2020

Connectiveity and Structures of Coloring Graphs

Xin Yutong

University of Richmond

Connectivity and Structures of Coloring Graphs

Yutong Xin
Honors Thesis[*]
Department of Computer Science
University of Richmond
April 27, 2020

[*]Under direction of Dr. Prateek Bhakta

The signatures below, by the thesis advisor, the departmental reader, and the honors coordinator for mathematics, certify that this thesis, prepared by Yutong Xin, has been approved, as to style and content.

Prateek Bhakta

(Dr. Prateek Bhakta, Thesis Advisor)

James A Davis

(Dr. James Davis, Departmental Reader)

Jory Denny

(Dr. Jory Denny, Honors Coordinator)

Thesis advisor: Professor Dr. Prateek Bhakta Yutong Xin

Connectivity and Structures of Coloring Graphs

Abstract

Reconfiguration problems have been studied and applied to solve problems in various areas, including in Math, Computer science, and Chemistry. Due to the close relatedness between coloring graphs and reconfiguration problem, the connectivity and structure of coloring graphs give valuable information to a solution set of the corresponding reconfiguration problems. In this paper we will discuss $2-$connectedness and cut-vertices of coloring graphs, and forbidden structures on a coloring graph with cut-vertices.

Contents

1
Introduction

Proper colorings of graphs are studied in many contexts in mathematics and computer science, including in geometry[9], linear programming[5], constraint satisfaction problem[7], and others. A proper $k-$coloring of a graph G is an assignment of k colors to all vertices of G such that no neighboring vertices of G share the same color. On the other hand, a proper $k-$coloring graph $C_k(G)$ of the base graph G is a graph of all such proper $k-$colorings.

Each vertex of $C_k(G)$ is a proper $k-$coloring of G, while each edge between two vertices of $C_k(G)$ suggests the two coloring differ by the color of exactly one vertex of G

A coloring graph of a base graph can be viewed as a reconfiguration graph of a given problem. Reconfiguration problems has been studied in various areas, including in computer science, mathematics, and physics[3], and such problems include problems of vertex cover, clique, independent set, matching problem, power supply, etc.[8] Reconfiguration problems arise every times there is a step-by-step transition between two solutions in the solution set to the given problem[8]. Given finding a proper k-coloring of G as a reconfiguration problem, we can view the coloring graph $C_k(G)$ as the corresponding reconfiguration graph. Each proper $k-$coloring, or a vertex of $C_k(G)$, is a solution, and each edge of $C_k(G)$ is a one-step transition between two solutions. It follows that the connectivity of $C_k(G)$ is suggestive of the flexibility of traversing the solution set. The connectivity of the above reconfiguration graph has been of interest in the fields of chemistry and knot theory[3].

Cereceda, Heuvel, and Johnson explored the relationship between the chromatic number of a base graph and the connectivity of its corresponding coloring graph[6], and later Bhakta et al. went further to study the bi-connectivity of the coloring graph[4]. As a different perspective for understanding coloring graphs better, Beier et al. made a classification of graphs that are possible to become coloring graphs, and forbidden induced subgraphs of coloring graphs[3], and as an expansion of that, Alvarado et al., constructed structures of forbidden minimal induced subgraphs for coloring graphs[1].

Although the coloring graph is useful, there are many challenges we face in studying this object. First of all, the number of proper k-colorings for a given graph G can be very large compared to the number of vertices of G. The direct result of the large number of colorings is a coloring graph with an exponentially large number of vertices and edges. Being able to get a completed coloring graph is desirable, as we would have the ability of studying the connectivity of the coloring graph both locally and globally. However, it can be realistically impossible to draw a completed coloring graph for a even moderately sized base graph. As such a task is realistically almost impossible to achieve, the need of borrowing the power of programming tools arose. Yet even with programming, the challenges still stay. The large number of vertices and edges could put pressure on the computer, and it wasn't very hard to run out of memory or time during graph generation or exploration in our tests. Hence, the data structures and algorithm implementations used need to be considered with great care.

We will start to discuss more details about our work in the next few chapters. We'll introduce the relevant definitions, algorithm and lemma in the preliminaries chapter, and then proceed to introduce the first conjecture we had, as well as the program we built to assist our work. In chapter 3 we'll give a detailed explanation of the Tarjan's algorithm, a core algorithm we used in our program. In the following chapter 4 we will present the counterexample we found against our initial conjecture, after which we slightly switched the direction to study the forbidden structures in a coloring graph with cut-vertices. In chapter 5 we will give two proofs about the nonexistence of small odd cycles containing a cut-vertex in a polyp. Lastly, we will talk about what the directions we can take from the results we have at current stage.

