Computational Complexity of Matching Match Puzzle

Yuki Iburi
The Digital Value, LTD., Tokyo, Japan

Ryuhei Uehara
School of Information Science, Japan Advanced Institute of Science and Technology, Tokyo, Japan

Abstract

Various forms of graph coloring problems have been studied over the years in the society of graph theory. Recently, some original puzzles are popularized in Japanese 100-yen shops, and one of them can be formalized as a graph coloring problem in a natural way. However, this natural graph coloring problem has not been investigated in the context of the graph theory. In this paper, we investigate this puzzle as a graph coloring problem. We first prove that this graph coloring problem is NP-complete even when the graph is restricted to a path or a spider. In these cases, diameter of the graphs seems to play an important role for its difficulty. We then show that the problem can be solved in polynomial time when the graph is restricted to some graph classes of constant diameter.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Graph coloring, Matching Match puzzle, NP-complete, polynomial-time solvable

Digital Object Identifier 10.4230/LIPIcs.FUN.2024.17

Funding *Ryuhei Uehara*: JSPS KAKENHI Grant Numbers 20H05961, 20H05964, 24H00690.

1 Introduction

Research on computational complexity of puzzles and games has a long history. One of the reasons is that there exist some puzzles that characterize major computational complexity classes in natural and simple ways, and hence the features of these puzzles give us some understanding of such complexity classes [3, 6]. Especially, many NP-complete puzzles have helped us to obtain some intuitions for the class NP. They may lead us to the solution to the P≠NP conjecture, which is one of the millennium prize problems.

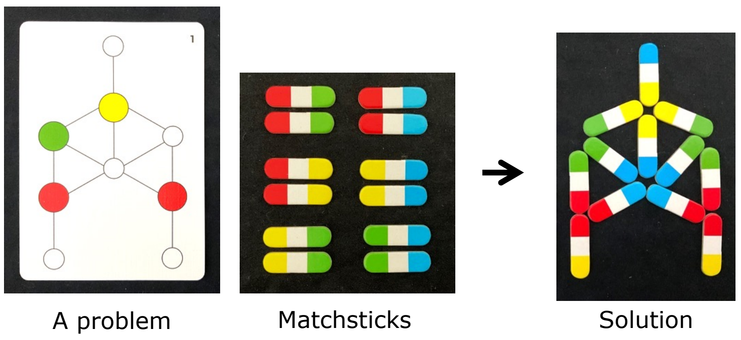

Figure 1 The matching match puzzle sold at Daiso.

In this paper, we investigate a puzzle named "Matching Match" (Figure 1). This puzzle is designed by Rikachi, a Japanese puzzle designer, and it is a commercial product produced by Daiso, which is one of the major 100-yen shops in Japan.[1] As a commercial product, the rule of this puzzle is simple and understandable even for children. You are given a card and a set of "matchsticks". On the card, a planar graph is drawn and some vertices of it are colored. Each matchstick has its own colors on both endpoints (the colors can be the same). The number of matchsticks is equal to the number of edges of the graph. That is, each matchstick corresponds to an edge of the graph and vice versa. The puzzle asks us to find an arrangement of the matchsticks on the edges so that every color matches at each vertex of the graph. The color of an endpoint should be matched to the color on the vertex if the vertex is colored. On the other hand, we have to assign a color to each vertex if it is not colored. That is, at a vertex without color, all the endpoints of the matchsticks sharing the vertex have to have the same color.

It is easy to see that the puzzle can be easily solved if every vertex of the graph on a card has its color. Namely, for a given graph, this puzzle asks us to find a proper coloring of the uncolored vertices in the graph for a given set of edges with pre-colored endpoints. As you can find in Figure 1, it is a natural problem not only in the context of the puzzle, but also as a variant of graph coloring problems. The graph coloring problem is one of classic problems which has been widely investigated in the context of theoretical computer science [4]. From the viewpoint of algorithmic technique, the color-coding is one of technique for solving a graph coloring problem efficiently [1]. However, as far as the authors know, this graph coloring problem corresponding to the matching-match puzzle has never been investigated in the context of graph coloring. One of the reasons may be that this graph coloring admits to color two neighbors with one color when we have an edge with endpoints with pre-colored by the same color. In fact, we will use such edges in our reduction.

We first show that the matching-match puzzle is NP-complete even if the graph is quite restricted. Precisely, this puzzle is NP-complete even if the graph is a spider, a path, or a cycle in general. On the other hand, when the graph is a complete graph or a star the matching-match puzzle is polynomial-time solvable. We note that a star is a spider with legs of length 1. That is, we have a constant B' such that the matching-mach puzzle is NP-complete on spiders with legs of length at least B', and it is polynomial-time solvable on spiders with legs of length at most B'. In this paper, we also show a polynomial-time algorithm on spiders with legs of length at most 2. That is, we prove that $B' \geq 2$. The keen threshold value of B' is open.

