Conflict-Free Vertex Coloring Of Planar Graphs

Shawn Seymour
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
seymo079@morris.umn.edu

ABSTRACT

The conflict-free coloring problem is a variation of the vertex coloring problem, a classical NP-hard optimization problem. The conflict-free coloring problem aims to color a possibly proper subset of vertices such that there is a unique color within the closed neighborhood (a vertex and its neighbors) of every vertex. This paper presents recent findings and heuristics to solve the conflict-free coloring problem on both general graphs and planar graphs.

1. INTRODUCTION

Consider the map of the 48 contiguous states in the United States of America. Suppose we would like to color each state such that no two states that share a boundary have the same color. This problem can be modeled with a graph. We can represent each state with a *vertex* and represent a boundary between two states with an *edge*.

This is a famous example of the *vertex coloring* problem and one of many graph coloring problems. The vertex coloring problem aims to find the minimum number of colors needed to color a graph such that no two adjacent vertices are colored with the same color. While some problems are relatively easy to solve, the vertex coloring problem is one of the most computationally complex problems in computer science and mathematics [3]. The vertex coloring problem has many real-world applications such as finding the minimum number of time slots needed to schedule final exam periods such that no two courses (taken by the same student) are scheduled at the same time slot.

The *conflict-free coloring* problem is a relaxed variation of the vertex coloring problem. The conflict-free coloring problem does not aim to color *every* vertex. Rather, it aims to color *some* vertices such that the neighborhood of every vertex contains at least one uniquely colored vertex.

This paper looks into some applications and heuristics of conflict-free coloring. We then look into the specific case of conflict-free coloring planar graphs. Section 2 provides the background necessary to understand the problem and how it is used. Section 3 introduces some applications for the conflict-free coloring problem. Section 4 describes a heuristic for conflict-free coloring general graphs. Section 5 looks into the specific case of conflict-free coloring planar graphs.

UMM CSci Senior Seminar Conference, April 2017 Morris, MN.

2. BACKGROUND

To understand the problem, the algorithms to solve it, and its results, we must first understand some graph theory, some computational complexity theory, and the precise definitions of the vertex coloring problem and the conflict-free coloring problem.

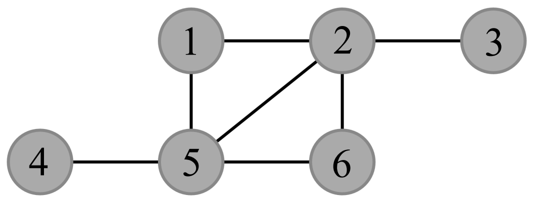

Figure 1: A simple, undirected graph G_1

2.1 Graph Theory

A graph, denoted $G = (V, E)$, is an ordered pair of two sets: a set of vertices V and a set of edges E. Each edge consists of a set of two unordered vertices from V. For example, $\{u, w\} \in E$ is an edge connecting vertices u and w where $u, w \in V$. Vertices are *adjacent* if they are connected by an edge. An edge that connects a vertex to itself is called a *loop*. A graph without loops is called *loopless*.

A simple graph is a loopless graph where no two edges connect the same pair of vertices. A planar graph is a graph that can be drawn on the plane such that its edges only intersect at their endpoints. This drawing is referred to as a planar drawing. All graphs used by the vertex coloring problem and conflict-free coloring problem are assumed to be simple graphs. An example graph is shown in Figure 1.

The open neighborhood of a vertex is a set of all vertices adjacent to v. A *closed* neighborhood, denoted $N_G(v)$, consists of the vertices adjacent to v plus v itself. For example, let's examine $N_{G_1}(1)$ from Figure 1. The open neighborhood of 1 is $\{2, 5\}$ while the closed neighborhood is $\{1, 2, 5\}$. For the rest of the paper, the term *neighborhood* will refer to closed neighborhoods unless stated otherwise.

