Chapter 7

Parallel Graph Algorithms – II

In the previous week we have seen some algorithms for basic graph problems such as connected components, minimum spanning trees, and shortest paths. In this week, we study more graph algorithms that require some additional techniques.

7.1 Graph Coloring

Vertex coloring of graphs is a problem of huge interest given its applications to various problems such as scheduling and routing. In the sequential setting a famous and simple algorithm is the greedy algorithm that assigns the smallest available color in the neighborhood to each vertex. This can be seen to use at most $\Delta + 1$ colors where $\Delta$ is the maximum degree of the graph. Further progress is often highly complicated given the inherent difficulty of solving this problem even approximately close. The same difficulty applies to the parallel setting also. Hence, most researchers have tried to arrive at a coloring that uses only $O(\Delta)$ colors and can run in a poly-logarithmic time. In this section, we study two approaches to the problem. Interestingly, both the approaches use randomization and it has been shown that no deterministic algorithm can exist in the parallel setting that uses only $O(\Delta)$ colors.

7.1.1 Algorithm – I: Luby’s algorithm

The first algorithm that we study has its origins in a paper of Luby [3]. The idea is to let vertices choose their colors from a set of available colors uniformly at random. This can lead to conflicts as two neighbors may choose a same color. This is where the difficulty lies in parallel computation, and the problem is called symmetry breaking. The conflict are resolved as follows. Let each node read the color choice of its neighbors. If there is any conflict, then the node stays uncolored and will try again in the next iteration. Otherwise, the node will finalize its choice and then inform its neighbors of the same. This process is repeated until all the nodes are eventually colored. The algorithm is as follows.

Algorithm Color($G$)

1. for $v \in V$ do in parallel
   state($v$) = uncolored, color($v$) = -1, List($v$) = \{1, 2, \ldots , 2\Delta\}.
2. While $v$ is not colored do
   Choose a color, $c(v)$, in List($v$), uniformly at random.
   If no neighbor of $v$ has the same choice then
     color($v$) = $c(v)$, state($v$) = colored.
   Delete from $L(v)$ those colors that have been finalized by nodes in the neighborhood of $v$.

End Algorithm.
It remains to see how many iterations are required by any node. We use the following claims and assume that the choices made by the nodes in each round are independent.

**Claim 7.1.1** Let $E_v$ be the event that a node $v$ stays uncolored during any round. Then, $\Pr(E_v) \leq 1/2$.

**Proof.**

\[
\Pr(E_v) = \Pr(\bigcup_{w \in N(v)} c(v) = c(w)) \leq \sum_{w \in N(v)} \Pr(c(v) = c(w)) = \sum_{w \in N(v)} 1/2\Delta \leq 1/2.
\]

The above claim holds during any round and for any vertex. This is a weak upper bound as the summation really has to be done only for those neighbors that are still uncolored. However, fortunately, that is enough for our purposes.

Using the above claim, since the choices of the nodes in each round are independent, it holds that the probability that a node $v$ is uncolored even after $r$ rounds is at most $(1/2)^r$. For $r = 2 \log n$, this probability is at most $O(1/n^2)$.

Using Boole’s inequality then it can be shown that the probability that there exists a node that is uncolored after $2 \log n$ iterations is at most $1/n$, which is polynomially small. Hence, we have:

**Theorem 7.1.2** $O(\Delta)$ coloring in $O(\log n)$ iterations.

Notice however that in the network model of computation, this algorithm requires a lot of communication and computation. The central problem is how to break the conflict. If conflicts can be broken up quickly, it should help matters. In this direction, a new approach is suggested.

### 7.1.2 Algorithm 2 – Oriented Graphs

**Model and Definitions**

We model the distributed system as a graph $G = (V, E)$ with $V$ representing the set of computing entities, or processors, and $E \subseteq V \times V$ representing all the available communication links. We assume that all the communication links are undirected and hence bidirectional. All the processors start at the same time and time proceeds in synchronized rounds. We let $n = |V|$. The degree of node $u$ is denoted $d_u$ and by $\Delta$ we denote the maximum degree of $G$, i.e., $\Delta = \max_{u \in V} d_u$. When there is no confusion, $d_u$ will also be used to refer to the number of uncolored neighbors of node $u$. By $N_u$ we denote the set of neighbors of node $u$ and when there is no confusion, we use $N_u$ to refer to the set of uncolored neighbors of $u$. We do not require that the nodes in $V$ have unique labels of any kind. For our algorithms to work, it is enough that each node knows a constant factor estimate of the logarithm of the size of the network apart from its own degree and neighbors. When we consider graphs of constant degree, no global knowledge is required for our algorithm and it suffices that each node knows its own degree.