2 Preliminaries

We begin by formalizing some of the concepts we will be discussing for the remaining of the paper.

2.1 Graph and Graph Coloring

Definition 2.1.1. A **graph** G is a pair $G = (V, E)$, where V is a set of vertices, and E is a set of edges. For each edge $e \in E$, $e = (u, v)$, $u, v \in V$.

Definition 2.1.2. A proper **k-coloring** of a graph G is an assignment of k colors to all vertices of G, such that no neighboring vertices in G are assigned the same color.

Definition 2.1.3. A **chromatic number** of a graph G, $\chi(G)$, is the minimum number of colors necessary to have a proper coloring of G.

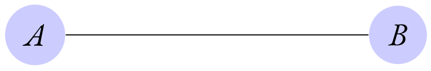


Figure 2.1: a simple graph G

The above Figure 2.1 is a simple graph G with 2 vertices and 1 edge. We can assign color c_0 to A, c_1 to B, which is a proper 2−coloring of G. Also, since there is no way to color G with only one color, the chromatic number of G is 2. One famous result of the 4−color map theorem states all planar graph are 4−colorable.[2]

2.2 Connectivity and Cut-vertices

Another important focus of our study is the connectivity of coloring graphs as mentioned in the first section. In order to take a closer look, let's understand connectivity in a more formal manner.

Definition 2.2.1. A graph G is **connected**, or **1-connected** if there exists at least one path between any two vertices in G.

Definition 2.2.2. A Graph G is **2-connected**, or **bi-connected** if after removing any one vertex and its incident edges from G, the remaining graph is still connected. Alternatively, G is bi-connected if there exist at least two paths between any two vertices in G.

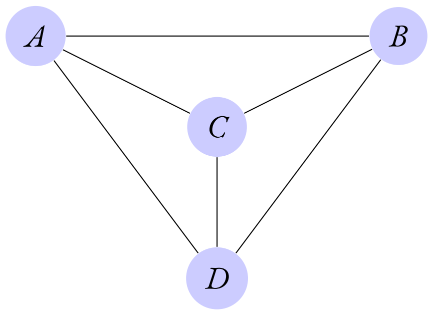
Definition 2.2.3. A connected component in a graph G is a subgraph H of G, such that all vertices in H are connected with each other through at least one path, and no vertices in H is connected to vertex in $G \setminus H$.

Now with the definition of connectivity in mind, we can introduce the concept of cut-set and cut-vertex.

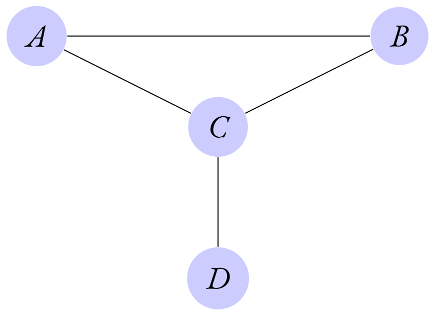
Definition 2.2.4. A vertex **cut-set** in a graph G is a set of vertices of G, the removal of which increases the number of connected components of G.

Definition 2.2.5. A vertex v in a graph G is a **cut-vertex**, if removing v creates more connected components in G. In other words, v is a cut vertex if it is a cut-set of size 1.

In the figure below, (a) is a 2-connected, or bi-connected graph, as the graph stays connected after removing any vertex. On the other hand, (b) is a connected but not 2-connected graph, as removing vertex C disconnects the graph. C is a cut-vertex in this graph.



(a) a 2-connected graph



(b) a connected but not 2-connected graph

Figure 2.2

Definition 2.2.6. A **local cut** vertex, or a **second-degree cut** vertex is a member of a cut set of size 2 but not a member of cut set of size 1.

Definition 2.2.7. A **k**$^{\textbf{th}}$ **degree cut** vertex, is a member of a cut set of size k but not a member of cut set of size $k-1$.

One focus of our work was on the connectivity of coloring graphs. To be able to compute the $2-$connectedness of coloring graphs, we built up a program that implemented a modified version of Tarjan's algorithm, which can find cut vertices of a graph efficiently. Tarjan's algorithm is based on depth first search, with time and space complexity being $O(|V| + |E|)$. It takes an undirected graph G as input, and outputs the cut-vertices of G.