A *path* is a sequence of edges that connect a sequence of distinct vertices. A *subgraph* is a graph H that can be formed from a subset of the vertices and edges of G. A graph is *connected* if there exists a path between every pair of vertices. A *component* of a graph G is a subgraph H where H is connected and H is not contained in any connected subgraph of G that has more vertices or edges than H has. An isolated path is a component that is itself a path.

The distance of a path can be thought of as the smallest number of edges needed to get from one vertex to another vertex. For example, in Figure 1, the distance from vertex 3 to vertex 4 would be 3 as it takes 3 edges to get there. A *distance-3-set* is a set of vertices that all have exactly pairwise distance 3 from each other. An example of a distance-3-set from Figure 1 would be $\{3, 4\}$. [3, 11]

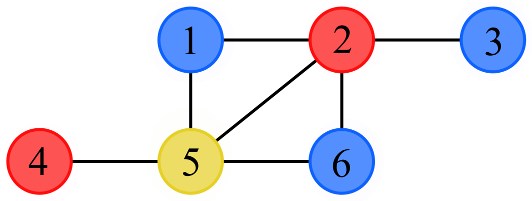

Figure 2: A minimum vertex coloring of G_1

2.2 Graph Coloring

A *vertex coloring* is an assignment of colors to each vertex of a graph G. A *proper vertex coloring* assigns colors such that no two adjacent vertices share the same color. Mathematically, it can be described as a function $f : V \to S = \{1, 2, \ldots, k\}$ such that $\forall u, w \in V$, if $(u, w) \in E$, then $f(u) \neq f(w)$. The *chromatic number* of G, denoted $\chi(G)$, is the minimum number of colors needed to properly color G. The *vertex coloring problem* (VCP), when given a simple graph G, is to find $\chi(G)$. [3]

A graph G is said to be *k-colorable* if it can be colored using k or fewer colors, i.e. $\chi(G) \leq k$. A graph having $\chi(G) = k$ is said to be a *k-chromatic* graph. The *k-colorability* problem asks if a graph can be colored using k colors. This problem is slightly different than the VCP as it is a decision problem rather than an optimization problem. An example of a 3-colorable graph and a vertex coloring is shown in Figure 2.

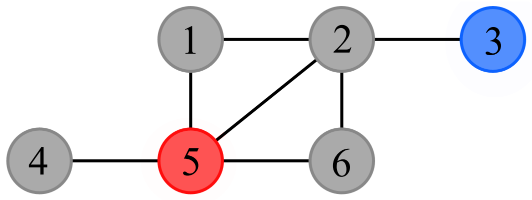

Figure 3: A minimum conflict-free coloring of G_1

A *conflict-free k-coloring* of a simple graph G assigns colors, $\{1, 2, \ldots, k\}$, to a subset $P \subseteq V$ of vertices such that $\forall v \in V$, there is a vertex $u \in N(v)$ where the color of u is unique in the neighborhood of v. The set $V \setminus P$ is the uncolored vertices, if any. Generally, a *conflict-free coloring* of a graph assigns colors to some of the vertices such that, for every vertex v, there is a unique color assigned to a vertex among v and v's neighbors. The vertex u can be thought of the *conflict-free neighbor* of v. The *conflict-free chromatic number* of G, denoted $\chi_{CF}(G)$, is the smallest k for which a conflict-free coloring exists. The *conflict-free coloring problem* (CFCP) aims to find $\chi_{CF}(G)$. [1]

An example of a conflict-free coloring is shown in Figure 3. All uncolored vertices are in gray. It is a valid conflict-free coloring because every vertex has a unique color within its neighborhood. For example, $N(1) = \{1 : \textit{uncolored},\ 2 : \textit{uncolored},\ 5 : \textit{red}\}$. As there are no other vertices colored red, we have verified there is a unique color in the neighborhood of vertex 1. We observe from Figures 2 and 3, $\chi(G_1) = 3$ and $\chi_{CF}(G_1) = 2$. It is worth noting that all proper vertex colorings are also conflict-free colorings. This is because in a proper vertex coloring, every vertex is its own conflict-free neighbor [1].

