Distributed Graph Coloring

Jane Bae

CME 323 Final Project

1 Introduction

In graph theory, graph coloring or vertex coloring is an assigned of labels called colors to graph vertices such that no two adjacent vertices share the same color. More formally, we can define graph coloring as

Definition 1.1 *Given an undirected graph $G = (V, E)$, a k-***color coloring** *assigns a color $c_v \in \{1, 2, ..., k\}$ to each node $v \in V$ such that for $\forall(u, v) \in E$, $c_u \neq c_v$.*

There has been many results on the bounds on the chromatic number, $\chi(G) = \min_G(k)$ for a given graph G. 1-coloring is possible if and only if G is edgeless; 2-coloring is possible if and only if G is a bipartite graph. However, most generally, a greedy coloring shows that every graph can be colored with one more color than the maximum vertex degree $\Delta(G)$,

$$\chi(G) \leq \Delta(G) + 1.$$

There are many more studies on the lower bounds on the chromatic number; however, for the purpose of this project, we focus on finding the $(\Delta(G) + 1)$-coloring of a graph. In the following sections, we denote $\Delta = \Delta(G)$. We also assume $\Delta \ll n$.

2 Naïve Algorithm

The most straight forward way of finding a $(\Delta + 1)$ coloring is the following naïve algorithm.

Algorithm 1 Naïve color-reduction algorithm

```
function ColorReduction($G = (V, E), \Delta$)
    $n = |V|$
    for $v = 1, n$ do
        $c_v = v$
    end for
    for $v = \Delta + 2, n$ do
        $c_v = \min(\{1, ..., \Delta + 1\} \setminus \{c_u | (u, v) \in E\})$
    end for
end function
```

Due to the existence of the $(\Delta+1)$-coloring, this algorithm will find the proper coloring. This algorithm is called **color reduction**, because essentially, we start with n-coloring and we reduce the number of colors down to $\Delta+1$. The time complexity of this algorithm is $O(n)$. This algorithm cannot be parallelized in a straight forward way. So, the depth of the algorithm is $D(n) = n$, or for a better bound, $D(n) = n - \Delta$.

3 Kuhn-Wattenhofer color-reduction

We can improve the color reduction algorithm by a divide an conquer method. Given a k-colored graph G, we first divide the colors into bins of size $2(\Delta+1)$. The last bin can be of size between $2(\Delta+1)$ and $4(\Delta+1)-1$. We then apply the naïve color reduction in parallel to these bins to get $\Delta+1$ colors in each bin. We then combine two bins at a time and repeat the process.

Algorithm 2 Kuhn-Wattenhofer color-reduction algorithm [1]

```
function KWColorReduction($G = (V, E), \Delta$)
    $n = |V|$
    for $v = 1, n$ do in parallel
        $c_v = v$
    end for
    $k$ = unique elements in $\{c_v : v \in V\}$
    while $k > \Delta + 1$ do
        Divide colors into bins of size $2(\Delta + 1)$
        Let each bin be denoted as $G_i = (V_i, E_i)$.
        for $i$ do in parallel
            ColorReduction($G_i = (V_i, E_i), \Delta$)
            $k \leftarrow k - \Delta + 1$
        end for
    end while
end function
```

Starting with the n-coloring, this gives a parallel depth of $O\left(\Delta \log \frac{n}{\Delta}\right)$ as each call to the ColorReduction function is of order $O(\Delta)$, and we call this for al total of $\log \frac{n}{\Delta}$ times.

4 Linial's Algorithm

We started with the n-coloring and applied the Kuhn-Wattenhofer algorithm in the previous section. However, if we can reduce the number of colors rapidly in parallel, we can achieve a better time-complexity. We first state a combinatorial lemma due to Erdös et. al. [2]

Lemma 4.1 *For two integers n and Δ, $n > \Delta \geq 4$, there exists a family $\mathcal{J}$ of n subsets of the set*

$\{1, 2, ..., m\}$, $m = 5 \cdot \lceil \Delta^2 \cdot \log n \rceil$, such taht if $F_0, F_1, ..., F_\Delta \in \mathcal{J}$ then

$$F_0 \not\subseteq \bigcup_{i=1}^{\Delta} F_i.$$

Proof: We build a random collection $\mathcal{J}$ of subsets of $\{1, ..., m\}$ in the following way. For every element $x \in \{1, ..., m\}$, and for every index $i \in \{1, ..., n\}$, we insert x into the set S_i with probability $1/\Delta$ independently of the other pairs (x, i). For a fixed element x, and fixed distinct indices $i_0, ..., i_\Delta$, it holds that

$$\mathbb{P}\left(x \in S_{i_0} \setminus \bigcup_{j=1}^{\Delta} S_{i_j}\right) = \frac{1}{\Delta} \cdot \left(1 - \frac{1}{\Delta}\right)^{\Delta} \geq \frac{1}{4\Delta}.$$