2.3 Fundamentals of Coloring Graphs

We have introduced all necessary concepts for this paper related to coloring and connectivity, and let us shift our focus to the definition of coloring graph.

Definition 2.3.1. Given a base graph G, the k coloring graph $C_k(G)$ of G is a graph with vertex set $V(C_k(G))$ and edge set $E(C_k(G))$ where

- the vertex set $V(C_k(G))$ is composed of all proper $k-$colorings of G
- the edge set $E(C_k(G)) = \{e = (\alpha, \beta) \mid \exists u \in V_G \forall v \in V_G, \alpha(u) \neq \beta(v) \iff u = v\}$. In other words, there is an edge between two colorings α, β if α and β only differ at one vertex.

With the definition of coloring graph in mind, we can relate to the previous discussed concepts of cut-vertex and connectivity, and understand them in regards to coloring graphs.

Definition 2.3.2. A **k-cut-coloring** of the graph G is a cut vertex of $C_k(G)$.

Definition 2.3.3. A base graph G is **k-mixing** if its coloring graph $C_k(G)$ is connected.

Definition 2.3.4. A **block-cut tree**, or a **metagraph** of a graph G is a tree generated by decomposing G into a tree. Each node of the tree is either a $2-$connected component of G, or a cut-vertex.

2.4 Structure of Coloring Graphs

Since in a lot of cases, the coloring graph have large vertex and edge sets, and in turn also many connected components. More than often, for a connected coloring graph, the metagraph of the connected components is symmetric. The metagraph has a central component, from where many symmetric offshoots branch out. We define the central component as the mothership, and the offshoots as the polyps.

Let $\mathcal{C}$ be the set of $2-$connected components in a coloring graph $C_k(G)$, where G is the base graph with n vertices.

Definition 2.4.1. A **small mothership** $M \in \mathcal{C}$ is the maximum $2-$connected component in $\mathcal{C}$ such that for all color permutations $\sigma \in S_n$, and for all colorings $\in V_M$, $\sigma(c) \in V_M$.

Definition 2.4.2. A **large mothership** $L \in \mathcal{C}$ is a $2-$connected component in $\mathcal{C}$ such that there exists a color permutation $\sigma \in S_n$, such that for all colorings $\in V_L$, $\sigma(c) \in V_L$.

Definition 2.4.3. A **polyp** $P \in \mathcal{C}$ is a $2-$connected component that is not the mothership.

Another differentiation important to make is between a vertex that cannot change color in the base graph and a coloring vertex in the coloring graph where the coloring assignment to all base graph vertices is fixed. We introduce the concepts of a locked vertex and a frozen coloring.

Definition 2.4.4. A vertex v in a base graph G is **locked** if, for all colorings $\alpha, \beta \in C$, where C is a bi-connected component of the coloring graph $C_k(G)$, $\alpha(v) = \beta(v)$.

Definition 2.4.5. Given a base graph G, a $k-$coloring of G α is a **frozen coloring** if none of the vertices in G can change color from the assignment of $\alpha(G)$. In the coloring graph $C_k(G)$, $\alpha \in V_{C_k(G)}$ has no incident edge.

Another focus of our work was on exploring the structure of coloring graphs. Specifically, we were working towards finding and classifying forbidden structures of coloring graphs. In chapter 5 we will show 2 proofs about the non-existence of small odd-cycles containing a cut-vertex in the coloring graph. The proofs used a previous result by Bhakta, et. al[4].

Lemma 2.4.6. *Let α be a $k-$cut coloring of a graph G. There exists a unique color c such that every neighbor of α in $C_k(G)$ is obtained from α by recoloring some vertex to color c.*

3 Tarjan's algorithm modified

We formed a conjecture whose details will be discussed in the next chapter. In the process of trying to disprove the conjecture, we developed a program to aid in our search, due to the large number of proper colorings of a base graph.

There were 2 main tasks this program needed to achieve. The first step was to generate the coloring graph $G_k(G)$ given a base graph G and number of colors k. We apply the backtrack approach to complete the first task. We generate all possible colorings given a base graph using backtracking search, and form the coloring graph $C_k(G)$ as each new proper coloring is discovered. The second step was to output the connectivity of the coloring graph. We implemented Tarjan's algorithm to achieve this task. After the coloring graph $C_k(G)$ is generated, we feed $C_k(G)$ to Tarjan's algorithm. If Tarjan's terminates before all the vertices are visited, $C_k(G)$ is then disconnected. On the other hand, if the graph is connected, and Tarjan's finds a set of cut-vertices, $C_k(G)$ is then connected but not $2-$connected. In this case, our algorithm outputs the size of the cut-vertices; if Tarjan's finds no cut-vertex, $C_k(G)$ is then $2-$connected. Now we will give a more detailed description of Tarjan's algorithm.