Let us denote by $[x]$ the set $\{1, 2, \ldots, x\}$ if $x \in \mathbb{N}$. If $x \in \mathbb{R}^+$, then $[x]$ would be the set $\{1, 2, \ldots, \lfloor x \rfloor\}$. Given a graph $G = (V, E)$ a vertex coloring is a mapping $c : V \rightarrow [C]$ such that if $\{u, v\} \in E$ then $c(u) \neq c(v)$, i.e., no two adjacent vertices receive the same color. Here $C$ denotes the number of colors used in the coloring. We say that a coloring is a local coloring if every node $u$ with degree $d_u$ has a color in $[ed_u]$ when the coloring uses $\epsilon \Delta$ colors. The interest in local coloring arises from the fact that a local coloring has nice implications when using the coloring in scheduling and routing problems [2].
In our model, the measure of efficiency is the number of bits exchanged. We also refer to this as the bit complexity. We view each round of the algorithm as consisting of 1 or more bit rounds. In each bit round each node can send/receive at most 1 bit from each of its neighbors. We assume that the rounds of the algorithm are synchronized. The bit complexity of algorithm $A$ is then defined as the number of bit rounds required by algorithm $A$. We note that, since the nodes are synchronized, each round of the algorithm requires as many bit rounds as the maximum number of bit rounds needed by any node in this round. In our model, we do not count local computation performed by the nodes. This is reasonable as in our algorithms nodes perform only simple local computation.

In our model, we assume that the edges in $E$ have an orientation associated with them. That is, for any two neighbors $v, w$ exactly one of the following holds for the edge $\{v, w\}$: $\{v, w\}$ is oriented either $v \rightarrow w$ or as $w \rightarrow v$. In the former we also call $v$ superior to $w$ and vice-versa in the latter. Having orientation on the edges is a property that has not been studied in the context of vertex coloring though it is a natural property since networks usually evolve and for every connection there is usually a node that initiated it. We show that algorithms for symmetry breaking can be greatly improved provided that the underlying graph is oriented. The exact way in which orientation is used for symmetry breaking is explained in Figure 7.1. As shown, if nodes $v$ and $w$ choose the same color during any round of the algorithm, in the existing algorithms, both nodes remain uncolored as in Figure 7.1(b) and have to try in a later round. With orientation, if the edge $\{v, w\}$ is oriented as $v \rightarrow w$ as shown in Figure 7.1(c), then node $v$ can retain its choice provided that there is no edge $\{u, v\}$ oriented $u \rightarrow v$ and $u$ also chooses the same color.

One parameter that will be important for our investigations is the length of the shortest cycle in the orientation. We formalize this notion in the following definition.

**Definition 7.1.3 (ℓ–acyclic Orientation)** An orientation of the edges of a graph is said to be $\ell$–acyclic if the minimum length of any directed cycle induced by the orientation is at least $\ell$. Note that this is not the girth of the given graph.

We always assume that the input graph is provided with a $\sqrt{\log n}$–acyclic orientation.

**Summary of our approach**

We now provide a brief summary of our basic approach. Our approach has the same flavor as existing distributed vertex coloring algorithms [4, 1]. Given any $\sqrt{\log n}$–acyclic oriented graph $G = (V, E)$ of constant degree $\Delta$, the algorithm for $(\Delta + 1)$–coloring proceeds as follows. Communication proceeds in rounds and in each round each yet uncolored node $v$ chooses a color $c_v$ among the available colors in $[\Delta + 1]$ uniformly at random. Node $v$ then communicates this color choice to all of its uncolored neighbors. If a node chooses a color that is in conflict with any of the choices of its neighbors, the conflict resolution rule specifies the course of action. In the algorithm of Luby[4], Johansson[1], and most other works, the conflict resolution rule is that uncolored nodes in conflict remain uncolored and have to try again in subsequent rounds. The conflict resolution rule we use is based on the orientation on the edges as explained in Section

![Figure 7.1: Orientation helps in symmetry breaking. In Figure (a) both $v$ and $w$ choose the same color. In (b), for existing algorithms both remain uncolored whereas in (c), when using orientation, node $v$ may get colored.](image-url)
7.1.2. Our algorithm is thus similar to the existing distributed vertex coloring algorithms [4, 1] except for the conflict resolution rule.