2 Preliminaries

In this paper, we only consider a simple graph $G = (V, E)$ with $|V| = n$ and $|E| = m$. A sequence of the vertices $(v_0, v_1, \ldots, v_k)$ is a *path* of *length* k in G when $\{v_i, v_{i+1}\} \in E$ for each $i = 0, \ldots, k-1$. It is a *cycle* of size k if $v_0 = v_k$ and $k > 2$. A graph is a *tree* if it is acyclic and connected. A tree is a *spider* if it has only one vertex of degree greater than 2. The unique vertex of degree greater than 2 of a spider is called the *body* of it. A spider consists of three or more paths sharing the body. Each path (including the body) is called a *leg* of the spider. A graph is a *star* if it is a spider with legs of length 1. A graph is *complete* if every pair of vertices are joined by an edge. A complete graph is denoted by K_n if it consists of n vertices. For a vertex v in V, its *neighbor set* in $G = (V, E)$ is defined by

[1] English instruction can be found at `https://www.daiso-syuppan.com/noutore/`.

$\{u \mid \{u,v\} \in E\}$ and denoted by $N_G(v)$ (or $N(v)$ if G is clear in the context). We also use a notation $N_G[v]$ and $N[v]$ defined by $N_G(v) \cup \{v\}$. The *distance* between two vertices u and v in G is the minimum length of a path joining u and v. The diameter of G is the maximum distance between all pairs of vertices in G. For a given graph $G = (V, E)$, an edge set M is a *matching* if no two edges share a common vertex. For a given graph $G = (V, E)$ and a vertex subset V', we call the graph $G' = (V', E')$ with $E' = \{\{v, v'\} \mid v, v' \in V' \text{ and } \{v, v'\} \in E\}$ an *induced subgraph* by V', and it is denoted by $G[V']$. A vertex set Q is called a *clique* if $G[Q]$ is a complete graph.

Now we turn to the definition of the matching-match puzzle. An instance of the matching-match puzzle consists of a graph $G = (V, E)$ with $V = \{v_0, v_1, \ldots, v_{n-1}\}$ and a set of *sticks* $S = \{s_0, s_1, \ldots, s_{m-1}\}$. We define the set of endpoints of sticks by U. That is, each stick s_i is a pair of distinct vertices $\{u_i, u'_{i,2}\}$, where $u_i, u'_{i,2} \in U$ and hence $|U| = 2m$. We will use two color sets $C_0 = \{0, 1, \ldots, c\}$ and $C_1 = \{1, \ldots, c\}$ for some positive integer c to distinguish colors in V and U. Each vertex v in V is colored by a function $\mathcal{C}_0 : V \to C_0$ and each vertex u in U is colored by a function $\mathcal{C}_1 : U \to C_1$. We say a vertex v in V is *not colored* if $\mathcal{C}_0(v) = 0$. The other vertices in V and U are *colored*. Then the input of the matching-match puzzle is a 5-tuple $(G, S, c, \mathcal{C}_0, \mathcal{C}_1)$. An instance $(G, S, c, \mathcal{C}_0, \mathcal{C}_1)$ is *feasible* if and only if there exists a mapping $\mathcal{M}$ from U to V such that (1) for each $s_i = (u_i, u'_i)$, $\{\mathcal{M}(u_i), \mathcal{M}(u'_i)\}$ is in E, (2) for each $u \in U$, $\mathcal{C}_1(u) = \mathcal{C}_0(\mathcal{M}(u))$ or $\mathcal{C}_0(\mathcal{M}(u)) = 0$, (3) for each $e \in E$, there exists a stick $s_i = \{u_i, u'_i\}$ with $e = \{\mathcal{M}(u_i), \mathcal{M}(u'_i)\}$, and (4) for each $v \in V$ with $\mathcal{C}_0(v) = 0$, there exists a color c' such that all vertices $u \in U$ with $\mathcal{M}(u) = v$ satisfy $\mathcal{C}_1(u) = c'$. The *matching-match puzzle* asks us if there exists a feasible mapping $\mathcal{M}$ for a given instance $(G, S, c, \mathcal{C}_0, \mathcal{C}_1)$. That is, we can formalize the matching-match puzzle as follows:

<u>MATCHING-MATCH PUZZLE</u>
Input: A graph $G = (V, E)$, a set S of sticks, an integer $c > 0$, and two functions $\mathcal{C}_0$ and $\mathcal{C}_1$.
Output: Determine whether there exists a feasible mapping $\mathcal{M}$.[2]

We note that we consider general graphs and they are not necessarily planar. However, almost all graphs in this paper are planar except complete graphs (with at least 5 vertices). Except for complete graphs, all graphs in this paper can be drawn on a plane with edges of unit length without crossing. That is, they can be real problems in Matching Match puzzle.

3 NP-completeness

In this section, we prove that the matching-match puzzle is intractable even if G is quite restricted.

