Homework Report

Shawn Seymour
University of Minnesota Morris

Last modified: February 15, 2017

Vertex Coloring

The vertex coloring problem (VCP) is one of the most common and most researched problems in graph theory. Vertex coloring is an assignment of colors, or labels, to each vertex in a graph $G = (V, E)$ such that no edge connects two vertices of the same color. The vertex coloring problem aims to find the chromatic number of a graph, denoted $\chi(G)$, the minimum number of colors needed to properly color a graph as defined above. The *k-Colorability* problem asks where a graph can be colored with k colors.

Proposition 1. *Vertex coloring is a NP-hard problem.*

Coloring a graph is computationally complex. It is NP-hard to compute the chromatic number of a given graph [4].

Proposition 2. *The k-Colorability problem, i.e. can a given graph be colored in k-colors, is a NP-complete problem.*

To prove this, we need to reduce a known NP-complete problem to our coloring problem in polynomial time. By reducing a known NP-complete problem to k-Colorability, we can conclude that k-Colorability is NP-complete as all NP-complete problems are redicible to all other NP-complete problems [5]. We will reduce the well-known 3-SAT problem first to the 3-Colorability problem, and then from there reduce to the k-Colorability problem.

3-Satisfiability (3-SAT)

The 3-SAT problem is an NP-complete problem that is a reduction of the original NP-complete problem, satisfiability (SAT). The 3-SAT problem is defined and proven to be NP-complete in [5]. 3-SAT consists of a conjunction of a number of clauses. A clause is a disjunction of a given number of propositions or their negations. We will assume all of our satisfiability problems are in conjunctive normal form (CNF). 3-SAT asks whether the variables can be assigned *true* or *false* in such a way that the whole expression evaluates to *true*.

Definition 1. SAT is a conjunction of m clauses, i.e. $C_1 \wedge C_2 \wedge \cdots \wedge C_m$. Each clause is a disjunction of at most n literals, i.e. $L_1 \vee L_2 \vee \cdots \vee L_n$. Each literal can be a variable or negative of a variable, i.e. $x_1, \neg x_1, x_2, \neg x_2, \ldots, x_k, \neg x_k$ where k is the number of distinct literals. Each literal has a value of $true$ or $false$. 3-SAT is where each clause has exactly 3 literals ($n = 3$).

Reducing 3-SAT to 3-Colorability

3-Colorability is NP-complete and was proven by [9]. To show this reduction, we need to:

1. Given an instance of 3-SAT, construct an instance of 3-Colorability
2. Show that the graph is 3-colorable, then the given 3-SAT problem is satisfied

To show how this reduction can be done, I'll create an example 3-SAT problem and reduce it to 3-Colorability. This reduction will create a graph $G = (V, E)$ having $(2n + 3m + 1)$ vertices and $(3n + 6m)$ edges [9]. The process for creating the graph, described below, was taken from [10].

Let's take the following 3-SAT expression:

$$E = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

Creating the graph

1. Create a *triangle* for each variable, i.e. x_i. Each has a common vertex B, called the base vertex. This base vertex will be colored a color, say p_1, so that the other two vertices correspond to the variable and it's negation.
2. Create a *triangle* for each clause in the formula.
3. Connect each vertex of a clause triangle to the corresponding literal vertex.

Following the above steps, we generate the following graph G from our 3-SAT expression E:

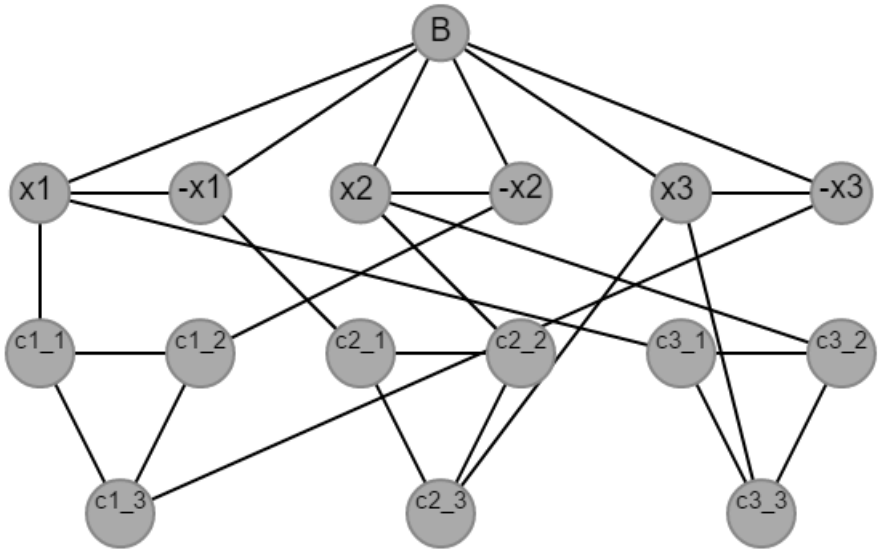

Figure 1: E represented as a graph G

If we can properly color graph G with 3 colors, then the original 3-SAT is satisfiable and the truth values of the variables are denoted by the color they received. This graph is *3-colorable* and the coloring is shown below.

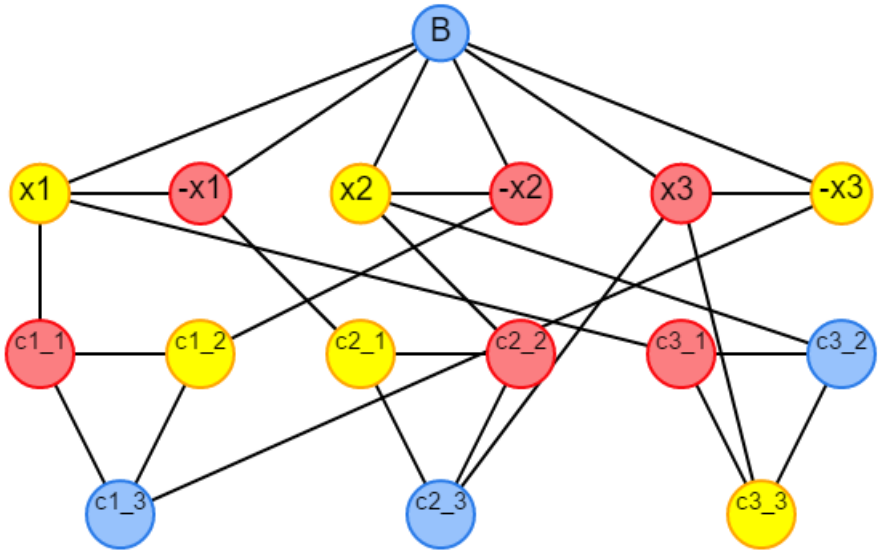

Figure 2: A possible 3-coloring of G

As we see above, *blue* is used for the base vertex and as part of some of the clause triangles. If we let *yellow* denote *true* and *red* denote *false*, we can see that our original 3-SAT is satisfied:

$$\begin{aligned} E &= (T \vee \neg T \vee \neg F) \wedge (\neg T \vee T \vee F) \wedge (T \vee T \vee F) \\ &= (T \vee F \vee T) \wedge (F \vee T \vee F) \wedge (T \vee T \vee F) \\ &= T \wedge T \wedge T \\ &= True \end{aligned}$$

This shows an example of the reduction from 3-SAT to 3-Colorability. If G was unable to be colored with 3 colors, then we would know that E is unable to be satisfied.

Heuristics

As shown above, solving the VCP is very computationally heavy. There are heuristics that give a good, but not necessarily optimal, coloring of a given graph in polynomial time. I will be analyzing and comparing a few of the popular heuristics for solving the VCP. The input for all heuristics defined here is a simple graph $G = (V, E)$. I will define them as follows:

Heuristic A

Heuristic A is the greedy algorithm.