A dominating set is a subset D of V such that every vertex not in D is adjacent to at least one vertex in D. The set of colored vertices in a conflict-free coloring is a dominating set [1]. The domination number of G, denoted $\gamma(G)$, is the size of a minimum dominating set of G. An example of a dominating set is $\{3, 5\}$ of graph G_1 where $\gamma(G_1) = 2$. The *conflict-free domination number* for some k, denoted $\gamma^k_{CF}(G)$, is the minimum number of vertices that have to be colored in a conflict-free k-coloring of G. The *k-conflict-free dominating set* problem asks if a given k-coloring of a graph colors the minimum amount of vertices needed to have a conflict-free coloring that utilizes k colors.

2.3 Computational Complexity Theory

A *decision* problem, as mentioned earlier, is a problem that can be answered with 'yes' or 'no' [9]. A problem can be solved in polynomial time if an algorithm with input size n can run in at most n^k steps where k is a constant that does not depend on n. A decision problem is said to be in the class P if in the worst case, it can be solved with an algorithm that runs in polynomial time.

Given a decision problem and a proposed solution, it can be possible to verify the answer quickly. If a decision problem can be verified in polynomial time but not necessarily solved in polynomial time, it is said to be in the class *NP*. Take note that this does not exclude problems in class P; P is a subset of NP.

There are certain problems, called NP-hard, that can be proven to be as hard as every problem in NP. If problem Y can be transformed into problem X by a polynomial-time algorithm, it is said that Y can be polynomially reduced to X. A problem is NP-hard if it every problem in NP can be polynomially reduced to it. This means that if an NP-hard problem can be solved with a polynomial-time algorithm, then any problem in NP could be solved in polynomial time. A decision problem is said to be *NP-complete* when it is both in NP and NP-hard.

The k-colorability problem as well as the conflict-free k-colorability problem are NP-complete. This can be proven by using a reduction from a known NP-complete problem [5]. If we find a known hard problem Y, we can prove that another problem X is hard by a reduction from Y to X. The VCP and the CFCP have both been shown to be NP-hard [1, 6]. The k-colorability problem is proven to be NP-complete with a reduction from 3-SAT, another well-known NP-complete problem [8]. The conflict-free k-colorability problem is shown to be NP-complete by a reduction from k-colorability [1].

It is often desirable and necessary to use approximation algorithms, known as heuristics, to solve complex problems like the VCP and the CFCP. In our case, heuristics are polynomial-time algorithms that give good, but not optimal, solutions to our optimization problems. A polynomial-time approximation scheme (PTAS) is any polynomial-time approximation algorithm that takes an instance of an optimization problem and a parameter $\epsilon > 0$ and produces a solution that is within a factor $(1 + \epsilon)$ (for minimization problems) of being optimal.

3. APPLICATIONS

Before digging into methods for solving the CFCP, it is important to understand what motivated the recent studies into conflict-free coloring.

3.1 Wireless Networks

The main application for conflict-free coloring surrounds wireless networks. Cellular networks, radio, television broadcasting, and satellite communication utilize some form of wireless networks. For each of these systems, a frequency assignment problem arises with specific characteristics.

For example, imagine a cellular network. Cellular networks are heterogeneous networks with two different types of nodes. They have *base-stations* (servers) and *clients*. Base stations are all interconnected through an external backbone network. Clients can only be connected to base stations and connect via radio links. Base stations are each assigned a fixed frequency. Clients, however, are constantly searching frequencies for base-stations with strong reception.

The main problem when initially setting up these networks comes into play when assigning frequencies. Imagine two base-stations near each other having the same frequency. If a client is in reception range of both base-stations, then mutual interference occurs and the radio link is too noisy to be used for proper communication. The goal is to assign frequencies to base stations such that every client is served by some base-station and to minimize the number of frequencies used.