▶ **Theorem 1.** *The matching-match puzzle is NP-complete in general even if G is (1) a spider, (2) a path, or (3) a cycle.*

Proof. Let $(G, S, c, \mathcal{C}_0, \mathcal{C}_1)$ be an instance of the matching-match puzzle. If there is a feasible mapping $\mathcal{M}$, it is easy to confirm that $\mathcal{M}$ is feasible. Thus the matching-match puzzle is in NP. Therefore we show NP-hardness. To show NP-hardness, we reduce the following 3-Partition problem to our problem:

[2] In the commercial puzzle, each of 40 instances has a unique solution. We do not assume it in this paper.

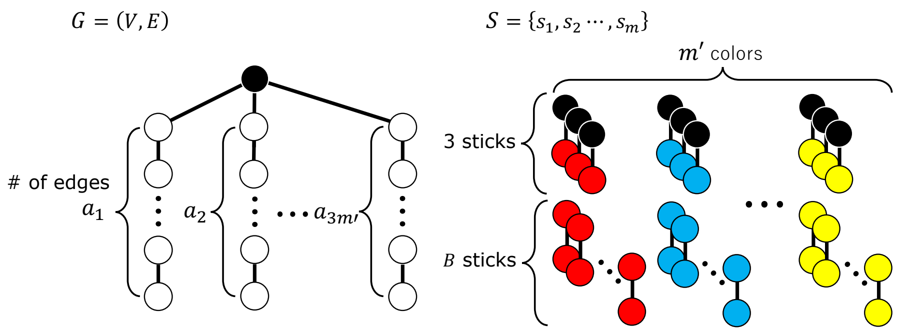

Figure 2 Construction of a spider.

3-PARTITION

Input: Positive integers $a_1, a_2, a_3, \ldots, a_{3m'}$ such that $\sum_{i=1}^{3m'} a_i = m'B$ for some positive integer B and $B/4 < a_i < B/2$ for $1 \leq i \leq 3m'$.

Output: Determine whether we can partition $\{1, 2, \ldots, 3m'\}$ into m' subsets $A_1, A_2, \ldots, A_{m'}$ so that $\sum_{i \in A_j} a_i = B$ for $1 \leq j \leq m'$.

It is well known that the 3-PARTITION problem is strongly NP-complete [2].

(1) We first show a reduction to a spider. For a given instance $a_1, a_2, a_3, \ldots, a_{3m'}$ of the 3-PARTITION problem, we construct G and S as follows (Figure 2). We use $m' + 1$ colors, namely, $C_0 = \{0, 1, \ldots, m' + 1\}$. The graph G is a spider with $3m'$ legs. For each i with $1 \leq i \leq 3m'$, the ith leg is a path of length $a_i + 1$. The body of the spider is colored by 1, and all the other vertices are not colored. That is, $\mathcal{C}_0(b) = 1$ for the body vertex $b \in V$ and $\mathcal{C}_0(v) = 0$ for each vertex $v \in V \setminus \{b\}$. The set S consists of $m = m'(3 + B)$ sticks. For each j with $1 \leq j \leq m'$, S contains B sticks $s = \{u, w\}$ with $\mathcal{C}_1(u) = \mathcal{C}_1(w) = j + 1$ and 3 sticks $s' = \{u', w'\}$ with $\mathcal{C}_1(u') = 1$ and $\mathcal{C}_1(w') = j + 1$.

Clearly, the reduction can be done in polynomial time. Thus we show that the instance $a_1, a_2, a_3, \ldots, a_{3m'}$ of the 3-PARTITION problem has a solution if and only if G and S are feasible.

We first assume that the instance $a_1, a_2, a_3, \ldots, a_{3m'}$ has a solution. Then they can be partitioned into m' subsets $A_1, A_2, \ldots, A_{m'}$ such that $\sum_{i \in A_j} a_i = B$ for $1 \leq j \leq m'$. Without loss of generality, we assume that $A_1 = \{a_1, a_2, a_3\}$. Then we match three sticks $s_1 = s_2 = s_3 = \{1, 2\}$ to indicate three legs of lengths a_1, a_2, and a_3. That is, the function $\mathcal{M}$ maps each 1 to the body of the spider, and 2 to the neighbors of the body corresponding to the three legs of lengths a_1, a_2, and a_3. Now we match the other sticks $\{2, 2\}$ to these three legs. Since $a_1 + a_2 + a_3 = B$, we can match all sticks to these legs and no sticks $\{2, 2\}$ remain. We repeat the same process for each of $A_2, \ldots, A_{m'}$. Since they are a solution of the 3-PARTITION problem, we can match all the sticks.

We next assume that we can match all the sticks in S to the edges of the spider G. We first observe that all sticks that have an endpoint of color 1 to join legs to the body. Therefore, we cannot use two or more colors on a leg except the edge joining the leg to the body. Thus, each set of B sticks of the same color should be on three legs such that the total length is B. Hence we can construct a solution of the 3-PARTITION problem from the solution of the matching-match puzzle.

