WARSAW UNIVERSITY OF TECHNOLOGY

Chromatic Graph Theory

Faculty of Mathematics and Information Science

Exact Coloring Algorithms

- Preliminary Documentation -

By **Elie SAAD & Diego DELGADO**

January 4, 2022

JANUARY 4, 2022

Contents

1 Problem Formulation

In this document we will be attempting to address problem statement given in the following section.

1.1 Problem Statement

We are given the following problem statement:

Implement and compare Lawler, Eppstein and Byskov exact coloring algorithms (Byskov,J.M.:Enumerating maximal independent sets with applications to graph coloring. Operations Research Letters 32(6), 547-556)

And we are tasked with understanding the problem, coming up with a method to solve it, and then implementing it.

1.2 Problem Description

We are asked to implement three algorithms that compute the chromatic number of a given graph. The three algorithms are as known as follows: the Lawler algorithm, the Eppstein algorithm, and the Byskov algorithm.

We begin by defining, in section 2, the key terms and ideas which we will be referencing later on in the document. Then we will use these definitions to explain in depth the theory behind the three given algorithms in section 3. Then we will finish off by pointing out our testing methodology, in section 4, that we will adopt during the implementation phase of the project (which is the next phase).

2 Preliminaries

We begin with some definitions that will be used in the rest of this document. Let $G = (V, E)$ be an arbitrary graph of ordered pairs, where V stands for the finite set of elements called *vertices*, while E is the finite set of unordered pairs of vertices called *edges*.

Definition 2.1 (Vertex Coloring)**.** A *vertex-coloring* of graph $G = (V, E)$, is a function $c : V \longrightarrow \mathbb{N}$ in which any two adjacent vertices $u, v \in V$ are assigned different colors, that is $\{u, v\} \in E \implies c(u) \neq c(v)$.

1. The function c is called the coloring function
2. A graph G for which there exists a vertex-coloring which requires k colors is called k-colorable, while such a coloring is called k-coloring

3. In this case, the coloring functions induces a partition of graph G into independent subsets $V_1, V_2, ..., V_k$ for which $V_i \cap V_j = \emptyset$ and $V_1 \cup V_2 \cup ... \cup V_k = V$

Definition 2.2 (Chromatic Number)**.** The smallest number k for which there exists a k-coloring of graph G is denoted by $\mathcal{X}(G)$. Such a graph G is called k-chromatic, while any coloring of G which requires $k = \mathcal{X}(G)$ colors is called *chromatic* or *optimal*.

3 Algorithms

We will describe the algorithms in detail in this section, where we begin by explaining Lawler's algorithm, then we go on to explaining Eppstein's algorithm, and then we finish off by explaining Byskov's algorithm.

3.1 Lawler's Algorithm

Lawler [3] applied a straightforward dynamic programming algorithm, which makes use of the following simple observation. A graph G on node set N is k-colorable if and only if there is a node subset S such that S is an independent set and the restriction of G to the remaining nodes $N - S$ is $(k-1)$-colorable. By dynamic programming the subproblems for different subsets S and numbers k can be solved in $O(3^n)$ total time for n-vertex graphs. Lawler noticed that it is sufficient to consider maximal independent sets, and by a careful analysis of the number of maximal independent sets, he was able to reduce the run time to $O(2.4422^n)$ for arbitrary graphs and to $O(1.4422^n)$ for graphs that are 3-colorable.

Lawler in [3] also notes that in a coloring of a graph one of the color classes can be assumed to be maximal independent set stating the following recursion to be possible for finding the chromatic number $\mathcal{X}(G)$:

$$\mathcal{X}(G[S]) = \begin{cases} 1 + \min\{\mathcal{X}(G[S \setminus I]) | I \in I(G[S])\} & \text{if } S \neq \emptyset \\ 0 & \text{if } S = \emptyset \end{cases}$$

Algorithm 1 Lawler's algorithm for finding the chromatic number of a graph

Input:
A graph G;
Output:
$\mathcal{X}(V)$: The chromatic number

```
1: let X be an array indexed from 0 to 2^n − 1
2: for S = 0 to 2^n − 1 do
3:     for all MISs I of G[S] do
4:         X[S] = min(X[S], X[S \ I] + 1)
5:     end for
6: end for
7: return X[V]
```

where $G[S]$ denotes the vertex-induced subgraph of $S \subseteq V$ and $I(G)$ denotes the set of all maximal independent sets of G. If we index all subsets from 0 to $2^n - 1$ such that the bit-representation of each index is a bit vector denoting for each vertex whether it is in the set or not, Algorithm 1 will find the chromatic number of G.

We do note that Lawler did not explicitly define the algorithm in [3], instead, he simply implies the given recursive relation. We also note that the running time of the algorithm is calculated in [3] to be $O((2.4423^n)$, and the space that it uses is $O(2^n)$ to store X.

