Chromatic polynomial and Chromatic equivalence of graphs

Kwon Seok Jin

December 20, 2023

Abstract

In this paper, we first explore the definition of graph coloring and chromatic polynomial of graphs. We then examine chromatic polynomial of specific graphs such as O_n, K_n, C_n, T_n,etc, and extend our analysis to arbitrary graphs by using Fundamental Reduction Theorem : FRT and K_r-gluing of two graphs. Fulthermore, we analyze FRT computationally by writing pseudocode and real code of FRT with python. Lastly, basics of chromatic equivalence of graphs will be introduced.

1 Introduction

Although the 4 Color Theorem was proven by a computer in 1979, there is still no mathematical proof. The field of graph coloring is filled with NP-complete problems, which may seem simple and straightforward in the problems themselves, but proving them mathematically is extremely difficult. The purpose of this paper is to deeply understand chromatic number and polynomial of graphs and apply these mathematical concepts to computational field to solve the NP-complete problems like evaluating $\chi(G)$, determining $P(G, \lambda)$ and etc. Especially, we focus on chromatic polynomial of graphs by implementing program that calculates chromatic polynomial of arbitrary graphs based on FRT.

2 Chromatic Polynomial

To start with, it's essential to define what coloring of graph means. Here is the definition of graph colorings.

Definition 2.1 (Graph Colorings) *Let G be a graph and $\lambda \in \mathbf{N}$. A mapping $f : V(G) -> \{1, 2, ..., \lambda\}$ is called a λ-coloring of G if $f(u) \neq f(v)$ whenever the vertices u and v are adjacent in G.*

Two λ-colorings f and g of G are regarded as distinct if $f(x) \neq g(x)$ for some vertex x in G.

Notice that we don't consider rotating vertices. Vertices of graph are fixed and we only color each vertices.

Definition 2.2 (Chromatic Number) *Let G be a graph. Its chromatic number, $\chi(G)$, is the smallest number of colors that suffice to color G properly.*

Minimizing resource usage is crucial in any endeavor, especially in this real world where resources are limited. In this context, the chromatic number is not just a fundamental concept of graph coloring but also one of the most important problems.

Since evaluating $\chi(G)$ of arbitrary graph G is extremely difficult, we let variable λ as number of colors and focus on how many λ-colorings exist in graph G. Here comes the definition of chromatic polynomial of graph G.

Definition 2.3 (Chromatic Polynomial) *The number of distinct λ - colorings of G is denoted by $P(G, \lambda)$.The polynomial $P(G,\lambda)$ is called the chromatic polynomial of G.*

From the definition of the chromatic polynomial, we can directly get chromatic polynomial of two graphs, O_n and K_n.

2.0.1 Example 1 (Chromatic Polynomial of O_n)

For the empty graph O_n of order n, it has no edges. Every vertex can choose λ colors each and thus $P(O_n, \lambda) = \lambda^n$.

2.0.2 Example 2 (Chromatic Polynomial of K_n)

For the complete graph K_n of order n, every vertex is adjacent to any other vertices. We can choose λ colors at first vertex, λ-1 colors at next vertex, continues this process. Thus $P(K_n, \lambda) = \lambda(\lambda-1)(\lambda-2)...(\lambda-n+1) = \lambda^{\underline{n}}$.

Now let's extend our analysis to arbitrary graph G. Is there any way to get chromatic polynomial of G? Fundamental Reduction Theorem: FRT gives one way to do that.

3 Fundamental Reduction Theorem: FRT

Theorem 3.1 (Fundamental Reduction Theorem : Extension Version) *Let x and y be two non-adjacent vertices in a graph G. Then*

$$P(G, \lambda) = P(G + xy, \lambda) + P(G \bullet xy, \lambda)$$

Proof). Consider two non-adjacent vertices x and y. Then clearly, λ-colorings of G is divided into two ways.

i) x and y are colored with same color. *ii*) x and y are colored with different colors.

Since these two sets are disjoint and sum of two cases is total number of λ-coloring of G, it partitions λ-colorings of G. The number of λ-colorings of G for case *i*) is equal to $P(G \bullet xy, \lambda)$ and the number of λ-colorings of G for case *ii*) is equal to $P(G + xy, \lambda)$. Thus P(G,λ) = $P(G + xy, \lambda) + P(G \bullet xy, \lambda)$. [3]

3.1 Example 1

Suppose we have graph G as follows. Then we get $P(G, \lambda)$ by using extension version of FRT.