(2) We next show a reduction to a path. The basic idea is similar to the case (1) (Figure 3). The graph G is a path $(v_0, v_1, \ldots, v_m)$ of length $m = m'(B + 6)$. It has $3m' + 1$ vertices v with $C_0(v) = 1$, and all the other vertices are not colored. By the vertices v with

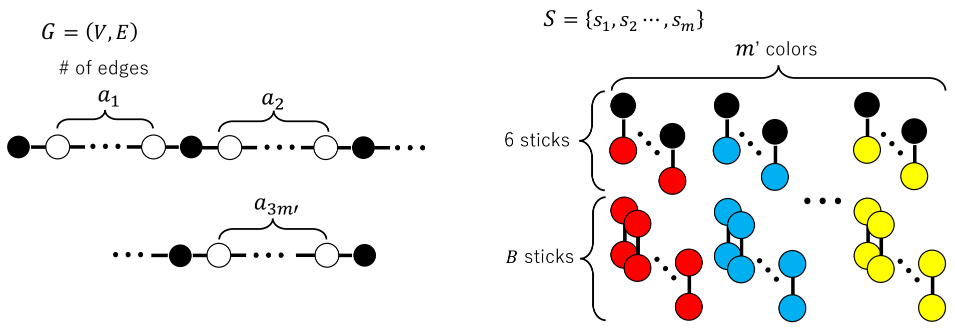

Figure 3 Construction of a path.

$C_0(v) = 1$, the path is partitioned into $3m'$ subpaths since both endpoints v of the path satisfy $C_0(v) = 1$. The ith subpath has length a_i for each $i = 1, 2, \ldots, 3m'$. The set S consists of $m = m'(6+B)$ sticks. For each j with $1 \le j \le m'$, S contains B sticks $s = \{u, w\}$ with $\mathcal{C}_1(u) = \mathcal{C}_1(w) = j+1$ and 6 sticks $s' = \{u', w'\}$ with $\mathcal{C}_1(u') = 1$ and $\mathcal{C}_1(w') = j+1$. The reduction can be done in polynomial time. By using the same argument, we can show that the instance $a_1, a_2, a_3, \ldots, a_{3m'}$ of the 3-Partition problem has a solution if and only if G and S are feasible.

(3) In the construction (2), by unifying the endpoints of the path G, we can obtain a cycle. On the cycle, the same argument works.

Therefore, the matching-match puzzle is NP-complete in general even if G is a spider, a path, or a cycle. ◀

By the proof of Theorem 1(1), we obtain the following corollary.

▶ **Corollary 2.** *The matching-match puzzle is NP-complete even if G is a spider and its body is the only colored vertex. Moreover, the matching-match puzzle is NP-complete even if G is a spider and no vertex is colored.*

Proof. The proof of Theorem 1(1) meets the first claim. Thus we focus on the case that the vertices in G are not colored. The construction of the graph G and the set S is the same as the proof of Theorem 1(1). The coloring of $\mathcal{C}_1$ is also the same, and we define $\mathcal{C}_0(v) = 0$ for all vertices in V. To derive a contradiction, we assume that the body b of the spider G is colored by the other color, say $\mathcal{C}_0(b) = 2$, than 1 in a solution. In the original instance of the 3-Partition problem, we can assume that $m' > 6$ without loss of generality. Since we have at most B sticks $\{u, u'\}$ with $\mathcal{C}_1(u) = \mathcal{C}_1(u') = 2$ to cover the legs, we have to change the color on some legs from 2 to the other colors in the middle of the legs. To change the color, we have to use the vertices v with $\mathcal{C}_0(v) = 1$. However, we have only three sticks $\{u, u'\}$ with $\mathcal{C}_1(u) = 2$ and $\mathcal{C}_1(u') = 1$. Thus, since $m' > 6$, there are at least 4 legs that should be totally colored by 2 from the body to their leaves. However, by the assumption that $B/4 < a_i < B/2$, we cannot cover 4 legs by B sticks, which is a contradiction. Therefore, we cannot color the body by any other color than 1. That is, we can assume that $\mathcal{C}_0(b) = 1$ for the body vertex b without loss of generality. Thus the matching-match puzzle is NP-complete even if no vertex in G is colored. ◀

4 Polynomial time algorithms

In this section, we show polynomial-time algorithms for the matching-match puzzle on some graph classes. We first consider two simple cases that G is a complete graph and G is a star. In these two cases, we can solve the matching-match puzzle efficiently. Next we turn to the spider with legs of length 2. We give a polynomial-time algorithm for this case.

