Chromatic Polynomials

CSCI 4041 Honors Project: Spring 2022

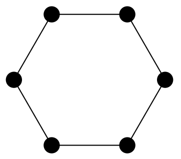
Kyler Sood

1 Introducing Graphs

We will define chromatic polynomials only for *simple undirected graphs.* These graphs can be connected or disconnected without concern. By nature, there are no natural extensions to directed graphs (where each edge has a direction) or multigraphs (where multiple edges between vertices and loops are permitted).

Definition. A *simple undirected graph* is a pair of vertex and edge sets $G = (V, E)$, with V being some set and $E \subseteq \binom{V}{2}$ containing any subsets of V with size 2.

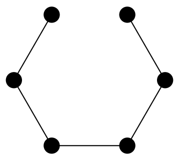
Example. The following graph G has six vertices which we can label $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ going clockwise. Then, the set of edges will be $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_4, v_5\}, \{v_5, v_6\}, \{v_6, v_1\}\}$



We will proceed by defining two operations on graphs which formulate many recursions in algebraic graph theory, deletion and contraction.

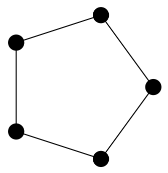
Definition. The *deletion* of edge $e = \{\{u, v\}\} \in E$ from a graph $G = (V, E)$ is another graph $G - e = (V, E \backslash \{u, v\})$ where we remove the edge but preserve the vertices u, v. Notice that this may disconnect the graph, but this is acceptable.

Example. Do deletion on any edge $e \in E$ in the hexagon graph G to obtain $G - e$.

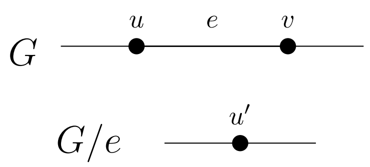


Definition. The *contraction* of an edge $e = \{u, v\} \in E$ from a graph $G = (V, E)$ is another graph G/e in which we combine the two vertices into one vertex which connects to all of the edges including u and v. Contraction may create a graph with two edges between two vertices (contraction on a triangular cycle). For the purpose of chromatic polynomials, we will combine these two edges into one edge.

Example. Do contraction on any edge $e \in E$ in the hexagon graph G to obtain G/e.



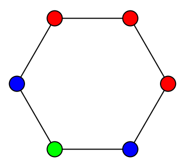
To formalize contraction: Suppose we an edge $\{u, v\} \in E$ and do contraction to obtain G/e:



2 Vertex Colorings on Graphs

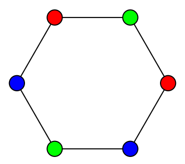
Definition. Suppose we are given a set of k colors C and a graph $G = (V, E)$. We define a $k-$*coloring* on a graph as a function $f : V \to C$ which pairs each vertex with a color.

Example. Take the hexagon graph G from before and coloring set $C = \{\text{red, green, blue}\}$:



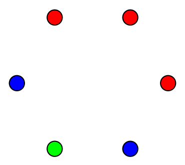
Definition. A *proper k-coloring* is a k-coloring such that no vertices connected by an edge have the same colors. That is, with f being the coloring function, for all edges $\{u, v\} \in E$, $f(u) \neq f(v)$

Example. A proper 3-coloring on hexagon graph G:



Theorem. Let C be a set of k colors and $G = (V, E)$ be a graph. Then, G has $k^{|V|}$ distinct $k-$colorings.

Question. How many distinct proper k-colorings exist for a graph G?
We have already answered this question for ordinary $k-$colorings of G. If $G = (V, \varnothing)$ is an empty graph with $|V|$ vertices and no edges, then every coloring is proper and there are $k^{|V|}$ proper $k-$colorings of G.



We will proceed by exploring the chromatic polynomial, which will answer this for nontrivial graphs.

3 Chromatic Polynomials

Definition. Let $G = (V, E)$ be a graph. We define the *chromatic polynomial* of G as:

$$\chi_G(k) = \text{number of proper } k\text{-colorings of } G$$

Example. From before, we know the empty graph $G = (V, \varnothing)$ has chromatic polynomial $\chi_G(k) = k^{|V|}$

Theorem. *Deletion-contraction theorem:* Let $e \in E$ be any edge of $G = (V, E)$. Then, for chromatic polynomial χ_G, the following recursion holds:

$$\chi_G(k) = \chi_{G-e}(k) - \chi_{G/e}(k)$$

Proof. Instead, prove an equivalent statement:

$$\chi_{G-e}(k) = \chi_G(k) + \chi_{G/e}(k)$$

By definition of the chromatic polynomial, this is equivalent to stating:

number of proper k-colorings of $G-e$ = (number of proper k-colorings of G)+(number of proper k-colorings of G/e)

Let $e = \{u, v\}$. Proceed by choosing any proper $k-$coloring of $G - e = (V, E\backslash e)$. In $G - e$, we know that the endpoints of e, u and v, are not connected by an edge (as we just deleted it). Therefore, either u and v can be the same color or different colors in a proper $k-$coloring of $G - e$. Suppose they are different colors. Then, restoring the edge e but preserving the coloring gives exactly one proper $k-$coloring of G:

Now suppose u and v have the same colors in the coloring of $G - e$. If we now combine vertices u and v, we obtain a proper $k-$coloring of G/e.