Algorithm 1 tarjanHelper

```
Input: Graph $G$
Output: List $cutVertices$
1: Declare $cutVertices \leftarrow \emptyset$,
2: for all Vertex $v$ : $G.vertexList$ do
3:     $rootNumOfChildren \leftarrow 0$
4:     $currTime \leftarrow 0$
5:     if $\neg v$.isVisited then
6:         $v$.isRoot = $true$
7:         $currTime = 0$
8:         tarjan($G, v, cutVertices$)
9:         if $rootNumOfChildren \geq 2$ then
10:            $cutVertices$.add($v$)
11: return $cutVertices$
```

Algorithm 2 tarjan

```
Input: Graph $G$, Vertex $v$, List $cutVertices$
 1: $v$.visitedTime = $currTime$, $v$.lowTime = $currTime$
 2: $currTime++$
 3: $v$.visited = $true$
 4: for all Vertex $u$ : $v$.neighbors do
 5:     if $u \neq v$.parent then
 6:         if $\neg u$.visited then
 7:             if $v$.isRoot then
 8:                 $rootNumOfChildren++$
 9:             $u$.parent = $v$
10:             tarjan($G, u, cutVertices$)
11:         else
12:             if $v$.visitedTime $\leq u$.lowTime $\cap \neg v$.isRoot then
13:                 $cutVertices$.add($v$)
14:     $v$.lowTime = Math.min($u$.lowTime, $v$.visitedTime)
```

The algorithm takes an undirected graph G as input. Each vertex v in the graph has the following extra fields: **visitedTime**, **lowTime**, **parent**, **visited**. The algorithm also has global variables: **currTime**, and **rootNumOfChildren**.

- global variables:
  - **currTime**: gets updated every time a vertex is visited
  - **rootNumOfChildren**: keeps track of how many children get explored from root as parent
- instance variables of Vertex:
  - **visitedTime**: equals the **currTime** when v gets visited.
  - **lowTime**: keeps track of the **visitedTime** of vertex v_0, where v_0 has mininum **visitedTime** among all the vertices reacheable from v during the search.

- **parent**: records the vertex through which v is visited.
- **visited**: records whether v is visited.

In this algorithm, we have 2 criteria for checking whether a vertex is a cut-vertex: v is a cut vertex if

- v is a root, and $\mathit{rootNumOfChildren} \geq 2$
- v is not a root, and $v.\texttt{visitedTime} \leq u.\texttt{lowTime}$, where u is a non-parent neighbor of v

Assume v is a root vertex. Assume v is in or adjacent to n $2-$connected components C_i, and $\forall C_i,\ v$ is adjacent to a set of $U_i = \{u_i \in C \,|\, (u_i, v) \in E_G\}$ vertices. Then according to the algorithm, for each U_i, there only exists one $u_0 \in U_i$, such that $u_0.\texttt{parent}$ is v, and all other $u' \in U_i$ are not visited immediately from v. Since `rootNumOfChildren` gets incremented every time a neighbor u is visited within the call on the root vertex v, we have $\texttt{rootNumOfChildren} = n$. Thus, if $\mathit{rootNumOfChildren} \geq 2$, v connects more than 1 $2-$connected components, and thus is a cut-vertex.

Note what line 14 in `tarjan` does is to update the `lowTime` of each vertex v, such that $v.\texttt{lowTime}$ always keeps track of the minimum `visitedTime` of the vertices in C, where C is a currently being explored $2-$connected component containing v. Now assume v is not a root vertex. If $v.\texttt{visitedTime} < u.\texttt{lowTime}$ ($v.\texttt{visitedTime}$ is smaller than the minimum `visitedTime` associated with the $2-$connected component containing u), u and v are not in the same $2-$connected component. Thus, v is a cut-vertex; if $v.\texttt{visitedTime} = u.\texttt{lowTime}$, v, u are in the same $2-$connected component C, and v is visited before all other vertices in C are visited. However, since v is not the root vertex, there must exists a vertex w, a neighbor of v, visited before v. Thus $w.\texttt{visitedTime} < v.\texttt{visitedTime}$, and hence $w \notin C$. Therefore, v is a cut-vertex.