Algorithm 1 Greedy algorithm

1: Label each vertex in V, i.e. $v_1, v_2, \ldots, v_n$
2: **for** each $v \in V$ **do**
3: Assign a color p_i to v_i using the smallest available p_i

Heuristic B

Heuristic B is the greedy algorithm with degree sequencing. It orders the vertices according to the decreasing value of their degree. This is also known as the Welsh-Powell algorithm, which is defined in [12].

Algorithm 2 Welsh-Powell algorithm

1: Label each vertex in V, i.e. $v_1, v_2, \ldots, v_n$, such that $d_G(v_1) \geq d_G(v_2) \geq \cdots \geq d_G(v_n)$
2: **for all** $v \in V$ **do**
3: Assign a color p_i to v_i using the smallest available p_i

Heuristic C

Heuristic C is very similar to heuristic B, but colors based on the *saturation* degree of vertices, which is defined below by [11]. The original DSATUR algorithm was proposed by [2].

Definition 2 (Saturation degree)**.** The saturation degree of a vertex is the number of different colors used for vertices adjacency to it.

Algorithm 3 DSATUR algorithm

1: Find the vertex $\in V$ with maximum degree and color it with color 1
2: **while** There are uncolored vertices **do**
3: Choose an uncolored vertex with maximum saturation degree
4: If there is a tie, choose the vertex with maximum degree
5: Color the vertex with the least possible number

Heuristic D

Heuristic D colors a graph by finding maximal independent sets of G.

Algorithm 4 Coloring via maximal independent set algorithm

1: **while** G is non-empty **do**
2: Find a maximal independent set of G, i.e. S_i
3: Color all vertices in S_i with color p, where p is the smallest color available
4: Let $G \leftarrow G[V \setminus S_i]$

Findings

Proposition 3. *Heuristic A and Heuristic B do not always produce an optimal solution, i.e. they do not always produce a minimum coloring of a graph.*

I will show this for *heuristic B*. It is trivial to show the same for *heuristic A* as *heuristic B* includes the same steps as *heuristic A*. I will construct a simple graph, $G = (V, E)$, such that:

$$|V| \geq 8 \tag{1}$$
$$\chi(G) > \chi^*(G) \tag{2}$$

Let χ refer to the coloring of G generated by *heuristic B*. Let χ^* be the optimal coloring of G.

Example 1

I constructed a simple graph, $G = (V, E)$, such that $\Delta(G) = 3$ and $\delta(G) = 1$. I've let $|V| = 8$ for this example.

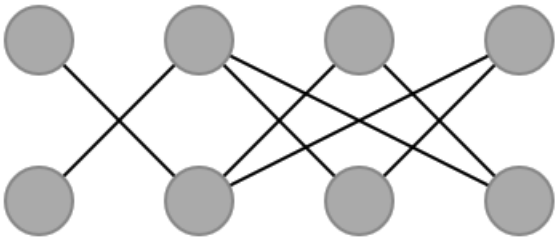

Figure 3: Uncolored original graph G

By applying *heuristic B*, we get the following coloring. The numbers indicate the ordering of vertices before applying the heuristic. Any vertex of the same degree got assigned arbitrarily. This results in $\chi(G) = 3$.

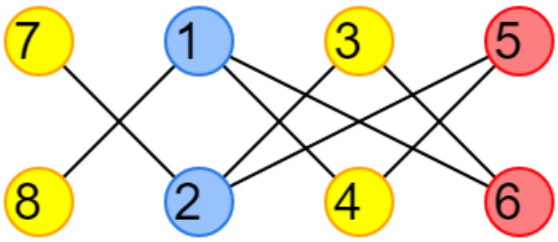

Figure 4: Coloring from applying heuristic B

Definition 3 (Bipartite graph)**.** A bipartite graph is one whose vertex set can be partitioned into two subsets X and Y such that each edge has one end in X and one end in Y

We can see that this is a bipartite graph, defined above by [1]. Thus, G is *2-colorable*. This means $\chi^*(G) = 2$. G is an example graph that satisfies conditions (1) and (2). The optimal coloring is shown below.