3.2 Eppstein's Algorithm

Eppstein [2] was the first to improve Lawler's result, to $O(2.4151^n)$. The algorithm proposed by Eppstein computes a table $X[S]$ for all $S \subseteq V$ similar to Lawler's algorithm, but the difference is, that evey time it reaches S it updates X for all supersets of S for which the value $X[S]$ could potentially give better values. In other words, if $G[S]$ is a maximal k-colourable subgraph of G, it has a maximal $(k-1)$-colourable subgraph $G[S']$ of size at least $|S'| \geq \frac{k-1}{k.|S|}$ and $S \setminus S'$ is a maximal independent set of $G[V \setminus S']$. Thus, when the chromatic number of $G[S']$ is computed, only the values of $G[S' \cup I]$ for all maximal independent sets I of size at most $\frac{|S'|}{\mathcal{X}(G[S'])}$ in $G[V \setminus S']$, are updated. We present Eppstein's algorithm in Algorithm 2.

Algorithm 2 Eppstein's algorithm for finding the chromatic number of a graph

Input:
A graph G;
Output:
$\mathcal{X}(V)$: The chromatic number

```
1: let X be an array indexed from 0 to 2^n − 1
2: for S = 0 to 2^n − 1 do
3:     if X(S) ≤ 3 then
4:         X[S] = X(S)
5:     else
6:         X[S] = ∞
7:     end if
8: end for
9: for S = 0 to 2^n − 1 do
10:    if 3 ≤ X(S) < ∞ then
11:        for all MISs of G[V \ S] of size at most |S|/X[S] do
12:            X[S ∪ I] = min(X[S ∪ I], X[S] + 1)
13:        end for
14:    end if
15: end for
16: return X[V]
```

The first part of the algorithm checks 1- and 2-colourability in polynomial time and 3-colourability using the 3-colouring algorithm of [2] with running time $O(1.3289^n)$ of all subgraphs of G. The second part runs through X and for each subgraph $G[S]$ it finds all maximal independent sets I of $G[V \setminus S]$ of size at most $\frac{|S|}{X[S]}$ and updates the value of $X[S \cup I]$.

It is clear that for any set $S \subseteq V$, $\mathcal{X}(G[S]) \leq X[S]$ during the execution of the algorithm, since $X[S]$ is only updated when the algorithm actually finds a colouring with $X[S]$ colours. For every k, all maximal k-colourable subgraphs $G[M]$ have $X[M] = k$, since they have so for $k \leq 3$ after the first half of the algorithm has run, and thus by induction also for larger k, by the argument above. Since G is a maximal $\mathcal{X}(G)$-colourable subgraph of itself, the algorithm correctly computes the chromatic number of G.

The running time of the the first part of the algorithm is $O(2.3289^n)$, while the running time of the the second part of the algorithm is $O(2.4151^n)$, which makes it the running time of the entire algorithm. We note that the

space usage of the algorithm is $O(2^n)$.

3.3 Byskov's Algorithm

The current record, based on Lawler's approach, is $O(2.4023^n)$ by Byskov. Byskov [1] shows how to obtain a 4-, 5- or 6-coloring in time $O(1.7504^n)$, $O(2.1592^n)$ and $O(2.3289^n)$, respectively, using polynomial space. He also shows how to obtain a 6- or 7-coloring in time $O(2.268^n)$ and $O(2.4023^n)$, respectively, using exponential space. As for finding the chromatic number exactly, Byskov [1] improved the bound further and showed that $\mathcal{X}(G)$ can be computed exactly in $O(2.4023^n)$ time. Byskov's algorithm is shown in Algorithm 3.

The first part of the algorithm the same as Eppstein's, namely marking all 3-colourable subgraphs of the graph in a bit vector. Then Byskov's algorithm finds all maximal independent sets I of the graph and for each checks 3-colourability of all subgraphs S of $G[V \setminus I]$, by looking in the bit vector. If they are 3-colourable, $G[S \cup I]$ is 4-colourable. This will find all maximal 4-colourable subgraphs (and maybe some that are not maximal). Then Byskov's algorithm performs the second step of Eppstein's algorithm, but it only needs to consider maximal independent sets of size at most $\frac{|S|}{4}$.

Algorithm 3 Byskov's algorithm for finding the chromatic number of a graph

Input:
A graph G;
Output:
$\mathcal{X}(V)$: The chromatic number

let X be an array indexed from 0 to $2^n - 1$
for $S = 0$ to $2^n - 1$ **do**
    if $\mathcal{X}(S) \leq 3$ **then**
        $X[S] = \mathcal{X}(S)$
    else
        $X[S] = \infty$
    end if
end for
for all MISs I in G **do**
    for all subsets S of $V \setminus I$ **do**
        if $X[S] = 3$ **then**
            $X[S \cup I] = \min(X[S \cup I], 4)$
        end if
    end for
end for
for $S = 0$ to $2^n - 1$ **do**
    if $4 \leq X[S] < \infty$ **then**
        for MISs of $G[V \setminus S]$ of size at most $|S|/X[S]$ **do**
            $X[S \cup I] = \min(X[S \cup I], X[S] + 1)$
        end for
    end if
end for
return $X[V]$

3.4 Methodology

Given the time complexity of the algorithm, graphs with greater size than 15 are not feasible in our environment.