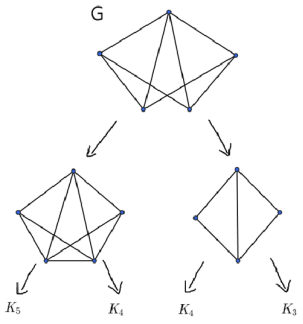

Figure 1: Example of FRT: Extension Version.

From the image which shows entire process of applying FRT above, $P(G, \lambda) = P(K_5, \lambda) + 2P(K_4, \lambda) + P(K_3, \lambda) = \lambda^5 - 8\lambda^4 + 24\lambda^3 - 31\lambda^2 + 14\lambda$. [4]

We can view the equation of FRT in a different way. Let $H = G + xy$. Then $G = H - xy$ and $H \bullet e = G \bullet e$. Substituting H to G leads to new form of FRT.

Theorem 3.2 (Fundamental Reduction Theorem : Subtraction Version) *Let G be a graph and $e \in E(G)$. Then*

$$P(G, \lambda) = P(G - e, \lambda) - P(G \bullet e, \lambda)$$

These two types of FRT can be used in different situations. When given graph G has lots of edges with respect to total number of possible edges, i.e, $\binom{n}{2}$, which means G is relatively close to complete graph, then it's desirable to use extension version of FRT. On the other hand, if graph G has fewer edges with respect to total number of possible edges, i.e, $\binom{n}{2}$, which means G is relatively close to empty graph, then using subtraction version of FRT will be appropriate.

Since we know chromatic polynomial of O_n and K_n, FRT makes possible to get chromatic polynomial of arbitrary graphs.

3.2 Example 2

Suppose we have graph G as follows. Then we get $P(G, \lambda)$ by using subtraction version of FRT.

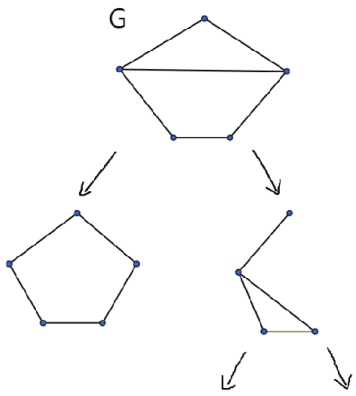

Figure 2: Example of FRT: Subtraction Version. [4]

From the image which shows entire process of applying FRT above, $P(G, \lambda) = P(C_5, \lambda) - (P(T_4, \lambda) - P(T_3, \lambda)) = \lambda^5 - 6\lambda^4 + 14\lambda^3 - 15\lambda^2 + 6\lambda$.

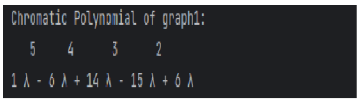

Figure 3: Result of FRT by program

4 Chromatic Polynomial Calculator

As we stated earlier, since chromatic polynomial of arbitrary graphs can be figured out by using FRT, we can design a calculator for calculating chromatic polynomial.

Since the form of FRT is defined recursively, we can write it as codes. Here's the pseudocode of FRT to get $P(G, \lambda)$.

Algorithm 1 Fundamental Reduction Theorem: FRT

function FRT(G, *density*, *level*)

if G is an empty graph **then**
  return $P(G, \lambda) = \lambda^n$
else if G is a complete graph **then**
  return $P(G, \lambda) = \lambda^{\underline{n}}$
else if G is a cycle graph **then**
  return $P(G, \lambda) = (\lambda - 1)^n + (-1)^n(\lambda - 1)$
else if G is a tree **then**
  return $P(G, \lambda) = \lambda(\lambda - 1)^{n-1}$
end if

if *density* > 0.5 **then**
  $x, y \leftarrow$ *Extract two vertices not adjacent*
  return $FRT(G + xy, density, level + 1) + FRT(G \bullet xy, density, level + 1)$
else
  $x, y \leftarrow$ *Extract two vertices adjacent*
  return $FRT(G - xy, density, level + 1) - FRT(G \bullet xy, density, level + 1)$

The FRT function has 3 parameters, target graph G, density and level. The density is a value which shows how dense graph G is, i.e, density $= \frac{2m}{n(n-1)}$ where $n = |V(G)|$ and $m = |E(G)|$. The level is current level of FRT extension tree.