▶ **Theorem 3.** *For a given instance $(G, S, c, \mathcal{C}_0, \mathcal{C}_1)$, the matching-match puzzle can be solved in $O(n+m)$ time when G is a complete graph.*

Proof. We first check if G is a complete graph, and output "No" if it is not in $O(n+m)$ time. In the set of sticks in S, we can count the number of appearances of each color. We denote by $\#(i)$ the number of color i in S. Precisely, $\#(i) = \sum_{s=\{u,u'\}\in S}(\delta(\mathcal{C}_1(u) = i) + \delta(\mathcal{C}_1(u') = i))$ for each $i \in \{1, \dots, c\}$, where $\delta(\mathcal{C}_1(u) = i) = 1$ when $\mathcal{C}_1(u) = i$ and $\delta(\mathcal{C}_1(u) = i) = 0$ when $\mathcal{C}_1(u) \neq i$. Then, $\#(i)/(n-1)$ gives the number of vertices v in G with $\mathcal{C}_0(v) = i$ after matching by a solution $\mathcal{M}$. If $\mathcal{C}_0$ cannot satisfy this condition by extension to $\mathcal{C}_1$, the answer is "No". Thus we assume that the given function $\mathcal{C}_0$ in the instance is consistent with the condition. (Precisely, for each color i, the number of vertices with $\mathcal{C}_0(v) = i$ is equal to or less than $\#(i)/(n-1)$.) Now we assign the colors to the vertices v in V with $\mathcal{C}_0(v) = 0$ so that $\mathcal{C}_0(v) > 0$ and they are consistent with the condition. Since G is a complete graph, the assignment can be done according to the numbers $\#(i)/(n-1)$. After the assignment, we check the consistency of the sticks in S. Precisely, for each edge $\{u, u'\}$, we decrease $\#(\mathcal{C}_0(u))$ and $\#(\mathcal{C}_0(u'))$ by 1, respectively. Then the sticks are consistent if and only if $\#(\mathcal{C}_0(v)) = 0$ after the decreasements. Since all the vertices in G are now colored, it can be done in $O(m)$ time. The correctness of the algorithm is trivial. Thus we can solve it in $O(m)$ time in total when G is a complete graph. ◀

▶ **Theorem 4.** *For a given instance $(G, S, c, \mathcal{C}_0, \mathcal{C}_1)$, the matching-match puzzle can be solved in $O(n)$ time when G is a star.*

Proof. We first check if G is a star, and output "No" if it is not. It can be done in $O(n)$ time since G is a star if and only if it has one vertex of degree $n-1$ and $n-1$ vertices of degree 1. Assume $G = (V, E)$ is a star and $b \in V$ is the body vertex of degree $n-1$. Now we pick any stick $s = \{u, u'\}$ in S. Then it should be either $\mathcal{M}(u) = b$ or $\mathcal{M}(u') = b$ if it is a yes-instance. We first check whether $\mathcal{M}(u) = b$. In this case, all sticks $s' = \{w, w'\}$ should satisfy $\mathcal{C}_1(u) = \mathcal{C}_1(w)$ or $\mathcal{C}_1(u) = \mathcal{C}_1(w')$. If all other sticks satisfy the condition, we output "Yes". Otherwise, we check whether $\mathcal{M}(u') = u$ in the same way. If all other sticks satisfy the condition, we output "Yes", otherwise, output "No". The correctness of the algorithm is trivial, and it can be done in $O(n)$ time. ◀

Now we turn to the main theorem in this section:

▶ **Theorem 5.** *For a given instance $(G, S, c, \mathcal{C}_0, \mathcal{C}_1)$, the matching-match puzzle can be solved in polynomial time when G is a spider with legs of length 2, and $\mathcal{C}_0(v) = 0$ for all $v \in V$.*

Proof. Our algorithm checks all cases that $\mathcal{C}_0(b) = i$ with $i = 1, \dots, c$ for the body vertex b of G. Therefore, hereafter, we fix that $\mathcal{C}_0(b) = 1$ and $\mathcal{C}_0(v) = 0$ for all vertices $v \in V \setminus \{b\}$ without loss of generality. Then the set S of sticks can be partitioned into three subsets $S_1 = \{\{u, u'\} \mid \mathcal{C}_1(u) = \mathcal{C}_1(u') = 1\}$, $S_2 = \{\{u, u'\} \mid \mathcal{C}_1(u) = 1 \text{ and } \mathcal{C}_1(u') \neq 1\}$, and $S_3 = \{\{u, u'\} \mid \mathcal{C}_1(u) \neq 1 \text{ and } \mathcal{C}_1(u') \neq 1\}$ (Figure 4(a)).

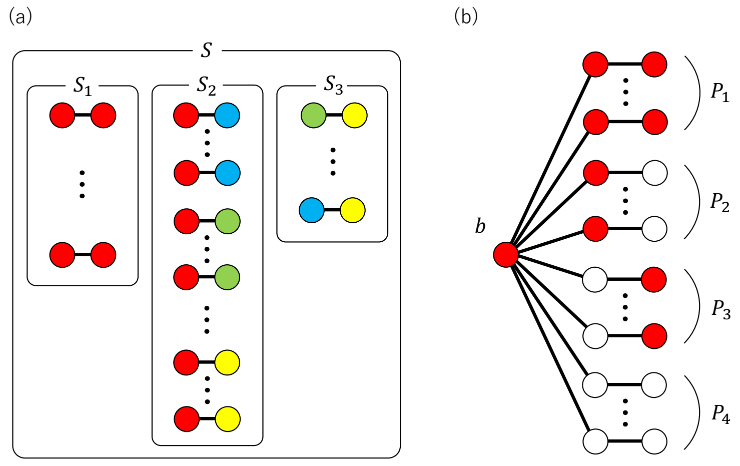

Figure 4 Classifications of S and V.