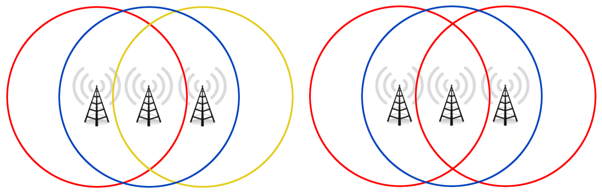

Figure 4: Cellular tower VCP and CFCP example

This problem was initially modeled as a vertex coloring problem, where the vertices are the base-stations and the edges are the pairs of base-stations that overlap in their reception range. This model is too wasteful and restrictive. Imagine a situation where a client is within the reception range of 3 base-stations. The VCP model would require 3 colors, one for each base station. If we assign one base-station a color (say blue) and the other 2 a *different* color (say red), then we only utilize two colors and the client utilizes the blue base-station and has no mutual interference. This model is shown in Figure 4. This is exactly what a conflict-free coloring does. [10]

3.2 RFID Networks

Radio frequency identification (RFID) networks are similar to wireless networks. An object, such as a credit card with an RFID chip, has a specific tag attached to it. A reader, such as a credit card scanner, can sense the presence of this object and read an ID that is assigned to the tag of the object. RFID is used for tracking progress of automobiles through a production assembly line, timing marathons and races, security access control to parking garages and buildings, and much more.

Multiple RFID readers are often set up in a given location to improve coverage of the overall area. There can also be multiple RFID readers set up that each do a different action after reading a tagged object. Unlike cellular towers, a reader can only be reading (i.e. connected to) a single tag at a time. Two readers trying to access a tagged object at the same time can cause mutual interference. The goal is to schedule for each reader a time slot for when the reader will be active.

Imagine we have a set of readers, R. Suppose they all are assigned the same frequency. We would like to schedule each reader $r \in R$ a time slot $t(r)$ where the reader r will be active. We have a set of tags P (i.e. product RFID chips). We can say that P is read by our schedule if for every tag $p \in P$, there is at least one reader $r \in R$ and a time t such that p is read by r at time t. We want to minimize the total time slots used by the schedule. This can be accomplished by modeling the situation as a conflict-free coloring problem. Again, we aim to find the minimum number of colors needed. [4, 10]

4. CF COLORING OF GENERAL GRAPHS

For this section, we will consider the NP-complete problem of conflict-free k-colorability. Given a graph G and an integer k, determine if graph G can be colored using k or fewer colors. Although all graphs can be conflict-free colored, a given graph G may not be able to be colored with k colors.

As shown with graph G_1 in section 2, even though all proper vertex colorings are also conflict-free colorings, fewer colors can usually be used. This leads us to use heuristics that differ from the vertex coloring problem. Abel et al. [1] present an efficient heuristic to color certain general graphs with k colors in a conflict-free manner. They call this heuristic *iterated elimination of distance-3-sets*.

4.1 Guaranteeing CF k-Colorability

It is wasteful to spend time coloring a graph that cannot be conflict-free colored. This leads Abel et al. to provide sufficient criterion to guarantee the conflict-free k-colorability of a certain graphs. To introduce this criterion, we need to introduce a few specific graphs. A complete graph is a simple, undirected graph where every pair of distinct vertices is connected by an edge. A complete graph on n vertices is denoted as K_n. The graph K_n^{-3} is the graph obtained by removing any three edges forming one single triangle, i.e. K_3.

A graph H is called a *minor* of a graph G if H can be formed by deleting edges and/or contracting edges (combining the vertices of an edge) from G. [1, 3]

THEOREM 1. *Let G be a graph and $k \geq 1$. If G has neither K_{k+2} nor K_{k+3}^{-3} as a minor, G has a conflict-free coloring that can be found in polynomial time using iterated elimination of distance-3-sets.*

Abel et al. give a criterion that guarantees k-colorability which is stated in Theorem 1. In the proof of this theorem, found in [1], the authors prove that their heuristic, given in section 4.3, will always generate a k-colorable graph when the specified criterion is met.

4.2 Setting Up Example CF Coloring

Before demonstrating Algorithm 1, we will demonstrate how to check if a graph, say G_2 in Figure 5, meets the given criterion.