In the process of building up our program as planned, we came across several problems.

The first problem arose when trying to generate the coloring graph $C_k(G)$ of the base graph G. Since each vertex of $C_k(G)$ (CVertex) is a coloring of G, we stored each colored base graph as a field in each CVertex. Additionally, a base graph G can have a lot of proper colorings, forcing us to store a lot of colored graph objects. Hence, when trying to store too many CVertices, we faced OutOfMemoryError.

The second and third problems occurred during Tarjan's algorithm. Tarjan's came naturally as a recursive algorithm. Again due to the large number of CVertices, we hit recursive depth limit during the depth-first search. Another problem was each CVertex object stored a list of CVertex neighbors. Due to the large number of CVertices and the number of neighboring CVertices, the space needed was still huge. Hence another OutOfMemoryError occurred.

We modified the original code to solve the problems one by one.

For the first OutOfMemoryError, since storing a colored graph object in each CVertex took excessive memory, we figured out a new way to represent each coloring. We each colored graph object with an **long** type encoding. We treated the encoding as a number in base k. Each vertex in the base graph corresponds with a digit, and the range of value in each digit is $\{0, 1, \cdots, k-1\}$. This replacement significantly cut of the memory usage, so the first problem was solved.

For the recursive depth limit problem, we needed to change recursion to iteration. We achieved this change by using a Stack.

The last OutOfMemoryError persistently occurred even after we changed the representation of the coloring in each CVertex. Indeed, even though the memory of the coloring representation field was reduced, the neighbors list field still took a large space. Deciding it wasn't really possible for us to reduce any other data fields in CVertex, we needed to cut off the neighbor list. However, each CVertex still needed to have knowledge about its neighbors. Our solution was to generate neighbors of each CVertex in real-time. In real-time we, based on the encoding of the CVertex itself, generated the encoding of its neighbors. Since we needed to know when all neighbors of a CVertex were explored, we included a neighborTrack field to each CVertex. Each time the encoding of a potential neighbor was generated, we increment the neighborTrack. We stopped when neighborTrack reaches the maximum number of potential neighbors one CVertex could have($(k-1)^{|V_G|}$). By the above modification, our last problem was solved.

4
Inital result: a counterexample

Our initial conjecture 4.1 was inspired by a previous known result about the connectivity of coloring graph, which relates to the degeneracy number of a graph. The degeneracy number of a graph G is the smallest number k, such that for each subgraph H of G, all vertices of H have at most degree k. The previous results states: for a base graph G with degeneracy number d, the corresponding coloring graph $C_k(G)$ is connected when $k \geq d + 2$, and $2-$connected when $k \geq d+3$[4]. Inspired by this result, we came up with the following conjecture:

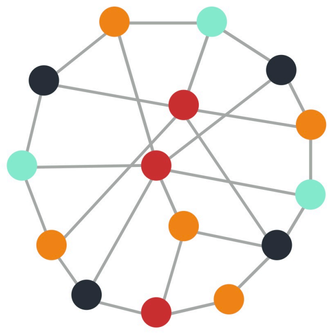
Conjecture 4.1. Given a base graph G and a number of colors i, if $\forall k \geq i$, the coloring graph $C_k(G)$ is connected, then $\forall k \geq i+1$, $C_k(G)$ is 2-connected.

Our plan was to first look for counterexamples against our conjecture, and, if we failed to find any, then continue to prove the conjecture. With our program, we are able to input any given base graph with reasonable size and number of colors to the program, and compute whether the coloring graph is $2-$connected. To make this searching process more efficiently, we decided to generate random graphs as input to our program, and output the graphs whose coloring graph had cut-vertices with $k=4$.

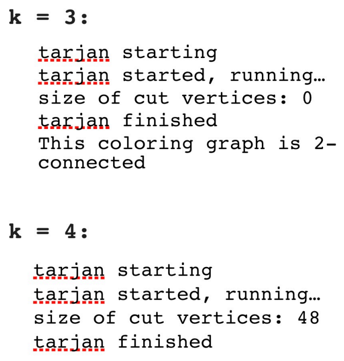
Given a fixed number of vertices, one approach to generate random graphs was to randomly generate edges between all pairs of vertices with a probability p. However, considering the strict structural nature of graphs with cut-colorings, such an approach had a low chance of finding a desirable base graph. Therefore, we choose another approach. Given a fixed number of vertices n,