In our analysis, after $O(\sqrt{\log n})$ rounds we arrive at the situation where connected components of uncolored nodes only have simple oriented paths of length less than $\sqrt{\log n}$, with high probability. Coupled with the $\sqrt{\log n}$–acyclic orientation, it can be shown that the nodes in each such connected component can be organized into less than $\sqrt{\log n}$ layers. The layering has the property that all the oriented edges are from a node in a lower-numbered layer to a node in a higher numbered layer. This property of the layering guarantees a successful coloring of all remaining uncolored nodes in less than $\sqrt{\log n}$ rounds. This gives us the result for constant degree oriented graphs. (Theorem 7.1.4.). To arrive at the bit complexity for arbitrary graphs, (Theorem 7.1.4) we need a few additional tricks as our analysis shows.

**Upper Bound for Constant Degree Oriented Graphs**

In this section we present and analyze the algorithm for $(\Delta + 1)$–coloring constant degree oriented graphs. This demonstrates the efficacy of using orientation in vertex coloring algorithms. We defer the case of arbitrary oriented graphs to Section ?? as it requires more complicated arguments than for constant degree graphs.

The algorithm for vertex coloring constant degree oriented graphs is given in Figure 7.2. In the algorithm, the parameter $C_u$ refers to the number of colors used in the coloring by node $u$. Each node executes the algorithm Color-Random until it gets colored.

**Algorithm Color-Random**

```
While u is not colored do
    1. Node u chooses a color $c_u$ from the available colors in $[C_u]$ uniformly at random.
    2. Node u communicates its choice $c_u$, from step 1, to all of its uncolored neighbors that have a lower priority over $u$, i.e. to nodes $v$ such that $u \rightarrow v$.
    3. If node $u$ does not receive a message from any of its neighbors $w$ with $w \rightarrow u$ and $c_w = c_u$, then node $u$ gets colored with color $c_u$. Otherwise node $u$ remains uncolored.
    4. If $u$ is colored during step 3 of the current round, then $u$ informs all of its uncolored neighbors about the color of $u$.
    5. Node $u$ updates the list of available colors according to colors taken up by $u$’s neighbors.
```

**Figure 7.2:** Coloring constant degree oriented graphs by random choices.

We analyze algorithm Color-Random for constant degree oriented graphs with a $\sqrt{\log n}$–acyclic orientation and show that algorithm Color-Random can be used to obtain a $(\Delta + 1)$–coloring with a bit complexity of $O(\sqrt{\log n})$. The reduction in the bit complexity from $\Omega(\log n)$ (due to Theorem ??) to $O(\sqrt{\log n})$ comes from the fact that once every simple oriented path of length $\sqrt{\log n}$ has at least one colored node, the $\sqrt{\log n}$–acyclic orientation guarantees us connected components of uncolored nodes where each such component only has simple oriented paths of length less than $\sqrt{\log n}$. The $\sqrt{\log n}$-acyclicity of the orientation allows us to finish in a further $\sqrt{\log n}$ rounds.

**Theorem 7.1.4**

Given a $\sqrt{\log n}$–acyclic oriented graph $G = (V, E)$ of maximum degree $\Delta$, if $\Delta$ is a constant, a $(\Delta + 1)$–vertex coloring of $G$ can be obtained in $O(\sqrt{\log n})$ bit rounds, with high probability.

**Proof.** The analysis below cuts the time into two phases. Phase I ends once every simple oriented path of length $\ell = \sqrt{\log n}$ has at least one colored node, and phase II ends once all nodes are colored. We show that phase I takes at most $r = 4/\sqrt{\log n}$ rounds, with high probability. For Phase II, the proof uses the $\sqrt{\log n}$–acyclic orientation to argue that a further $\sqrt{\log n}$ rounds suffice to color all nodes. For simplicity,
we set \( C_u = 2\Delta \) for every node \( u \), but the analysis works, with minor modifications, for \( C_u = \Delta + 1 \), as long as \( \Delta \) is a constant.