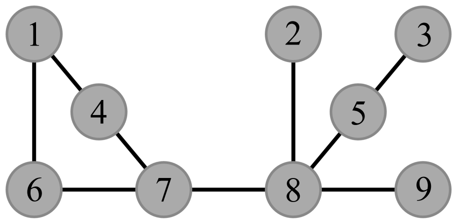

Figure 5: A simple graph G_2

We will be focusing on the simplest example, $k = 1$. Graph G_2 is a simple graph with 6 vertices. For $k = 1$, this means our graph cannot contain K_3 or K_4^{-3} as a minor. These graphs are shown in Figure 6.

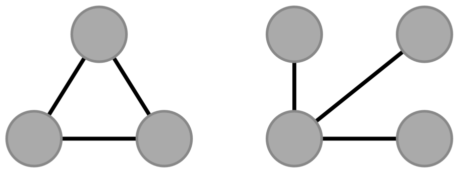

Figure 6: Graphs K_3 and K_4^{-3}, respectively

We can see that G_2 has both of these as minors. The transformation needed to see the minors is shown in Figure 7. By contracting the dotted edges, vertices 1 and 4 would combine as well as vertices 3 and 5. We would have both K_3 and K_4^{-3}. This graph does not fit our criterion; it even breaks both properties as it has both K_{k+2} and K_{k+4}^{-3} as a minor. Thus, this graph cannot be conflict-free colored with one color.

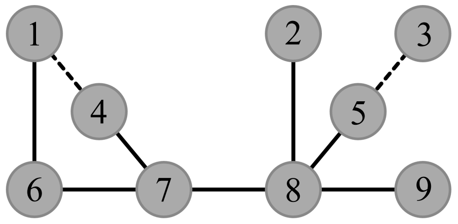

Figure 7: How to transform G_2 to see the minors

4.3 Iterated Elimination of Distance-3-Sets

The iterated elimination of distance-3-sets algorithm, defined in [1] and shown in Algorithm 1, is a polynomial-time heuristic for generating a conflict-free k-coloring of general graphs that satisfy the criterion given in Theorem 1. If the criterion is not met, the heuristic will produce a conflict-free coloring of more than k colors.

To illustrate this algorithm and show it produces a valid conflict-free coloring, we'll use graph G_3 shown in Figure 8. The algorithm starts by setting a variable i, the current color, to a (color) 1. We then remove all isolated paths from the input graph. There are currently no isolated paths in G_3, so we do not remove anything.

Algorithm 1 Iterated elimination of distance-3-sets

```
Input: A simple, undirected graph $G$
 1: $i \leftarrow 1$
 2: Remove all isolated paths from $G$
 3: while $G$ is not empty do
 4:     $D \leftarrow \emptyset$
 5:     for all components of $G$ do
 6:         Pick any vertex $v$
 7:         $D \leftarrow D \cup \{v\}$
 8:         while $\exists u$ at distance $\geq 3$ $\forall v \in D$ do
 9:             Pick $u$ at distance 3 from some vertex in $D$
10:             $D \leftarrow D \cup \{w\}$
11:         for all $u \in D$ do
12:             Color $u$ with color $i$
13:         $i \leftarrow i + 1$
14:         for all $u \in D$ do
15:             Remove $N(u)$ from $G$
16:         Remove all isolated paths from G
17: Color all removed isolated paths using color $i$
```

Since G_3 is not empty, we enter the first *while* loop and set a variable D to the empty set. G_3 has only one component (the whole graph) so we will only iterate through the *for all* on line 5 once. Pick any vertex from G_3, say vertex 8. We then add 8 to set D and thus $D \leftarrow \{8\}$. Now, while there exists a vertex with a distance of at least 3 from all vertices in D, we must pick a vertex of exactly distance 3 and add it to D. For example, vertex 2 has distance 3 from vertex 8. Now, we add vertex 2 to D and now $D \leftarrow \{2, 8\}$.