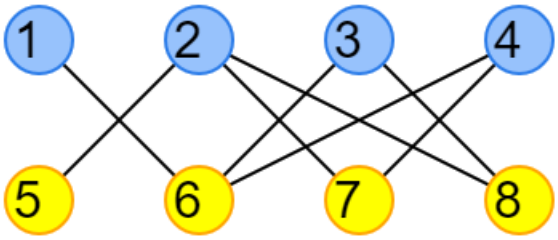

Figure 5: Optimal coloring of graph G

Proposition 4. *Heuristic D also does not always produce an optimal solution, i.e. it does not always produce a minimum coloring of a graph.*

Example 2

To show this, I will create another example that satisfies conditions (1) and (2) from above. I've constructed a graph $K = (V, E)$ for this example.

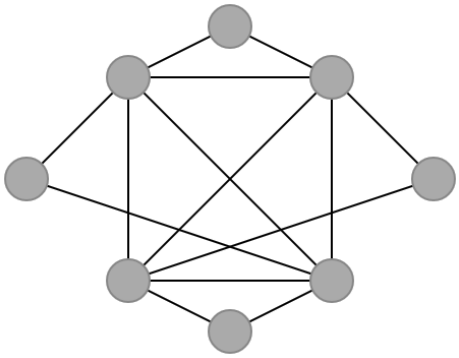

Figure 6: Uncolored original graph K

By applying heuristic D and finding maximal independent sets, we see that K is *5-colorable*.

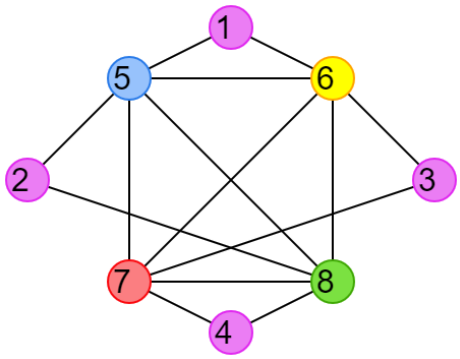

Figure 7: Coloring of K after applying heuristic D

This is a non-optimal solution. We can see that the subgraph created by vertices $\{5, 6, 7, 8\}$ is complete and thus we need at least 4 colors. This graph is *4-colorable* however, meaning $\chi^*(K) = 4$. Thus, we can see that heuristic D does not always give us an optimal solution.

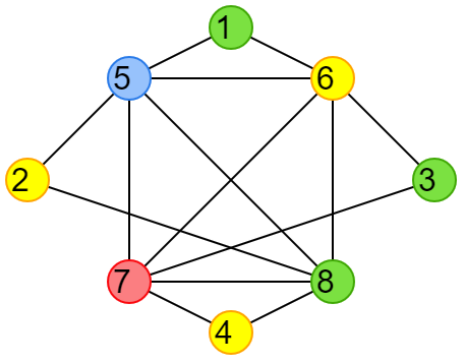

Figure 8: Optimal coloring of K

Proposition 5. *Heuristic A and heuristic B have an upper bound of* $\Delta + 1$.

To show this, I will create another example that satisfies conditions (1) and (2) from above. After doing some research into bipartite graphs, I learned that *crown graphs* are excellent at showing how bad greedy heuristics can be, as shown and defined in [6].

Definition 4 (Crown Graph)**.** A crown graph $CR_n = (V, E)$ is an undirected graph such that $U = \{u_1, u_2, \ldots, u_i\}$ and $W = \{w_1, w_2, \ldots, w_j\}$ where $V = U \cup W$, $U \cap W = \emptyset$, and $|U| = |W|$. There is an edge from u_i to w_j whenever $i \neq j$. A crown graph can also be viewed as a complete bipartite graph where the edges of a perfect matching have been removed.