We suppose that there exists a feasible mapping $\mathcal{M}$ from S to V, and consider conditions that $\mathcal{M}$ has to satisfy. Here we color the graph G according to this mapping $\mathcal{M}$, and we suppose that each color of a vertex v in G can be referred as $\mathcal{C}(v)$ to simplify. Since G is a spider with legs of length 2, we have $n' = (n-1)/2$ legs. Then, we partition these legs (b, v_1, v_2) into four subsets $P_1 = \{(b, v_1, v_2) \mid \mathcal{C}(v_1) = \mathcal{C}(v_2) = 1\}$, $P_2 = \{(b, v_1, v_2) \mid \mathcal{C}(v_1) = 1 \text{ and } \mathcal{C}(v_2) \neq 1\}$, $P_3 = \{(b, v_1, v_2) \mid \mathcal{C}(v_1) \neq 1 \text{ and } \mathcal{C}(v_2) = 1\}$, and $P_4 = \{(b, v_1, v_2) \mid \mathcal{C}(v_1) \neq 1 \text{ and } \mathcal{C}(v_2) \neq 1\}$ (Figure 4(b)).

Then we can observe that

$$\begin{aligned} 2|P_1| + |P_2| &= |S_1|, \\ |P_2| + 2|P_3| + |P_4| &= |S_2|, \\ |P_4| &= |S_3|. \end{aligned}$$

Thus, we can obtain $|P_4| = |S_3|$ first. Now the size $|P_3|$ is one of $0, 1, \ldots, n'$. Therefore, our algorithm checks all cases for $|P_3| = 0, 1, \ldots, n'$. Once $|P_4|$ and $|P_3|$ are fixed, $|P_2|$ and $|P_1|$ are also determined (when one of them is negative, the case is not feasible). Therefore, our algorithm checks if there is a feasible mapping $\mathcal{M}$ from S to V for given $|P_1|$, $|P_2|$, $|P_3|$, and $|P_4|$. Intuitively, we have to make two matchings; one between S_2 and S_3 in P_4 and another one among S_2 in P_3. The remaining edges in S_2 can be matched in P_2 in any way.

Now we construct an auxiliary graph $H = (S_2 \cup S_3, E')$, where

$$E' = \{\{u, v\} \mid \text{two sticks } u \text{ and } v \text{ share a color } i \text{ in } \{2, 3, \ldots, c\}\}.$$

That is, two sticks in $S_2 \cup S_3$ are joined by an edge in E' if they have endpoints of the same color (under mapping $\mathcal{M}$) but not the color 1. Then we have the following observation:

▶ **Observation 6.** *A mapping $\mathcal{M}$ is a feasible solution of the given instance of the matching-match puzzle if and only if H has a matching M such that (1) M contains $|P_4|$ edges joining a vertex in S_2 and another one in S_3, (2) M contains $|P_3|$ edges joining two vertices in S_2, and (3) M contains no any other edges.*

Proof. We can observe that each edge in M corresponding to the central vertex v_1 in each path (b, v_1, v_2) on G. That is, edges in M with the condition (1) correspond to the edges in H joining an edge in S_2 of G and an edge in S_3 on a leg in P_4, and edges in M with the condition (2) correspond to the edges in H joining two edges in S_2 on a leg in P_3. The remaining edges in S_2 are joined to edges in S_1 on legs in P_2. ◀(of Observation 6)

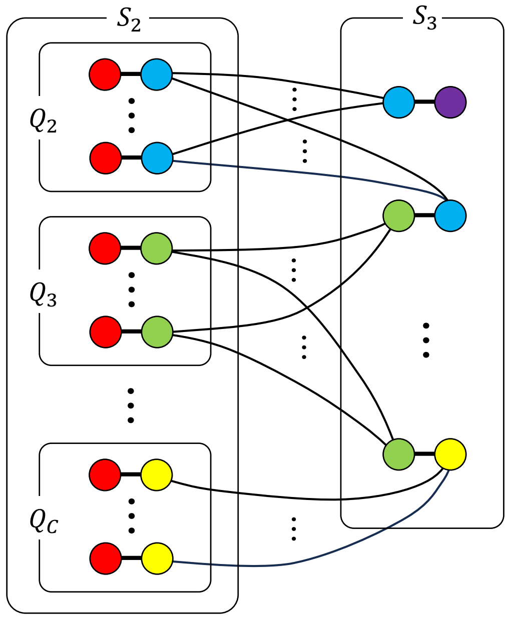

Figure 5 Graph $H' = (S_2 \cup S_3, E'')$.