There are 2 ideas we should take a look. First one is the exit conditions. It has basically 2 conditions, O_n and K_n in a naive way. To search faster we need more chromatic polynomial of specific graphs. In this pseudocode, we added cycle graph C_n and tree T_n. To improve its performance even better, we may add more chromatic polynomial of G by using K_r-gluing ideas, which we will examine later.

The remaining idea is about density. In the pseudocode, we divided two cases based on whether density value is greater than 0.5 or not. When density value is greater than 0.5, then it means G is relatively close to complete graph. Thus using extension version of FRT is definitely better than subtraction version of FRT. On the other hand, when density value is less than 0.5, then G is relatively close to empty graph and it's better to use subtraction version of FRT in this case.

Based on this pseudocode, we implemented a program for calculating chromatic polynomial of arbitrary graph G with python language. This program has 2 major features, calculating chromatic polynomial of given graph G and show its process visually with a tree structure using matplotlib.

```python
def FRT(self, G, is_dense, level):
    # This function produces chromatic polynomial of arbitrary graph G by using FRT
    # P(G, λ) = P(G + xy, λ) + P(G * xy, λ)
    if self.FRT_levels < level:
        self.FRT_levels = level

    # exit conditions.. If G is O_n , K_n, C_n, T_n, then return its chromatic polynomial
    if G.is_empty_graph():
       return self.cp_of_empty_graph(G)
    if G.is_complete_graph():
        return self.cp_of_complete_graph(G)
    if G.is_cycle_graph():
        return self.cp_of_cycle_graph(G)
    if G.is_tree():
        return self.cp_of_tree(G)
    cp = 0
    g1 = copy.deepcopy(G)
    g2 = copy.deepcopy(G)

    if is_dense:
        [x,y] = self.get_pair_of_vertices(G, False)
        g1.add_edge(x+1,y+1)
        self.FRT_results.append((g1, level+1))
        g2.contraction(x+1,y+1)
        self.FRT_results.append((g2,level+1))
        cp = self.FRT(g1, is_dense,level+1) + self.FRT(g2, is_dense,level+1)
    else:
        [x, y] = self.get_pair_of_vertices(G, True)
        g1.delete_edge(x + 1, y + 1)
        self.FRT_results.append((g1, level+1))
        g2.contraction(x + 1, y + 1)
        self.FRT_results.append((g2, level+1))
        cp = self.FRT(g1, is_dense,level+1) - self.FRT(g2, is_dense,level+1)

    cp = np.poly1d(cp, variable='λ')
    return cp
```

Figure 4: Overall Image of FRT code written in python

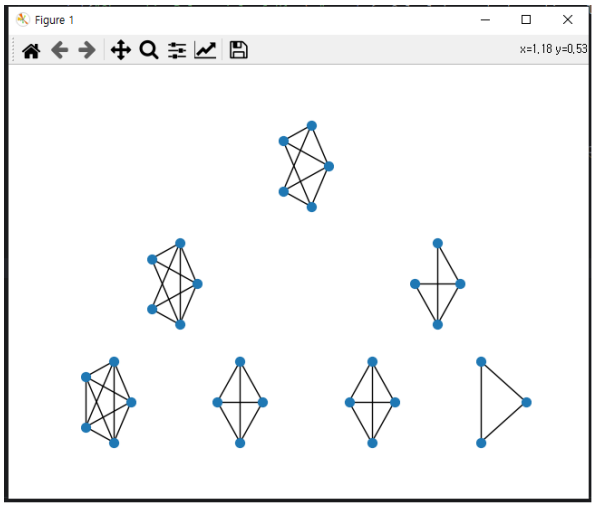

Figure 5: Show tree structure of result of FRT by program.

```
Order of graph G is 5  and Size of graph G is 8
Chromatic Polynomial of graph1:
   5      4       3       2
1 λ - 8 λ + 24 λ - 31 λ + 14 λ
```

Figure 6: Result of FRT by program.

Now we explore another method for calculating chromatic polynomial of graph G. K_r-gluing of two graphs gives an insight to view graph as attached two zigsaw puzzles.