Example 3

I constructed a simple crown graph, $H = (V, E)$. I've let $|V| = 10$ for this example.

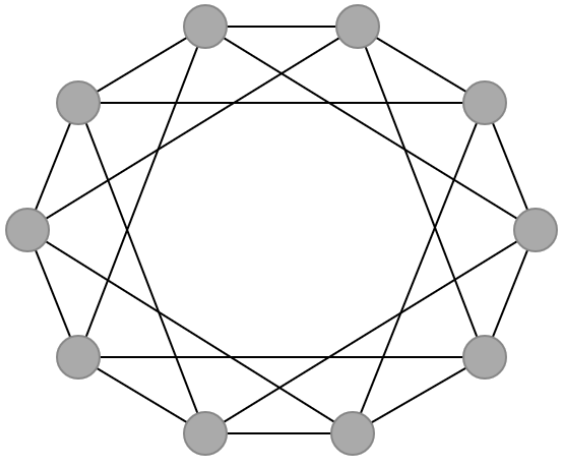

Figure 9: Uncolored original graph H

We can see that $\Delta(G) = 4$. We can also see $d_G(v) = 4$ for all $v \in V$. Thus, in both heuristics A and B, the greedy algorithm would pick an order arbitrarily. We can show using this crown graph the worst-case scenario of these heuristics.

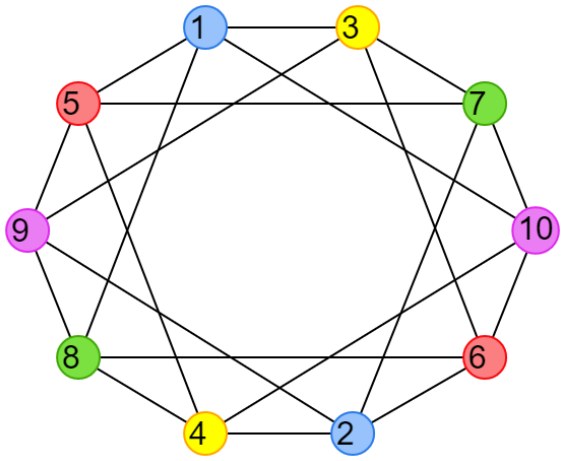

Figure 10: Worst-case coloring of H using either heuristic A or B

In Figure 5, using either heuristic with this ordering, we get $\chi(H) = 5$. This gives us $\frac{|V|}{2}$ colors. This is the worst-case for a crown graph as shown in [3], but this graph also demonstrates the upper bounds for these heuristics.

In General: Let's take a look at how this looks in general. Let $P = (V, E)$ be a simple, complete graph.

We can see that $\chi(H) = \Delta(H) + 1$. This is very easy to see in a *complete* graph, where all vertices are connected to every other vertex. This means all vertices have degree $|V| - 1$. Thus, every time we color a node, a new color is needed. And since we have $\Delta(P) = |V| - 1$ and $|V|$ vertices, we will need $\Delta(P) + 1$ colors. This is stated in *Brooks' Theorem*.

Theorem 1 (Brooks' Theorem)**.** *For any connected undirected graph G with maximum degree Δ, the chromatic number of G is at most Δ unless G is a complete graph or an odd cycle, in which case the chromatic number is $\Delta + 1$.*

The proof of *Brook's Theorem* can be found in [8]. Overall, heuristic A and heuristic B can produce some

very undesirable results. In graph H, at the worst case, these heuristics produce $\chi(H) = 5$ when $\chi^*(H) = 2$ as it is bipartite. This is shown below.