By the observation, we can remove all edges in H that joins two vertices in S_3, which are not necessary to construct the desired matching M in H. Let $H' = (S_2 \cup S_3, E'')$ be the graph obtained from H by removing them (Figure 5).

Now our goal is a construction of the matching M on H' that satisfies the conditions of Observation 6. We first note that $H'[S_2]$ induces a set of cliques Q_i for each color $i = 2, 3, \ldots, c$. That is, S_2 can be partitioned into $c-1$ subsets $S_2^i = \{\{u, u'\} \mid \mathcal{C}_1(u) = 1 \text{ and } \mathcal{C}_1(u') = i\}$ for each color $i = 2, 3, \ldots, c$, and then $H'[S_2^i]$ is a clique Q_i. By the definition of H', we can observe that $N_{H'}[q] = N_{H'}[q']$ for any $q, q' \in Q_i$. We also note that $|M| = |S_3|$ and $|S_2| - |S_3| = |P_2|$. That is, in the spider G, every stick in S_3 should be joined to a stick in S_2 in a proper way and other sticks in S_2 are connected to sticks in S_1 in any way.

To construct M, we first compute a maximum matching M' on the bipartite graph $H'' = (S_2, S_3, E''')$, where E''' is the set of edges joining vertices in S_2 and S_3 in H'. (In other words, we ignore the edges in the cliques Q_i.) If $|M'| < |S_3|$, the answer is clearly "No". Otherwise, we choose $|S_3|$ edges in H' as the legs in P_4. Let Q'_i be the set of cliques induced by the vertices not matched in M'. If $\sum_{i=2}^{c} \lfloor |Q'_i|/2 \rfloor \geq |P_3|$, we have enough pairs to construct legs in P_3. Then we choose any $|P_3|$ edges in $H'[\cup_i Q'_i]$ and add them to M', which is the desired M. Once we can obtain M, we assign the remaining edges in S_2 as legs in P_2. In this case, the answer is "Yes".

The last remaining possible situation is that (1) the maximum matching M' has enough edges as $|M'| \geq |S_3|$ and (2) $\sum_{i=2}^{c} \lfloor |Q'_i|/2 \rfloor < |P_3|$. In this case, some Q'_i may contain an odd number of vertices not matched in M'. When $|Q'_i|$ and $\left|Q'_j\right|$ with $i \neq j$ are odd, by changing the edges in M', we may make both of $|Q'_i|$ and $\left|Q'_j\right|$ even and then we can increase $\sum_{i=2}^{c} \lfloor |Q'_i|/2 \rfloor$ by one. If we can perform it repeatedly, we may achieve $\sum_{i=2}^{c} \lfloor |Q'_i|/2 \rfloor = |P_3|$. This can be done if we have an *alternating path* P in H' with respect to M' between Q'_i and Q'_j. Here, an alternating path P is a path $(v_0, v_1, \ldots, v_{2k})$ for some positive integer k such that the edges on P are in M alternately. Thus, by finding an alternating path P in H' with respect to M' and replacing M' by $M' \oplus \mathsf{P}$ (swapping the members in M' according to P), we can increase $\sum_{i=2}^{c} \lfloor |Q'_i|/2 \rfloor$ by one.

For the graph $H' = (S_2 \cup S_3, E'')$ and the maximum matching M', we have the following lemma:

▶ **Lemma 7.** *We assume that $|Q'_i|$ and $\left|Q'_j\right|$ with $i \neq j$ are odd. Then there exists an alternating path between two vertices Q_i and Q_j if and only if a connected component of H' contains both of Q_i and Q_j.*

Proof (Outline). We first consider a vertex $s = \{u, w\}$ in S_3 with $\mathcal{C}_1(u) = \mathcal{C}_1(w)$. Then $N(s) = Q_{\mathcal{C}_1(u)}$. For such a vertex, M' contains an edge $\{s, q\}$ for some $q \in Q_{\mathcal{C}_1(u)}$ and we have nothing to do. Thus we focus on a vertex $s = \{u, w\}$ in S_3 with $\mathcal{C}_1(u) \neq \mathcal{C}_1(w)$. Then s has two clique neighbors Q_i and Q_j when $\mathcal{C}_1(u) = i$ and $\mathcal{C}_1(w) = j$. That is, $N(s) = Q_i \cup Q_j$.

Now we prove the claim of this lemma by induction on the length of the distance between Q_i and Q_j in a connected component of H'.

