Dilara Tekinoglu
SU ID: 27868

Assignment

March 1, 2022

Task: For a given undirected graph G = (V, E), with a set V of vertices and a set E of edges, a vertex coloring of a graph is an assignment of colors to vertices such that no two adjacent vertices get the same color. A k-coloring of a graph uses k colors. The minimum number of colors necessary to color a graph is called the chromatic number of G and is denoted by $\chi(G)$. **How can you find the chromatic number of a given graph?**

1. Brute Force Algorithm

It is obvious that any graph can be colored at most with $|V|$ different colors (every vertex gets a different color). Therefore, upper-bound of chromatic number is equal to number of vertices of the graph. My algorithm to find the chromatic number of a given graph depends on trying all possible colorings of G with k-colors where k starts from 1 and is increment one by one if the graph is not k-colorable. The algorithm operates as follows:

Algorithm 1 Finding Chromatic Number of a Graph

```
Input: $G(V, E)$
Output: The chromatic number of $G(V, E)$
$k \leftarrow 1$
while $k \leq |V|$ do
    if k-colorable(G,k) == false: then
        $k \leftarrow k + 1$
    else
        return $k$
    end if
end while
```

Time complexity of my solution depends on the complexity of k-colorable(G,k) function since other statements within the body of *While* loop take O(1) time. I can apply below solution for k-colorability check:

Try all possible colorings of G with k many colors. Then, for each coloring, check whether it is a legal coloring or not. Since there are k choices for each vertex in V, this approach takes $k^{|V|}$ time. If we apply this brute force algorithm for checking colorability of graph with k colors, then chromatic number can be find in $|V|^{|V|}$ time in the worst case:

$$1^{|V|} + 2^{|V|} + ... + |V|^{|V|} = \mathrm{O}(|V|^{|V|})$$

The time complexity of my brute force approach is super-exponential in input size $|V|$. Therefore, it is a good idea to look at a more efficient algorithm which runs in $\mathrm{O}(2.4423^{|V|})$ time.

2. A Better Solution Using Dynamic Programming: Lawler's Algorithm

Lawler was the first who proposed a dynamic programming algorithm for graph coloring problem. His algorithm is based on the theorem that every graph has an optimal coloring in which at least one of the colors is a maximal independent set[1]. By utilizing from this theorem, he puts forward the following recursive formula:

$$\chi(G[S]) = \begin{cases} 1 + min\{\chi(G[S])|I \in I(G[S])\}, & \text{if } S \neq \emptyset. \\ 0 & \text{if } S = \emptyset \end{cases} \tag{1}$$

The pseudocode of the algorithm is as follows:

Algorithm 2 Lawler's Algorithm

```
  $n \leftarrow |V|$
  $X \leftarrow$ array of length $2^n$
  X[0] = 0
  for S = 0 to $2^n - 1$ do
    for each I of G[S] do
        X[S] = min(X[S], X[S] +1)
    end for
end for
return X[$2^n - 1$]
```

The algorithm runs in O($2.4423^{|V|}$) time, still exponential but much better than the brute force solution. There are other algorithms that runs slightly better than Lawler's algorithm, but they take exponential time too. For instance, Byskov's solution takes $O(2.4023^n)$ time and Eppstein's takes $O(2.4150^n)$ time. To the best of my knowledge, there is no chromatic number finding algorithm that runs in polynomial time.

3. Would a Greedy Algorithm Work?

Greedy graph coloring can be summarized in three steps:

1. Order the vertices $V_1, V_2, ..., V_n$ in any way you prefer.
2. Order the colors $C_1, C_2, ..., C_m$
3. By following the order of vertices, color each vertex with lowest available color (A vertex must get a color different from its neighbors).

Although it is very easy to implement and efficient, this greedy solution does not always give the optimal number of colors. The result depends on the order of the vertices. Yet, it is often used for graph coloring problem since its result is generally not too far away from the

[1]A subset of vertices $I \subset V$ is a maximal independent set if I contains no adjacent vertices and no proper superset of I is independent.

optimal solution. In fact, greedy coloring algorithm outputs at most $d+1$ as result where d is the maximum degree of vertices of G[2].

REFERENCES

D. Eppstein. Small maximal independent sets and faster exact graph coloring. J. Graph Algorithms Appl., v. 7, n. 2, p. 131–140, 2003.

E. L. Lawler. A note on the complexity of the chromatic number problem. Inf. Process. Lett., 5(3):6667, Aug. 1976.

J. M. Byskov. Chromatic number in time $O(2.4023^n)$ using maximal independent sets. BRICS Rep. Ser., v. 9, n. 45, p. 1–9, 2002.

T. Leighton. Mathematics for Computer Science. Fall 2010. Lecture 6. Massachusetts Institute of Technology: MIT OpenCouseWare, https://ocw.mit.edu/. License: Creative Commons BY-NC-SA.

[2]For the proof, see Tom Leighton. Mathematics for Computer Science. Fall 2010. Massachusetts Institute of Technology: MIT OpenCouseWare, https://ocw.mit.edu/. License: Creative Commons BY-NC-SA.