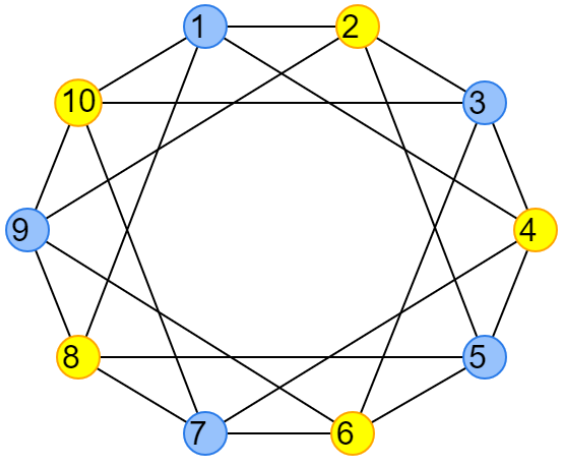

Figure 11: Optimal coloring of H

Proposition 6. *Heuristic A and Heuristic B have a running time of* $O(|V| + |E|)$.

It is possible to implement heuristic A and heuristic B efficiently and achieve a linear running time [7]. By using an adjacency list representation of a graph $G = (V, E)$, the greedy algorithm will look once at each vertex and each edge twice. This is because for each vertex, all of its neighbors are checked. This leads to the algorithm crossing each edge twuce. As this is still linear, we know $O(|V| + 2|E|) = O(|V| + |E|)$.

Proposition 7. *Heuristic A and Heuristic B produce a feasible, proper coloring of input graph* $G = (V, E)$.

Let $G = (V, E)$ be a simple graph. Assume that two vertices, say $v_1, v_2 \in V$, are connected by an edge $e_1 \in E$. Assume that v_1, v_2 are colored the same color, say p_1. We know that in both heuristics, we assign the smallest color p_i that is not being used by any of the neighboring vertices connected by an edge e_i. WLOG, the heuristic would color v_1 color p_1. The next iteration, while vertex v_2 is selected, the heuristic would see color p_1 is assigned to a neighboring vertex (v_1), and assign the next available color p_2. Thus, we've reached a contradiction.

References

[1] Bondy, J.A. and U.S.R. Murty [1976], *Graph theory with applications*, American Elsevier Publishing, New York, NY.

[2] Brélaz, D. [1979]. *New methods to color the vertices of a graph.* Communications of the ACM, 22(4), pp.251-256.

[3] Johnson, D. S. [1974], *Worst-case behavior of graph coloring algorithms*, Proc. 5th Southeastern Conf. on Combinatorics, Graph Theory, and Computing, Utilitas Mathematicae, Winnipeg, pp. 513–527.

[4] Garey, M. R., Johnson, D. S., and Stockmeyer, L. [1976], *Some simplified NP-complete graph problems*, Theoretical computer science, 1(3), 237-267.

[5] Garey, M.R. and Johnson, D.S. [2002], *Computers and intractability* (Vol. 29), New York: wh freeman.

[6] Kordecki, W. and A. Łyczkowska-Hańćkowiak [2016], *Greedy online colouring with buffering*, arXiv preprint arXiv:1601.00252.

[7] Kubale, Marek [2004], *Graph colorings*, Vol. 352, American Mathematical Soc.

[8] Lovász, L. [1975], *Three short proofs in graph theory*, Journal of Combinatorial Theory, Series B, 19(3), 269-271.

[9] Moret, B.M. [1998], *The theory of computation*, Addison-Wesley, Reading, Mass.

[10] Sharma, P. C. and Chaudhari, N.S. [2012], *A new reduction from 3-SAT to graph k-colorability for frequency assignment problem*, Int. J. Comp. Applic, 23-27.

[11] Spinrad, J. P. and Vijayan, G. [1985], *Worst case analysis of a graph coloring algorithm*. Discrete Applied Mathematics, 12(1), pp.89-92.

[12] Welsh, D. J. and Powell, M. B. [1967], *An upper bound for the chromatic number of a graph and its application to timetabling problems*, The Computer Journal, 10(1), 85-86.