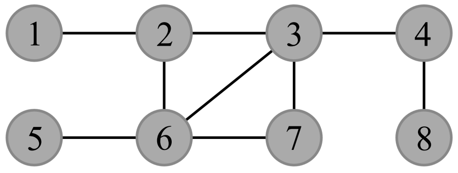

Figure 8: Graph G_3 to illustrate Algorithm 1

There are no more vertices of at least distance 3 from all vertices in D, so we move on to line 11. We color all vertices in D, $\{2, 8\}$, with color i which is known as color 1 (say red). This coloring is shown in Figure 9. We increase i by one leading to $i \leftarrow 2$.

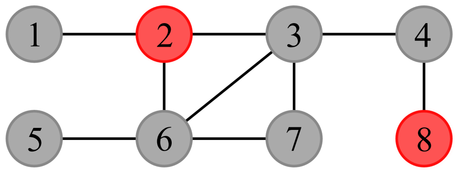

Figure 9: Graph G_3 after lines 1-13

Next, for every vertex in D, we must remove their closed neighborhood from the graph. The neighborhood of vertex 2 consists of $\{1, 2, 3, 6\}$ and the neighborhood of vertex 8 consists of $\{4, 8\}$. By removing these vertices and any edges containing these vertices, we are left with vertices $\{5, 7\}$. This is shown in Figure 10.

The last step of the algorithm is to remove all isolated

Figure 10: Graph G_3 after lines 14-15

paths from G. Since we are left with only two isolated vertices, which are trivially isolated paths, we have 2 isolated paths. We remove these isolated paths from G and are left with an empty graph. Since the graph is empty, we exit the outer *while* loop and execute line 17. We color all removed isolated paths using color i, which is color 2 (say blue). Since we had 2 isolated paths, which are really just two vertices 5 and 7, we color each vertex with color 2.

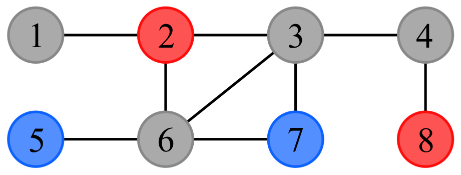

Figure 11: Coloring of G_3 based on algorithm

It was easy to color our removed isolated paths as they consisted solely of a vertex. In a more complex example, an isolated path is colored by coloring the middle vertex of every three vertices [1]. This graph is not 1-colorable as it includes K_3 as a minor. As we have shown that this graph is 2-colorable and Theorem 1 shows it is not 1-colorable, we can conclude that $\chi_{CF}(G_3) = 2$.

It is wise to note that this algorithm does not necessarily minimize the number of colored vertices. It can be beneficial to minimize the number of colored vertices (less cost) in certain applications such as building wireless networks. The conflict-free domination number for $k = 2$ of G_3, $\gamma^2_{CF}(G_3)$, is three even though we have four colored vertices in our example. This can be seen by coloring vertices $\{2, 8\}$ with one color and vertex 6 with another.

5. CF COLORING OF PLANAR GRAPHS

Conflict-free coloring has some interesting properties when we limit the input graph to be a planar graph. Recall that a graph is planar if it can be drawn on the plane in such a way that its edges intersect only at their endpoints.

5.1 Bounds for Planar Graphs

Recall the map of the USA example given in the introduction. This is an example of a planar graph and illustrates a famous theorem associated with it. We present this theorem, called the four color theorem, in Theorem 2. The most widely-accepted proof, which utilizes a computer to brute force every possible case, can be found in [7].

THEOREM 2. *Every loopless planar graph admits a proper vertex coloring with at most four distinct colors.*

Due to Theorem 2, we immediately know that every planar graph is conflict-free 4-colorable. This raises the question of whether there are planar graphs that require four colors or if fewer colors could suffice for all planar graphs. This leads us to the conflict-free variation of this theorem. We present this in Theorem 3. It is given and proven in [1].

