal{P}_H$ for $\mathcal{I}_H$ if and only if there exists a swapping plan $\mathcal{P}_G$ for $\mathcal{I}_G$ with $|\mathcal{P}_G| = |\mathcal{P}_H|$.*

Proof. Let $\mathcal{P}_H$ be a swapping plan for $\mathcal{I}_H$. The swapping plan $\mathcal{P}_G$ for $\mathcal{I}_G$ contains one cycle C' for each cycle $C \in \mathcal{P}_H$. The cycle C' is obtained by renaming each vertex v of C' into v' (i.e., the copy of v on side X of G) and then splitting each resulting edge (u', v') into the two

[6] With a little abuse of notation, we use the same functions c and t to denote the colors of the vertices and of the tokens of both instances, respectively.

edges (u', u'') and (u'', v') via the inclusion of the intermediate vertex u'' (see Figure 4 for an example). As each vertex v of H is spanned by a unique cycle in $\mathcal{P}_H$, say C, by construction of $\mathcal{P}_G$, the two vertices v' and v'' are spanned by the unique cycle C' that corresponds to C. This implies that $\mathcal{P}_G$ is a swapping plan for $\mathcal{I}_G$.

Consider now a swapping plan $\mathcal{P}_G$ for $\mathcal{I}_G$. The swapping plan $\mathcal{P}_H$ for $\mathcal{I}_H$ contains one cycle C for each cycle $C' \in \mathcal{P}_G$. Lemma 15 implies that a generic cycle $C' \in \mathcal{P}_G$ consists of an alternation of vertices from X and Y, where each vertex u'' of C' that is in Y is immediately preceded by vertex $u' \in X$. The cycle C is obtained from C' by considering each vertex u'' of C that is in Y, deleting it, and replacing its two incident edges in C, namely the incoming edge (u', u'') and an outgoing edge (u'', v') for some $u' \in X$, with the edge (u, v). As each vertex v' of G is spanned by a unique cycle in $\mathcal{P}_G$, say C', by construction of $\mathcal{P}_H$, v is spanned by the corresponding cycle C. Therefore $\mathcal{P}_H$ is a swapping plan for $\mathcal{I}_H$. ◀

Lemma 15 implies that all cycles in M are happy. Hence, all swapping plans $\mathcal{P}_G$ for $\mathcal{I}_G$ have $\eta_X(\mathcal{P}_G) = \eta_Y(\mathcal{P}_G) = 0$ and, from Equation (1), we have $f(\mathcal{P}_G) = |\mathcal{P}_G|$. Therefore the minimum number of swaps needed to solve $\mathcal{I}_G$ is $n - f(\mathcal{P}^*_G)$, where $\mathcal{P}^*_G$ is a swapping plan for $\mathcal{I}_G$ that maximizes $|\mathcal{P}^*_G|$.

As shown in [4], the minimum number of swaps needed to solve $\mathcal{I}_H$ is $n_H - |\mathcal{P}^*_H|$, where $\mathcal{P}^*_H$ is a swapping plan of maximum cardinality for $\mathcal{I}_H$. Lemma 16 and the above discussion imply that $|\mathcal{P}^*_H| = |\mathcal{P}^*_G|$, and hence $\mathcal{I}_G$ admits a solution with at most $n - x$ swaps if and only if $\mathcal{I}_H$ admits as solution with at most $n_H - x$ swaps. We thus have the following:

▶ **Theorem 17.** *The* Colored Token Swapping *problem on complete bipartite graphs is* NP*-hard.*

Moreover the `CTS` problem on a clique of n_H vertices cannot be solved in time $2^{o(n_H)}$ unless the exponential time hypothesis fails [3, 7], and our reduction ensures that $n = \Theta(n_H)$, a similar result also holds for complete bipartite graphs:

▶ **Theorem 18.** *The* Colored Token Swapping *problem on complete bipartite graphs cannot be solved in time $2^{o(n)}$, unless the exponential time hypothesis fails.*

References

1 Oswin Aichholzer, Erik D. Demaine, Matias Korman, Anna Lubiw, Jayson Lynch, Zuzana Masárová, Mikhail Rudoy, Virginia Vassilevska Williams, and Nicole Wein. Hardness of token swapping on trees. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 3:1–3:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPICS.ESA.2022.3`.

2 Ahmad Biniaz, Kshitij Jain, Anna Lubiw, Zuzana Masárová, Tillmann Miltzow, Debajyoti Mondal, Anurag Murty Naredla, Josef Tkadlec, and Alexi Turcotte. Token swapping on trees. *Discret. Math. Theor. Comput. Sci.*, 24(2), 2022. `doi:10.46298/DMTCS.8383`.