So, the probability that for every x, $x \notin S_{i_0} \setminus \cup_{j=1}^{\Delta} S_{i_j}$ is

$$\mathbb{P}\left(\forall x, x \notin S_{i_0} \setminus \bigcup_{j=1}^{\Delta} S_{i_j}\right) \leq \left(1 - \frac{1}{4\Delta}\right)^{m} \leq n^{-(5/4)\Delta},$$

which gives

$$\mathbb{P}\left(S_{i_0} \subseteq \bigcup_{j=1}^{\Delta} S_{i_j}\right) \leq n^{-(5/4)\Delta}.$$

The probability that there will be $(\Delta + 1)$ indices $i_0, ..., i_\Delta$, such that $S_{i_0} \subseteq \cup_{j=1}^{\Delta} S_{i_j}$ is then, by union-bound, $(\Delta + 1)\binom{n}{\Delta+1} \cdot n^{-(5/4)\Delta}$, so

$$\mathbb{P}\left(\forall \text{ distinct } i_0, ..., i_\Delta S_{i_0} \not\subseteq \bigcup_{j=1}^{\Delta} S_{i_j}\right) \geq 1 - (\Delta + 1)\binom{n}{\Delta + 1} \cdot n^{-(5/4)\Delta} > 0.$$

Therefore, such selection exists. If $\Delta = 1, 2, 3$, changing the coefficient 5 to 8 will produce the same result. ■

We can use this lemma for coloring. Each vertex $v \in V$ is assigned to a subset F_v from a collection of $\mathcal{J}$ of subsets of $\{1, ..., m\}$, where $m = 5\lceil \Delta^2 \cdot \log n \rceil$. Then, each vertex sends its set to all its neighbors. Given its set F_v, and the sets $F_{u_1}, ..., F_{u_h}$, $h \leq \Delta$, we set the color of vertex v to be the element that is in $F_v \setminus \cup_{i=1}^{h} F_{u_i}$.

Notice that this algorithm can be invoked iteratively. Within $\log^* n$ rounds of iteration, the resulting coloring gives a $O(\Delta^2 \log \Delta)$-coloring. We then state another combinatorial lemma from Linial's paper [3].

Lemma 4.2 *Let q be a prime power. Then, there is a collection $\mathcal{J}$ of q^3 subsets of $\{1, ..., q^2\}$ such that if $F_0, ..., F_m \in \mathcal{J}$, where $m = \lceil (q - 1)/2 \rceil$, then*

$$F_0 \not\subseteq \bigcup_{i=1}^{m} F_i.$$

Using this lemma as the one before, we can reduce the coloring to $O(\Delta^2)$ with an additional $O(1)$ rounds of iteration. This gives a total time complexity of $\log^* n + O(1)$ to achieve a $O(\Delta^2)$-coloring.

5 Linial's Algorithm with Kuhn-Wattenhofer color-reduction

Applying the Khun-Wattenhofer color-reduction algorithm to the result of Linial's algorithm, we can find a $(\Delta + 1)$ coloring in $O(\Delta \log \Delta) + \log^* n$ time.

Algorithm 3 Linial's Algorithm with Kuhn-Wattenhofer color-reduction algorithm

```
function DistGraphColoring($G = (V, E), \Delta$)
    LinialAlgorithm($G, \Delta$) in parallel
    KWColorReduction($G, \Delta$) in parallel
end function
```

6 Communication

We looked at the ideal time complexities without any communication costs. In this section, we study what types of communication is required for each algorithm. For Linial's algorithm, at each iteration, each node has to communicate the subset that it has to its neighbors. So, worst-case, it would be $2\Delta n$ communications for the entire graph. However, this is a one-to-one communication, so assuming an abundance of parallel processors, the communication for each processor would be $O(\Delta)$. For the Kuhn-Wattenhofer algorithm, we also have the same type of communication. Overall, the communication load for each processor is not too large.

7 Conclusion

Applying these two key algorithms to the graph coloring problem, we were able to reduce the depth of the parallelization from $O(n)$ to $O(\Delta \log \Delta) + \log^* n$. For example, given a graph with one trillion nodes with maximal degree of 10 thousand, $O(n) \sim 10^12$, whereas $O(\Delta \log \Delta) + \log^* n \sim 10^5$. The difference between the two becomes larger and larger as Δ/n decrease and as n increase.

References

[1] F. Kuhn and R. Wattenhofer. On the complexity of distributed graph coloring. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing*, PODC '06, pages 7–15. ACM, 2006.

[2] P. Frankl P. Erdös and Z. Füredi. Families of finite sets in which no set is covered by the union of two others. *Journal of Combinatorial Theory, Series A*, 33(2):158–166, 1982.

[3] N. Linial. Distributive graph algorithms: Global solutions from local data. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 0:331–335, 1987.