- initialize an empty graph G with n vertices
- generate all possible $\binom{n}{2}$ edges, and shuffle the edges to get a randomly ordered edge set S_e
- for each $e \in S$:
  - add e to G
  - compute connectivity of $C_4(G)$, and record G if $C_4(G)$ has cut-vertices

After one time of such searches on $n = 7, \cdots, 11$, we found in total 31 base graphs whose coloring graph had cut-vertices with $k = 4$. However, none of the graphs had connected coloring graph with $k = 3$. However, we were able to get more insights about the structure of these base graphs whose coloring graph had cut-vertices. With the help of such insights, we created a base graph with our programming tool by hand, shown in Figure 4.1a, whose coloring graph was connected with $k = 3$ colors.



(a) Counterexample - cut-coloring of a base graph, k = 4



```
k = 3:

   tarjan starting
   tarjan started, running…
   size of cut vertices: 0
   tarjan finished
   This coloring graph is 2-
   connected

k = 4:

   tarjan starting
   tarjan started, running…
   size of cut vertices: 48
   tarjan finished
```

(b) Results from Tarjan's Algorithm when k = 3, 4

Figure 4.1

After inputting the graph in 4.1a, we got the results listed in 4.1b. Our result showed when $k = 3$, $C_k(G) = C_3(G)$ is connected. Since the degeneracy number d of G is $d = 2$, from the precious result[6], we could conclude $\forall k \geq 4$, $C_k(G)$ is connected. Combing the above results, $\forall k \geq 3$, $C_k(G)$ is connected. If our conjecture was true, $C_{k+1}(G) = C_4(G)$ must be $2-$connected. However, as shown in the result 4.1b, $C_4(G)$ contains 48 cut-vertices, and hence is not $2-$connected. Therefore, our conjecture was proven false.

5
Forbidden structures in a polyp

Among coloring graphs we found that are not bi-connected, it seems the polyps of the coloring graphs are always bipartite. If that is true, there cannot exist any odd cycles in a polyp. In fact, we did prove the non-existence of 3 and $5-$cycles that contains a cut-vertex.

Proposition 5.1. *In a connected but not bi-connected graph, a* $3-$*cycle containing a cut-vertex cannot exist.*

Proof. Let α be a $k-$cut coloring of a graph G. Suppose a cycle containing α exists in the polyp. Let $\beta, \gamma, \beta \neq \gamma$ be the other two colorings in the $3-$cycle, and there are edges between (α, β), (β, γ), and (γ, α).

Let $v \in V_G$ be the vertex that changes color from $\alpha(G)$ to $\beta(G)$, and $w \in V_G$ be the vertex that changes color from $\alpha(G)$ to $\gamma(G)$. Since $\beta \neq \gamma$, $v \neq w$.

Note $\alpha(v) \neq \beta(v), \alpha(w) = \beta(w)$, and $\alpha(v) = \gamma(v), \alpha(w) \neq \gamma(w)$. Therefore $\beta(v) \neq \gamma(v), \beta(w) \neq \gamma(w)$, and thus β and γ cannot be neighbors, where we reach a contradiction.

□

Proposition 5.2. *In a connected but not bi-connected graph, a* $5-$*cycle containing a cut-vertex cannot exist.*

Proof. Suppose a 5-cycle exists in the polyp.

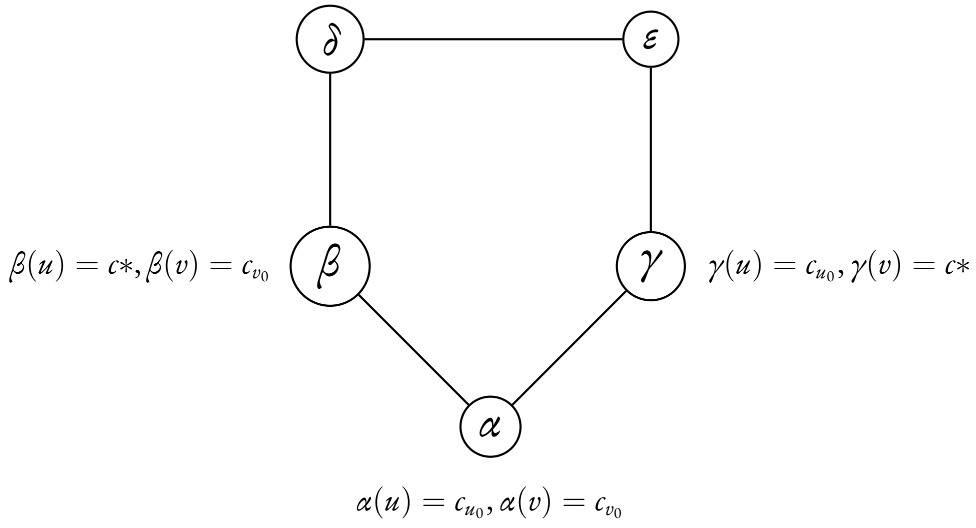
Let $C_k(G)$ be the coloring graph of G with k colors. Let α be a cut vertex in $C_k(G)$, and the 5-cycle is formed by $\alpha - \beta - \delta - \varepsilon - \gamma - \alpha$, as shown in Figure 5.1.