3 Édouard Bonnet, Tillmann Miltzow, and Pawel Rzazewski. Complexity of token swapping and its variants. *Algorithmica*, 80(9):2656–2682, 2018. `doi:10.1007/S00453-017-0387-0`.

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

5 Matt Groening, James L. Brooks, Sam Simon, Dan Castellaneta, and Deb Lacusta. The Simpsons. Fox Broadcasting Company, 1999. Season 11, Episode 18 (Days of Wine and D'oh'ses), Timestamp: 18:30. URL: `https://www.imdb.com/title/tt0701222`.

6 Lenwood S. Heath and John Paul C. Vergara. Sorting by short swaps. *J. Comput. Biol.*, 10(5):775–789, 2003. `doi:10.1089/106652703322539097`.

7 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/JCSS.2000.1727.

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

9 Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983. doi:10.1287/MOOR.8.4.538.

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

11 Tillmann Miltzow, Lothar Narins, Yoshio Okamoto, Günter Rote, Antonis Thomas, and Takeaki Uno. Approximation and hardness of token swapping. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 66:1–66:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICS.ESA.2016.66.

12 Victor Reis and Thomas Rothvoss. The subspace flatness conjecture and faster integer programming. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 974–988. IEEE, 2023. doi:10.1109/FOCS57990.2023.00060.

13 Theresa P. Vaughan. Factoring a permutation on a broom. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 30:129–148, 1999.

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

15 Katsuhisa Yamanaka, Takashi Horiyama, J. Mark Keil, David G. Kirkpatrick, Yota Otachi, Toshiki Saitoh, Ryuhei Uehara, and Yushi Uno. Swapping colored tokens on graphs. *Theor. Comput. Sci.*, 729:1–10, 2018. doi:10.1016/J.TCS.2018.03.016.

16 Katsuhisa Yamanaka, Takashi Horiyama, David G. Kirkpatrick, Yota Otachi, Toshiki Saitoh, Ryuhei Uehara, and Yushi Uno. Swapping colored tokens on graphs. In Frank Dehne, Jörg-Rüdiger Sack, and Ulrike Stege, editors, *Algorithms and Data Structures - 14th International Symposium, WADS 2015, Victoria, BC, Canada, August 5-7, 2015. Proceedings*, volume 9214 of *Lecture Notes in Computer Science*, pages 619–628. Springer, 2015. doi:10.1007/978-3-319-21840-3_51.

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

A Converting a swapping plan to a solution

In this section we show how a swapping plan $\mathcal{P}$ for and instance of CTS can be transformed into a solution consisting of $n - f(\mathcal{P})$ swaps in time $O(n)$. This is essentially an algorithmic rendition of the inductive proof of [14], which is constructive, and also implies the correctness of our algorithm.

We say that a vertex v of some cycle C of a swapping plan is *special* if the two vertices immediately following v in C are both on the opposite side of the bipartition compared to v.

As long as there is a cycle in $\mathcal{P}$ that is not a self-loop, the algorithm iteratively finds two nodes on opposite sides of the bipartition, swaps their tokens, and updates $\mathcal{P}$ to be a swapping plan for the resulting configuration. In particular, each iteration of the algorithm executes the first of applicable rule among the following three:

1. If $\mathcal{P}$ contains some XY-cycle C, then choose three vertices u, v, w of C as follows:
   - If C contains no special vertices (this is always the case when $|C| = 2$), let (u, v) be an arbitrary edge of C, and let w be the vertex that appears immediately after v in C (possibly $w = u$).

- Otherwise (i.e., $|C| \geq 3$ and C contains some special vertex), let u be a special vertex, and let v, w be the two vertices that appear immediately after u in C, in this order.

   Then swap the tokens on u and v. To update $\mathcal{P}$ replace (u, v) and (v, w) with the edges (u, w) and (v, v). This has the effect of removing v from C (possibly resulting in a self-loop) and creating a new self-loop on v.

2. If $\mathcal{P}$ contains an X-cycle C and an Y-cycle C', let u (resp. v) be an arbitrary vertex in C (resp. C'). Swap the tokens on u and v. To update $\mathcal{P}$ remove the two edges (u, u') and (v, v') outgoing from u and v, respectively, and add the two new edges (u, v') and (v, u'). This causes C and C' to merge into a single XY-cycle.
3. Otherwise let C be any X-cycle or Y-cycle in $\mathcal{P}$. Choose an arbitrary vertex u from C and an arbitrary (singleton) vertex v from the opposite side of the bipartition. Swap the tokens on u and v. To update $\mathcal{P}$ delete the edge (u, u') outgoing from u and the self-loop (v, v), and replace them with the edges (u, v) and (v, u'). This has the effect of merging C and $\langle v, v \rangle$ into a single cycle.