5 K_r-gluing of graphs

Definition 5.1 (K_r-gluing of two graphs) *Let G_1 and G_2 be two graphs, and $r \in \mathbf{N_0}$ with $r \leq min\{w(G_1), w(G_2)\}$, where $W(H)$ is the clique number of graph H. Choose a K_r from each G_1 and G_2 and form a new graph G from the union of G_1 and G_2 by identifying the two chosen K_r's in an arbitrary manner.*

We call a K_r-gluing of G_1 and G_2, and denote by $\mathcal{G}[G_1 \cup_r G_2]$ the family of all K_r-gluings of G_1 and G_2.

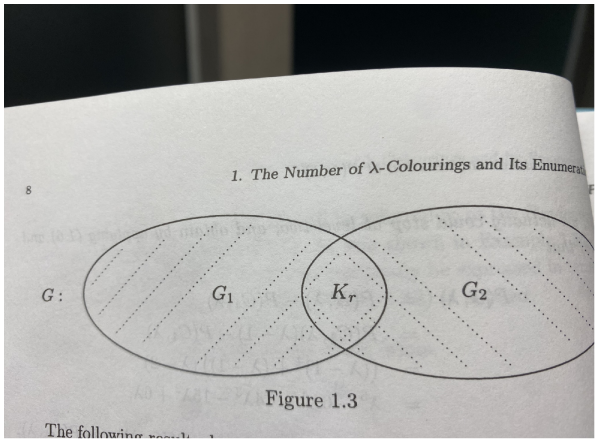

Figure 7: Set representation of K_r-gluing [4]

When piecing together a jigsaw puzzle, given two pieces must have a common slot. Similarly, we have two graphs, G_1 and G_2, we can view two graphs piece together with common slot, K_r.

Now the theorem follows gives the way to calculate chromatic polynomial of family of K_r-gluings.

Theorem 5.1 *Let G_1 and G_2 be two graphs and $G \in \mathcal{G}[G_1 \cup_r G_2]$. Then*

$$P(G, \lambda) = \frac{P(G_1, \lambda)P(G_2, \lambda)}{P(K_r, \lambda)}$$

Proof) The number of proper coloring of common part, K_r, is $P(K_r, \lambda)$. Since vertices that constructs K_r must be colored differently, the number of λ-colorings for remaining vertices of G_1 is $\frac{P(G_1,\lambda)}{P(K_r,\lambda)}$ and G_2 is $\frac{P(G_2,\lambda)}{P(K_r,\lambda)}$. By Counting Principle, $P(G, \lambda) = P(K_r, \lambda) \times \frac{P(G_1,\lambda)}{P(K_r,\lambda)} \times \frac{P(G_2,\lambda)}{P(K_r,\lambda)} = \frac{P(G_1,\lambda)P(G_2,\lambda)}{P(K_r,\lambda)}$ [2]

6 Chromatic Equivalence

We know that graph isomorphism is an equivalence relation on the set of all graphs. Thus this relation produces a partition of all graphs into equivalence classes, which are isomorphism classes. [5]

It's quite natural to think about what two graphs are chromatically equivalent means and its relation.

Definition 6.1 (Chromatic Equivalence) *Two graphs G and H are said to be chromatically equivalent or, simply, x-equivalent, written* $G \sim H$, *if* $P(G, \lambda) = P(H, \lambda)$

Relation '$\sim$' of being χ-equivalent is an equivalence relation on the family $\mathcal{G}$ of graphs and thus $\mathcal{G}$ is partitioned into equivalence classes, called χ-equivalence classes. [4]

It's quite trivial to prove relation '$\sim$' is an equivalence relation.

Since relation '$\sim$' is an equivalence relation, it produces a partition of all graphs into chromatic equivalence classes, which are called χ -equivalence classes.

6.1 χ -equivalence classes

For G $\in \mathcal{G}$, we write [G] $= \{H \in \mathcal{G} : H \sim G\}$.

G is said to be χ-unique if [G]= {G}; i.e., G does not share its chromatic polynomial with any other graph not isomorphic to it. Any empty graph O_n, complete graph K_n and cycle C_m, where n $\geq$ 1 and m ≥ 3 are χ-unique. [4]

6.2 Construction

We previously examined creating a new graph by K_r-gluing concept. Now we apply K_r-gluing idea to chromatic equivalence to construct χ-equivalent graphs.