Let $u, v, u \neq v$, be the two vertices in G such that the recoloring of u, v forms the edge (α, β), (α, γ), respectively, in $C_k(G)$. Since α is a cut-vertex, the color u and v changes to is the same by Lemma 2.4.6. Denote that color as $c*$.

Denote the color of u as c_{u_0}, the color of v as c_{v_0} in α. We then have $\beta(u) = c*, \beta(v) = c_{v_0}$, while $\gamma(u) = c_{u_0}, \gamma(v) = c*$. Since β and γ are 3 edges away, and they have 2 vertices, u, v, in G colored differently. Hence, 2 out of the 3 edges must correspond with the color change of u, v. Therefore, it is impossible for a vertex other than u, v to change color along the path

$\beta - \delta - \varepsilon - \gamma$, since it would have to change back to its original color, which would take 2 edges and result in at least $4 > 3$ edges between γ and β.

Thus, it must be one of u or v that changes to another color different from $c*$ and its original color in α. Without loss of generality, say u is the vertex that changes color twice, and denote the second color it changes to as $c_{u_1} \neq c*, c_{a_0}$. There can be two cases for this second color change to occur:



δ
ε
$\beta(u) = c*, \beta(v) = c_{v_0}$
β
γ
$\gamma(u) = c_{u_0}, \gamma(v) = c*$
α
$\alpha(u) = c_{u_0}, \alpha(v) = c_{v_0}$

Figure 5.1

- Case 1: u changes color from β to δ. In this case, $\delta(u) = c_{u_1}, \delta(v) = c_{v_0}$. α and δ should then be adjacent, as they are different only by the color of u. This implies recoloring u from c_{u_0} to c_{u_1} would obtain δ from α. However, since α is a cut vertex, and $c* \neq c_{a_1}$, this violates Lemma 2. Hence case 1 is not possible.

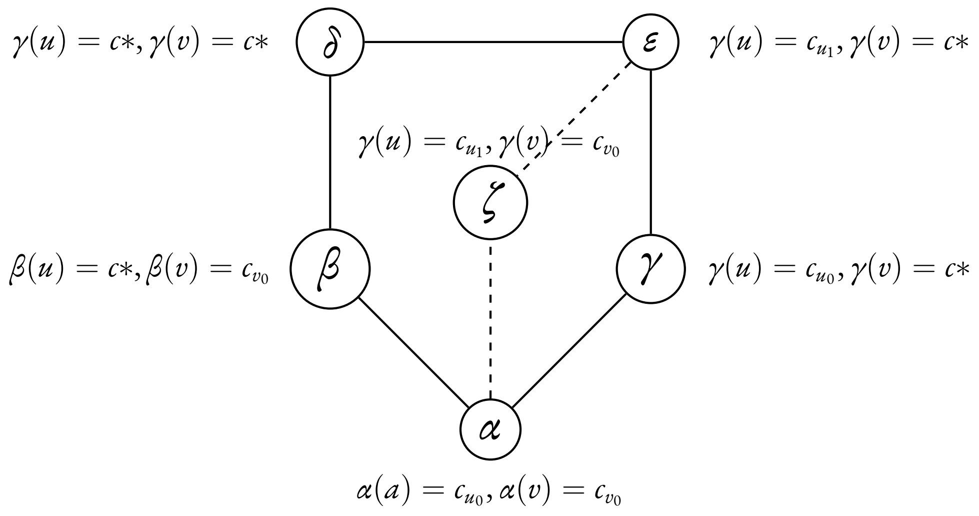

$\gamma(u) = c*, \gamma(v) = c*$
δ
ε
$\gamma(u) = c_{u_1}, \gamma(v) = c*$
$\gamma(u) = c_{u_1}, \gamma(v) = c_{v_0}$
ζ
$\beta(u) = c*, \beta(v) = c_{v_0}$
β
γ
$\gamma(u) = c_{u_0}, \gamma(v) = c*$
α
$\alpha(a) = c_{u_0}, \alpha(v) = c_{v_0}$

Figure 5.2

- Case 2: u does not change color from β to δ, but does change color from δ to ε. Hence, $\varepsilon(u) = c_{u_1}, \varepsilon(v) = c*$.