So for each proper $k-$coloring of $G - e$ there is exactly one proper $k-$coloring of either G or G/e. This transformation is invertible (reversing our constructions), so for each proper $k-$coloring of G/e or G there is exactly one proper $k-$coloring of $G - e$. This concludes the proof as the sets have equal size, so $\chi_G(k) = \chi_{G-e}(k) - \chi_{G/e}(k)$ holds. Notice that this was independent of our choice of e, so this holds for any edge $e \in E$. This gives a recursion for the chromatic polynomial. Given a graph, we can repeatedly apply this recursion until the remaining graphs are trivial and their chromatic polynomials can be determined by observation. For an algorithmic implementation, we will use the empty graph as this base case.

4 An Algorithmic Implementation for Deletion-Contraction

For this section, we will represent a graph G using an adjacency structure G.adj[v], which maps each vertex v to its set of adjacent vertices, or its *neighborhood* $\Gamma(v)$. We could also implement this for an adjacency matrix, but this would require that we either convert the matrix to a linked structure before the algorithms are implemented or write new algorithms suitable for an adjacency matrix.

The algorithmic implementation will consist of four functions: **delete**, **contract**, **nextEdge**, and **chromaticPolynomial**. First, define an edge e as an object with two integer attributes, $e.u$ and $e.v$ which are its endpoints. The graph G has only one attribute, $G.adj$ from which the entire structure can be recovered. Defining these procedures through pseudocode:

```
delete(Graph G, Edge e):
    remove all copies of e.u from G.adj[e.v]
    remove all copies of e.v from G.adj[e.u]
contract(Graph G, Edge e):
    Make a new adjacency structure temp with length —G.adj— - 1
    Copy the first $\max(e.u, e.v) - 1$ elements of G.adj into temp
    At index $\min(e.u, e.v)$, copy the result of merging G.adj[e.u] with G.adj[e.v] with no duplicates
    Copy into temp remaining elements starting at index $\max(e.u, e.v) + 1$ (skipping $\max(e.u, e.v)$)
    In each linked list in temp, replace all copies of $\max(e.u, e.v)$ by $\min(e.u, e.v)$ except for in
    temp[$\min(e.u, e.v)$] where we remove all copies of $\max(e.u, e.v)$
    For each value $i > \max(e.u, e.v)$, in every linked list of temp, replace $i$ with $i - 1$.
    Set G.adj = temp.
nextEdge(Graph G):
    Loop through each linked list in G.adj
    If the linked list contains any edge, return that edge. Otherwise, proceed to the next linked list.
    If none of the linked lists contain any edges, return NIL.
chromaticPolynomial(Graph G):
    if nextEdge(G) == NIL, return polynomial $k^{|G.adj|}$ which is the number of vertices in G.
    Otherwise, construct new graphs $G_1$ and $G_2$ with copies of G.adj.
    x = nextEdge($G_1$)
    y = nextEdge($G_2$)
    delete($G_1$, x)
    contract($G_2$, y)
    $P_1$ = chromaticPolynomial($G_1$)
    $P_2$ = chromaticPolynomial($G_2$)
    return $P_1 - P_2$
```

Notice that **chromaticPolynomial**() is a recursive algorithm, which we can qualify as divide-and-conquer since it creates two subproblems and solves both of them. The subproblems have complexity depending solely on the number of edges, and they have $(|E|-1)/|E|$ times the complexity of the original problem. Since the complexity of each subproblem is variable, we cannot employ the master theorem but will require the Akra-Bazzi theorem to determine algorithmic complexity, but I conjecture it to be $\mathcal{O}(\phi^{|G.E|+|G.V|})$ where $\phi = (1+\sqrt{5})/(2)$ for large graphs. This number ϕ is the golden ratio which also satisfies $\phi = \lim_{n\to\infty} F_{n+1}/F_n$ where F_n are the Fibonacci numbers. This conjecture is based on the Fibonacci numbers and the runtime of the deletion-contraction algorithm being similar.

5 Further Work

The chromatic polynomial is given also by the the formula $\chi_G(k) = k^c \chi_M(k)$, where c is the number of connected components of G and $\chi_M(k)$ is the *characteristic* polynomial of the adjacency matrix M of G (3). We can quickly approximate the characteristic polynomial using the QR algorithm for matrices, which converges to a diagonal matrix with the zeros of the characteristic polynomial along the diagonal. Then we can reconstruct the characteristic polynomial using the factor theorem.

Theorem. Let p be a polynomial of degree n so that it has n complex factors $x_1, \ldots, x_n$ (by the Fundamental Theorem of Algebra). Then, with constant $a \in \mathbb{C}$,

$$p(x) = a(x-x_1)(x-x_2)\ldots(x-x_n)$$

Then, if we count the number of connected components of G we will get an approximation (to some desired convergence threshold) of the characteristic polynomial. The QR algorithm is proven to have closed, decreasing error bounds by the Gershgorin circle theorem. I conjecture this numerical algorithm to admit lower time complexity than the previous divide-and-conquer algorithm, given that we run a reasonable number of iterations for the QR algorithm.

6 References

(1) Cormen, T. H., et al. (2009). *Introduction to Algorithms* (3rd ed.). The MIT Press.
(2) Bollobás, B. (1998). *Modern Graph Theory.* Springer, Graduate Texts in Mathematics.
(3) Huh, J. (2012) *Milnor Numbers of Projective Hypersurfaces and the Chromatic Polynomial of Graphs.*

7 Appendix

See attached file **README.txt** and Java code for a full implementation of a chromatic polynomial calculator for graphs. These procedures can be done in-place or not, depending on desired behavior. The code attached will not do operations in-place but instead return a modified adjacency structure.