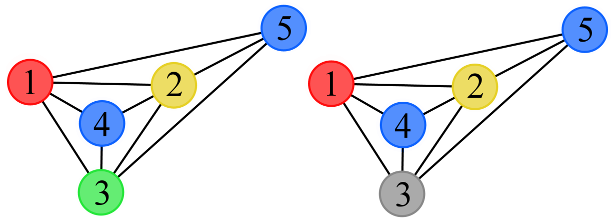

Figure 12: A vertex and CF coloring, respectively

THEOREM 3. *Every loopless planar graph admits a valid conflict-free coloring with at most three distinct colors.*

Figure 12 looks at a planar graph that requires four colors in a proper vertex coloring. Based on Theorem 3, we know that this planar graph can be conflict-free colored with only three colors. This is shown in Figure 12 as well.

5.2 CF Coloring via Dominating Set

Algorithm 1 presented in section 4.3 gives us an efficient heuristic for coloring general graphs including planar graphs. Algorithm 2, called conflict-free via dominating set, gives us another way of coloring planar graphs, albeit likely less efficient with regards to running time.

Algorithm 2 Conflict-Free Coloring via Dominating Set

```
Input: A planar graph $G = (V, E)$
1: Find a dominating set, $D$, of $G$
2: for all $v \in V \setminus D$ do
3:     Pick a vertex $u \in D$ where $\{u, v\} \in E$
4:     Contract the edge $\{u, v\}$ towards $u$
5: Find a proper vertex coloring of $G$
6: Color the original $G$ with the found coloring
```

To demonstrate this algorithm, we will utilize graph G_3 as we did for Algorithm 1. It is a planar graph, so we can use this algorithm to generate a conflict-free coloring. First, we must find a dominating set for G_3. Finding a dominating set can be done with a greedy algorithm. For small graphs, such as G_3, it is easy to find one by hand. A dominating set of G_3, $\{2, 6, 8\}$, is shown in Figure 13.

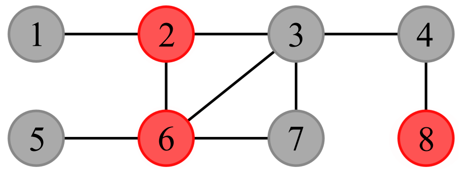

Figure 13: A dominating set of G_3

We assign our dominating set $\{2, 6, 8\}$ to D. Since $V \setminus D \neq \emptyset$, we enter the *for all* loop in line 2. Here, we iterate through all vertices in $V \setminus D$. For each vertex $v \in V \setminus D$, we will pick another vertex in D that is connected to v by an edge. Then, we will contract the edge $\{u, v\}$ towards u. This means that we combine u and v together and update any edges to reflect the changes.

To illustrate this, we start the loop by picking a vertex v, say vertex 3. We pick vertex u to be vertex 2 because $2 \in D$ and $\{2, 3\} \in E$. We contract the edge and show the resulting graph G in Figure 14. By iterating over the rest of the vertices in $V \setminus D$, we obtain a graph that is itself a path from vertex 2 to vertex 6 to vertex 8. This graph, along with its proper vertex coloring, is shown in Figure 15. We can find a proper vertex coloring by any heuristic, such as a greedy algorithm, for the VCP.

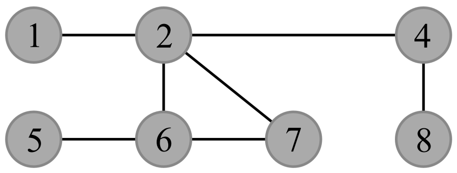

Figure 14: The graph after the first *for* loop

Lastly, we color our original graph G_3 based on the proper vertex coloring we generated. We leave any nodes not in the graph G uncolored when we color G_3. The coloring of G_3 is shown in Figure 16. It is clear that this has produced a conflict-free coloring.

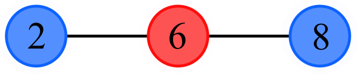

Figure 15: The graph after lines 2-5

Take note that this algorithm has produced fewer colored vertices than Algorithm 1. We had 4 colored vertices based on that algorithm where here we have 3. This algorithm helps us prove theorems based on minimizing the number of colored vertices.