  Note β and ε have 2 vertices in G, u, v colored differently. Since β and ε are 2 edges away, each edge must correspond the color change of u, v. By our case assumption, u does not change color from β to δ, so $\delta(u) = c*$, $\delta(v) = c*$. Since u and v have the same color here, they cannot be adjacent.

  Note α and ε have both u, v colored differently. Because u, v are not neighbors, nothing forbids us from first recoloring u from c_{u_0} to c_{u_1}, and then recoloring v from c_{b_0} to $c*$ to obtain ε from α. However, since $c_{u_1} \neq c*$, this is a violation to Lemma 2.4.6, so case 2 is not possible as well.

Since neither cases can happen, the initial assumption must be false. Hence we've proved by contradiction that 5−cycle that includes a cut-vertex cannot exist in the polyp. □

6 Future directions

Based on our results, we want to point to some future directions worth exploring.

Although our conjecture 4.1 was proven false, we have two related conjectures to explore, one is a modified version of 4.1, while the other one relates to the structure of base graphs.

Our conjecture was disproven because the coloring graph, connected for k number of colors, failed to be $2-$connected when we increased the number of colors by 1 to use $k + 1$ colors. But the conjecture might still apply for $k + i$ colors, $i \geq 2$. In order words, the following modified version of our original conjecture still awaits to be proven or disproven:

Conjecture 6.1. Given a base graph G, if $C_k(G)$ is connected, $C_{k+i}(G)$ is $2-$connected for all $i \geq 2$.

On the other hand, it is also worthwhile to consider the structure of the base graph. Our conjecture did fail in the counterexample we found, but there are base graphs where the conjecture hold. We are curious to see if we can identify sufficient criteria on the base graph for which our conjecture does hold, and also criteria for which it doesn't hold.

The nonexistence 3 and 5-cycle proofs also gave us new potential directions to explore on. One direction is to try to generate the results on the non-existence of all odd cycles containing a cut vertex in a polyp. If that is achieved, we may even further generalize to try to prove whether a polyp cannot have any odd cycles, regardless of whether the cycles contain a cut-vertex. Since our $3-$ and $5-$cycle proofs are motivated by the hypothesis that all polyps are bipartite, we can also continue on this hypothesis and try to prove or disprove it.

References

[1] Alvarado, F., Butts, A., Farquhar, L., & Russell, H. M. (2018). Forbidden subgraphs of coloring graphs. *Involve*, 11(2), 311–324.

[2] Appel, K. & Haken, W. (1977). Every planar map is four colorable. part i: Discharging. *Illinois J. Math.*, 21(3), 429–490.

[3] Beier, J., Fierson, J., Haas, R., Russell, H. M., & Shavo, K. (2016). Classifying coloring graphs. *Discrete Math.*, 339(8), 2100–2112.

[4] Bhakta, P., Buckner, B. B., Farquhar, L., Kamat, V., Krehbiel, S., & Russell, H. M. (2019). Cut-colorings in coloring graphs. *Graph. Comb.*, 35(1), 239–248.

[5] Borodin, O., Ivanova, A., Montassier, M., & Raspaud, A. (2011). (k,j)-coloring of sparse graphs. *Discrete Applied Mathematics*, 159(17), 1947 – 1953.

[6] Cereceda, L., Van Den Heuvel, J., & Johnson, M. (2008). Connectedness of the graph of vertex-colourings. *Discrete Math.*, 308(5–6), 913–919.

[7] Gualandi, S. & Malucelli, F. (2012). Exact solution of graph coloring problems via constraint programming and column generation. *INFORMS Journal on Computing*, 24(1), 81–100.

[8] Ito, T., Demaine, E. D., Harvey, N. J., Papadimitriou, C. H., Sideri, M., Uehara, R., & Uno, Y. (2011). On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12), 1054 – 1065.

[9] Wakabayashi, Y. (2019). Spin networks, ehrhart quasipolynomials, and combinatorics of dormant indigenous bundles. *Kyoto J. Math.*, 59(3), 649–684.