150 graphs were created with Erdos Renyi algorithm [5] in Python, with random node size n between 7 and 10 and p probability for edge creation (uniform random value between 0.5 and 0.95). Also graphs that are not connected are not taken into account.

In order to check if algorithm's output are correct, chromatic number is being computed at creation time and then compared with later result. For initial chromatic number computation, Grinpy [6] package has been used.

4 Experimental Results

The experimentation has been conducted in an attempt to test the time complexity of the algorithms and whether or not our results match the claimants'. And thus, to do this we calculated the time required to compute the chromatic number for each graph. We have divided the graphs into 4 categories: first one containing graphs that are composed of five nodes, second one containing graphs that are composed of ten nodes, third one containing graphs that are composed of eleven nodes, and finally the fourth one containing graphs that are composed of twelve nodes. We have chosen these batches based on our size constraint - which has been explained in the methodology section of this document.

After the calculation of time that is required to run all three algorithms that have been run on all one hundred and fifty graphs, we then take the mean value of the sum of all the times as follows

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

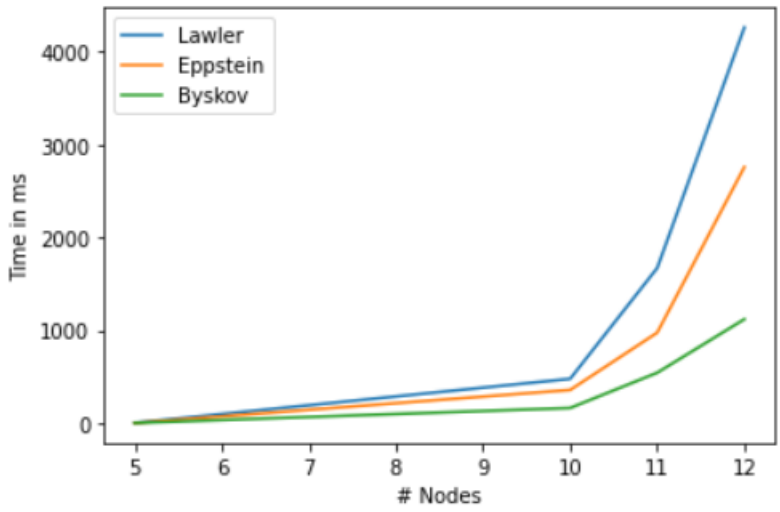
where $\bar{x}$ denotes the mean sum value, and x_i denotes each time result obtained for each of the three algorithms. The resulting averaged sum is what is shown in table 1 below

Table 1: Table showing the averaged results of the sum of all the time taken for each batch of graphs for each of the three algorithms.

Algorithm	Lawler				Eppstein				Byskov			
# Nodes	5	10	11	12	5	10	11	12	5	10	11	12
+ Avg Time (in ms)	2.36	476	1670	4259	0.78	356	974	2756	0.63	162	542	1118

To better understand these results, we have decided to plot them onto a graph to better be able to see what form of curve they form. The result of this endeavour is shown in the following figure 1

Figure 1: Results obtained from table 1 plotted on a graph where the x-axis denotes the categories that we have divided the graphs into, and the y-axis denote the time obtained in milliseconds.



In figure 1 we can see that all three algorithms' results are exponential claimed by [3, 2, 4]. We can also notice that Lawler's results are the worst in all possible cases, then comes Eppstein's and finally Byskov's who holds the best results reaffirming the claims in [3, 2, 4].

5 Analysis of the Experimental Results

The results from both table 1 and graph 1 show that the all three algorithms run in an exponential time complexity. The ranking of the algorithms from slowest to fastest is as follows: Lawler, Eppstein. The ranking reaffirms the claims made in [3, 2, 4].

All three algorithms are exact algorithms, and so, all of the results that we have obtained returned the exact chromatic number of every graph that we have fed into these algorithms. The only difference is the run time time complexity which was proven then again through our experimentation to be exponential as claimed in the original papers on the algorithms.

And thus, we can conclude that the best algorithm to be used for computing the chromatic number is Byskov based on our experimental results and analysis, and therefore is the most recommended since it is also the fastest.

References

[1] Jesper Makholm Byskov. "Enumerating maximal independent sets with applications to graph colouring". In: *Operations Research Letters* 32.6 (2004), pp. 547–556.

[2] David Eppstein. "Small maximal independent sets and faster exact graph coloring". In: *Workshop on Algorithms and Data Structures*. Springer. 2001, pp. 462–470.

[3] Eugene L Lawler and LAWLER EL. "A Note on the Complexity of the Chromatic Number Problem." In: (1976).

[4] Jesper Makholm Byskov. *Chromatic Number in Time O (2.4023^n) : $UsingMaximalIndependentSets$*. BRICS, 2002.

[5] *Erdos Renyi*. Aug. 2020. URL: `https://networkx.org/documentation/stable/auto_examples/graph/plot_erdos_renyi.html`.

[6] *Chromatic Number*. URL: `https://grinpy.readthedocs.io/en/latest/reference/invariants/chromatic.html`.

[7] *Software for Complex Networks*. Aug. 2020. URL: `https://networkx.org/documentation/stable/index.html`.