The base case is that the distance between Q_i and Q_j is 2. In this case, there exists a stick $s = \{u, w\}$ such that $\mathcal{C}_1(u) = i$ and $\mathcal{C}_1(w) = j$, and M' contains one of $\{s, q_i\}$ and $\{s, q_j\}$ for some vertices $q_i \in Q_i$ and $q_j \in Q_j$. Without loss of generality, we assume that M' contains $\{s, q_i\}$. Now, by assumption, we have an unmatched vertex q_j in Q_j. Then we can construct an alternating path is (q_i, s, q_j). In this case, we can replace M' with $M' \cup \{\{s, q_j\}\} \setminus \{\{s, q_i\}\}$. After replacement, both of $|Q'_i|$ and $\left|Q'_j\right|$ are even, and a new pair $\{q_i, q_{i'}\}$ with $q_i, q_{i'} \in Q'_i$ appears and it can be used to add a leg into P_3.

We turn to the inductive step: the distance between $q_i \in Q_i$ and $q_j \in Q_j$ is $2k$ for some $k > 1$ (and any pair of q_i and q_j). In a similar argument, we can assume that $\{s, q_i\}$ is in M' and $\{s, q_j\}$ is not in M' for some q_j in Q_j. Then, we can find a shortest path P of even length between q_i and q_j in H' when they are in the same connected component in H'. Since M' is a maximum matching, P is an alternating path and we can replace M' with $M' \oplus \mathsf{P}$. After replacement, we can see that $|Q'_i|$ and $\left|Q'_j\right|$ are even, and any other clique Q_k appearing on P does not change the parity of $|Q_k|$. Therefore, a new pair $\{q_i, q_{i'}\}$ with $q_i, q_{i'} \in Q'_i$ appears again and it can be used to add a leg into P_3.

When Q_i and Q_j are not in a connected component of H', it is trivial to see that we cannot make any alternating path joining two vertices in Q_i and Q_j, which completes the proof. ◀(of Lemma 7)

By Lemma 7, repeating the process, we eventually maximize $\sum_{i=2}^{c} \lfloor |Q'_i|/2 \rfloor$ for the maximum matching M'. When this maximum value $\sum_{i=2}^{c} \lfloor |Q'_i|/2 \rfloor$ is at least $|P_3|$, we can construct a desired matching M, which gives us a solution of the matching-match puzzle. On the other hand, when the maximum value is less than $|P_3|$, this case is not feasible.

The correctness of the algorithm follows with properties of the matroid. Using the standard technique for finding a maximum matching in a bipartite graph based on alternating paths (see, e.g., [5]), the algorithm can be performed in a polynomial time. ◀

Careful case analysis leads us to the following corollary:

▶ **Corollary 8.** *For a given instance $(G, S, c, \mathcal{C}_0, \mathcal{C}_1)$, the matching-match puzzle can be solved in polynomial time when G is a spider with legs of length* at most *2, and $\mathcal{C}_0(v) = 0$ for all $v \in V$.*

5 Concluding Remarks

In this paper, we introduced and investigated the matching-match puzzle in general form. It is based on a commercial product puzzle which can be modeled as a variant of the graph coloring problems in a natural way.

In the puzzle, the numbers of vertices and edges in the graph and the number of colors are variables and it is NP-complete even if the graph is quite restricted. In the proof of NP-completeness, the number of the colors is linear to the number of the vertices. It may be a reasonable assumption that the number of the colors is bounded above by a constant (in fact, the commercial product has 4 colors). For example, a natural brute force algorithm allows us to solve the matching-match puzzle on a k-partite complete graph $G = (V_1, V_2, \ldots, V_k, E)$ in $O\left(\frac{c(c+n)^{k(c-1)}}{(c!)^k}\right)$ time. It is another open problem that the puzzle is fixed-parameter tractable with respect to the number of colors. That is, is there an algorithm that runs in $O(f(c) \cdot poly(n))$ for some graph classes of graphs of n vertices with c color, where $poly(n)$ is a polynomial function and $f(c)$ is a computable function?

A natural extension of Corollary 8 is the case that some vertices of a spider G of legs of length at most 2 are pre-colored additional to the body. We consider our polynomial-time algorithm can be extended to this case, however, case-analysis is quite complicated. (We note that it can be solved in $O\left(\frac{(c(c+n)^{c^2})}{(c!)^c}\right)$ time by a brute force algorithm.) It is unlikely that there exists a simple polynomial-time algorithm for solving this case. Based on this fact, we conjecture that the matching-match puzzle is NP-complete if G is a spider with legs of length at most 3, which is another open problem.

References

1 N. Alon, R. Yuster, and U. Zwick. Color-Coding. *Journal of the ACM*, 42(4):844–856, 1995.

2 Michael R. Garey and David S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness.* Freeman, 1979.

3 R. A. Hearn and E. D. Demaine. *Games, Puzzles, and Computation.* A K Peters Ltd., 2009.

4 Tommy R. Jensen. *Graph Coloring Problems.* John Wiley & Sons, 1994.

5 C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization.* Dover, 1982.

6 Ryuhei UEHARA. Computational Complexity of Puzzles and Related Topics. *Interdisciplinary Information Sciences*, ID 2022.R.06:1–23, 2023. doi:10.4036/iis.2022.R.06.