6.2.1 Method 1

Let G_1 and G_2 be two connected graphs, and r,s $\in \mathbf{N_0}$ such that $r \leq s \leq min\{w(G_1), w(G_2)\}$. Then any graph in $\mathcal{G}[G_1 \cup_r G_2]$ is χ-equivalent to any graph in $\mathcal{G}[(G_1 \cup_s G_2) \cup_r K_s]$ [4]

Let H_1 and H_2 be two graphs such that $H_1 \in \mathcal{G}[G_1 \cup_r G_2]$ and $H_2 \in \mathcal{G}[(G_1 \cup_s G_2) \cup_r K_s]$. Then by Theorem 5.1 above, $P(H_2, \lambda) = \frac{P(G_1,\lambda)P(G_2,\lambda)}{P(K_s,\lambda)} \frac{P(K_s,\lambda)}{P(K_r,\lambda)} = \frac{P(G_1,\lambda)P(G_2,\lambda)}{P(K_r,\lambda)} = P(H_1, \lambda)$. Thus $H_1 \sim H_2$.

6.2.2 Example 1(Construction: Method 1)

Let G_1 and G_2 be graphs as follows. Let $r = 2$ and $s = 4$. Then by construction method 1, $H_1 \sim H_2$.

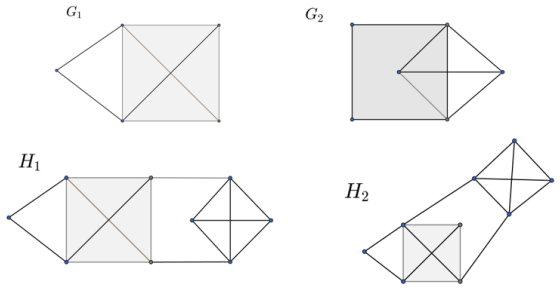

Figure 8: Construction of two χ-equivalent graphs, H_1 and H_2 [4]

Here, we used the program we made to calculate chromatic polynomial of H_1 and H_2 and check whether they are same. By the program, $P(H_1, \lambda) = P(H_2, \lambda) = \lambda^9 - 16\lambda^8 + 111\lambda^7 - 437\lambda^6 + 1070\lambda^5 - 1671\lambda^4 + 1626\lambda^3 - 900\lambda^2 + 216\lambda$.

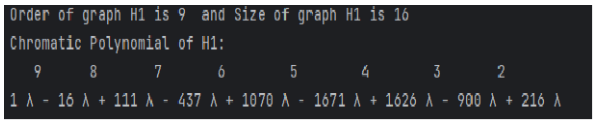

Figure 9: Chromatic Polynomial of H_1 calculated by program

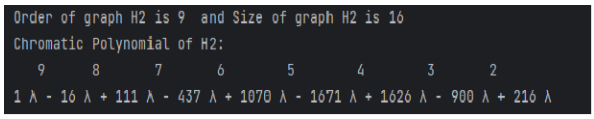

Figure 10: Chromatic Polynomial of H_2 calculated by program

We leave other 2 methods as references.

6.2.3 Method 2

Let G_1 and G_2 be two connected graphs, and $r \in \mathbf{N}$ such that $r \leq min\{w(G_1), w(G_2)\}$. Suppose that G is a K_r-gluing of G_1 and G_2 satisfying the condition. [4]

6.2.4 Method 3

Suppose that G is a connected graph having two non-adjacent vertices u and v, which form a cut such that $G - \{u, v\}$ has only two components, say G_1 and G_2. For each $i = 1, 2$, assume that there exists $x_i \in V(G_i) \cap N(u) \cap N(v)$. Let G^* be the graph obtained from G by identifying x_1 and x_2, by adding a new vertex w adjacent only to both u and v. Then $G^* \sim G$. [4]

7 Concluding remarks

We've explored the basics of graph colorings, chromatic number and chromatic polynomial of graphs. Even though we implemented the program for calculating chromatic polynomial of arbitrary graphs, its performance is not stable due to its recursive structure. In further study, we'll research how to improve the performance of FRT program (maybe apply K_r-gluing concepts) and extend to evaluating $\chi(G)$ of arbitrary graph G.

8 References

[1] Sangmok. Kim, Combinatorics (2023), 203-207.

[2] Russell Merris, Graph Theory (2001), 21-33.

[3] Ronald C. Read, An Introduction to Chromatic Polynomials (1968), 1-20.

[4] F. M. Dong, K. M. Koh, K. L. Teo, Chromatic Polynomials and Chromaticity of Graphs (2005), 1-13, 55-62.

[5] Gary Chartrand, P. Zhang, A First Course In Graph Theory (2012), 55-64.