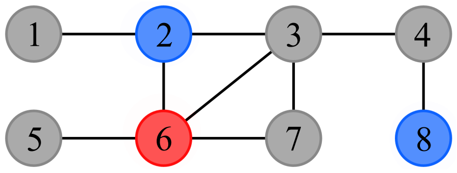

Figure 16: G_3 with minimized colored vertices

5.3 Minimizing Colored Vertices

Abel et. al [1] present results on minimizing colored vertices when conflict-free coloring planar graphs. They propose and prove Theorem 4 by using the polynomial-time algorithm described in Algorithm 2.

Because the planar minimum dominating set problem is NP-hard [5], we know that the conflict-free dominating set problem is NP-hard because we aim to find a dominating set. It follows that the corresponding decision problem, the k-conflict-free dominating set problem, is NP-complete.

THEOREM 4. *For $k \geq 4$, the k-conflict-free dominating set problem is NP-complete and $\gamma^k_{CF}(G) = \gamma(G)$ for a planar graph G. Also, there exists a PTAS for estimating $\gamma^k_{CF}(G)$.*

We get a PTAS for the conflict-free domination number by applying Algorithm 2 and finding the dominating set using the PTAS defined by Baker and Hill [2]. As the input graph G is planar, the resulting graph G' is also planar because G' is a minor of G. Thus, it can be colored with 4 colors or less by Theorem 2. We then color G with the colors of G', leaving any vertex in G but not in G' uncolored. This leaves us with a conflict-free coloring of G.

6. CONCLUSION

We have presented some of the recent heuristics and findings on the conflict-free coloring problem. Although the CFCP is NP-hard and finding the minimum number colors needed requires inefficient brute force algorithms, there are heuristics to find good conflict-free colorings.

There are many applications of the CFCP that benefit from the recent research into conflict-free coloring. It is easier to minimize frequency assignment for wireless networks and design RFID networks. Current research focuses on finding bounds and properties on specific cases such as interval graphs, types of hypergraphs, and more [4, 10].

There is also recent interest in variations of the CFCP such as requiring *another* vertex (not the selected vertex) to be colored within the neighborhood of a selected vertex. This allows for applications such as guiding a specific robot to other locations (i.e. the robot and its destination would both be a distinct unique color). Relaxing the proper vertex coloring problem to satisfy new requirements leads to new heuristics that can effectively solve more real-world problems.

7. ACKNOWLEDGMENTS

Thanks to Peter Dolan, Elena Machkasova, Peh Ng, and Jeremy Eberhardt for their advice and feedback.

8. REFERENCES

[1] Z. Abel, V. Alvarez, E. D. Demaine, S. P. Fekete, A. Gour, A. Hesterberg, P. Keldenich, and C. Scheffer. Three colors suffice: Conflict-free coloring of planar graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1951–1963. SIAM, 2017.
[2] B. S. Baker. Approximation algorithms for np-complete problems on planar graphs. *Journal of the ACM (JACM)*, 41(1):153–180, 1994.
[3] J. A. Bondy and U. S. R. Murty. *Graph theory with applications*, volume 290. Citeseer, 1976.
[4] P. Cheilaris, L. Gargano, A. A. Rescigno, and S. Smorodinsky. Strong conflict-free coloring for intervals. *Algorithmica*, 70(4):732–749, 2014.
[5] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.
[6] B. M. Moret. The theory of computation. Technical report, Addison-Wesley, Reading, Mass., 1998.
[7] N. Robertson, D. Sanders, P. Seymour, and R. Thomas. The four-colour theorem. *journal of combinatorial theory, Series B*, 70(1):2–44, 1997.
[8] P. C. Sharma and N. S. Chaudhari. A new reduction from 3-SAT to graph k-colorability for frequency assignment problem. *Int. J. Comp. Applic*, pages 23–27, 2012.
[9] M. Sipser. *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston, 2006.
[10] S. Smorodinsky. Conflict-free coloring and its applications. In *Geometry Intuitive, Discrete, and Convex*, pages 331–389. Springer, 2013.
[11] D. B. West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.