We now argue that the above algorithm can be implemented to run in time $O(n)$. We keep track of four lists of cycles that store self-loops, X-cycles, Y-cycles, and XY-cycles, respectively. Each C is represented by storing a doubly-linked list L_{vertices} of its vertices (in order), the number of vertices of C that are in X and in Y (respectively), and a list L_{special} of the special nodes of C. The generic element that stores a node v in L_{special} also has a pointer to the element that stores v in L_{vertices}, and vice-versa. Clearly, given $\mathcal{P}$, we can initialize the above data structures in time $O(n)$. Moreover, selecting the next rule to apply, and updating the data structure following the changes to $\mathcal{P}$ dictated by such rule can be done time $O(1)$. Hence the overall running time is $O(n + |\mathcal{P}|) = O(n)$.

B Merging a connected collection of base cycles in time $O(n)$

In this section we argue that the cyclces of a connected collection B of base cycles can be merged into a single cycle C^* in time $O(k + n_B)$, where n_B denotes the overall number of vertices spanned by the base cycles in B.

We start by building an auxiliary undirected bipartite graph H' in which the *color-vertices* on one side of the bipartition are the distinct colors of the vertices spanned by the cycles in B, and the *cycle-vertices* on the other side of the bipartition are the base cycles in B. H contains an edge (c, C) iff there is some vertex in cycle C that has color c, and this edge is labelled with the name of any such vertex. The above auxiliary graph can be built in time $O(n)$.[7]

▶ **Lemma 19.** *The graph H' is connected.*

Proof. Since each color-vertex in G has at least one cycle-vertex as a neighbor, it suffices to show that there exists a walk W in H' between any two distinct cycles $C, C' \in B$.

[7] For each $C \in B$, we can find a collection L_C containing exactly one vertex v with $c(v) = c$ for each distinct color c that appears in C. We start with $L_C = \emptyset$ and we examine the vertices of C one at a time while updating a *global* array of k flags. The c-th flag is set to true iff some vertex of color c has already been encountered in C. Whenever a vertex v of a new color c is encountered, the corresponding flag is set to true and v is added to L_C. After all the vertices of C have been processes, the flags of the colors in L_C are reset to false. The overall running time is $O(k + n_B)$.

Let τ and τ' denote the types of two arbitrary vertices from C and C', respectively. Since $H(B)$ is connected, there exists some (directed) path $\pi = \langle \tau = \tau_1, \tau_2, \dots, \tau_\ell = \tau'$ between τ and τ' in $H(B)$. Let $C'_1 = C, C'_\ell = C'$ and, for $i = 2, \dots, \ell - 1$, choose C'_i as an arbitrary cycle from B that contains some vertex of type τ_i. Moreover, let c_i and t_i be the colors of the vertex and the token of type τ_i, respectively.

For $i = 1, \dots, \ell - 1$, C_i contains a vertex of color $t_i = c_{i+1}$ and so does C_{i+1}. Hence the two edges (C_i, c_{i+1}) and (c_{i+1}, C_{i+1}) form a walk W_i between C_i and C_{i+1} in H'. We choose W as the composition of all walks $W_1, \dots, W_{\ell-1}$, in this order. ◀

Next, we compute a spanning tree T of H rooted in an arbitrary cycle-vertex (T exists by Lemma 19) and we iteratively (i) locate the deepest color vertex c; (ii) merge all its neighboring cycles (i.e., the parent all the children of C) in T into a single cycle C' by repeatedly performing merge operations; and (iii) delete c and all its children, and replace the former parent of c with a new cycle-vertex corresponding to C'. We stop this process when T contains a single cycle C^* as its root, and we return C^*.

Notice that the time spent to process T is proportional to the number n_T of vertices in T. Indeed the color vertices c can be listed in order of depth in time $O(n_T)$ by a BFS visit of T, and step (ii) can be performed in time proportional to the number of neighbors of c by exploiting the fact that a generic edge (c, C) incident to c in T is labelled with a vertex of C having color (see also Lemma 3). The overall time spent is therefore $O(k + n_T) = O(k + n_B)$ since H (and hence T) contains at most k color-vertices and $n_B/2$ cycle-vertices.