Consider any simple oriented path \( P \) of length \( \ell \). For any node \( u \in P \) with \( C'_u \) remaining colors and \( d'_u \) remaining uncolored neighbors, the probability that it chooses a color that is identical to the choice of any of its uncolored neighbors is at most \( \sum_{j=1}^{d'_u} 1/C'_u \leq d'_u/(2\Delta - (d_u - d'_u)) \leq 1/2 \) as \( C'_u = 2\Delta - (d_u - d'_u) \) and \( d'_u \leq d_u \).

For any \( i \geq 1 \), denote by \( E_{P,i} \) the event that all nodes in \( P \) have a color conflict in round \( i \). Since each node chooses the color independently and uniformly at random, and \( P \) is oriented, one can identify a distinct witness for each color conflict so as to upper bound \( \Pr[E_{P,i} | \bigcap_{j=i}^{r-1} E_{P,j}] \leq (1/2)^\ell \).

Denote by \( E_P \) the event that the event \( E_{P,i} \) occurs for \( r \) consecutive rounds. Then,

\[
\Pr[E_P] = \Pr[\bigcap_{i=1}^{r} E_{P,i}] = \prod_{i=1}^{r} \Pr[E_{P,i} | \bigcap_{j=i}^{r-1} E_{P,j}] \leq (1/2)^{\ell r}.
\]

Let \( E \) denote the event that for some simple oriented path \( P \) the event \( E_P \) occurs. The number of simple oriented paths of length \( \ell \) is at most \( n\Delta^\ell \) by choosing the first vertex from \( n \) available choices and choosing each of the next \( \ell \) vertices from the at most \( \Delta \) available choices. Thus,

\[
\Pr[E] = \Pr[\bigcup_P E_P] \leq n\Delta^\ell \Pr[E_P] \leq 1/n^2.
\]

for the above value of \( r \) since \( \Delta = O(1) \). This completes Phase I of the analysis.

Consider connected components of uncolored nodes. At the end of Phase I, since any simple oriented path of length \( \ell \) has at least one colored node, each such component only has simple oriented paths of length less than \( \sqrt{\log n} \) with high probability. Moreover, the input graph does not have oriented cycles of length less than \( \sqrt{\log n} \) which implies that each such component can be organized into less than \( \sqrt{\log n} \) layers with oriented edges going only from a node in a lower-numbered layer to a node in a higher numbered layer. This layering can be achieved by the following process. Nodes with no superiors are assigned to layer 0. After removing these nodes, nodes in the rest of the component with no superiors are assigned to layer 1, and so on, until there are no nodes left. Such a procedure terminates in less than \( \sqrt{\log n} \) rounds, implying that the layer number of any node is less than \( \sqrt{\log n} \). Otherwise, there must exist either a simple oriented path of length at least \( \sqrt{\log n} \) or an oriented cycle of length less than \( \sqrt{\log n} \). Both of these conditions result in a contradiction and hence the layering process must terminate in less than \( \sqrt{\log n} \) rounds. Figure 7.3 shows an example along with the assignment of nodes to layers.

Now, in Phase II, during every round the uncolored nodes assigned to the lowest layer number presently get colored as the nodes assigned to the lowest layer can always retain their color choice from Step 1. This implies that Phase II can finish in less than \( \sqrt{\log n} \) rounds.

Since in each round each uncolored node has to exchange \( O(\log \Delta) = O(1) \) bits, the bit complexity of the algorithm Color-Random is \( O(\sqrt{\log n}) \).

We note that the same proof also holds for 3-coloring cycle graphs, with any orientation, with minimal changes. Coupled with the lower bound result in Theorem ??, our analysis for the case of constant degree graphs is tight with respect to the bit complexity, up to constant factors. The algorithm and the analysis can be modified easily to achieve a local coloring also.

### 7.2 Fractional Independent Sets

In this section, we study yet another symmetry breaking algorithm to construct fractional independent sets. We first define the idea of a fractional independent set and then use randomization to construct such sets.
Definition 7.2.1 Given a graph $G = (V, E)$, a set $I \subseteq V$ is called a $(c, d)$–fractional independent set (of $G$) if:

- $I$ is an independent set,
- For any vertex in $v \in I$, the degree of $v$ is at most $d$, and
- $|I| \geq |V|/c$

Notice that linked lists are a simple example of a graph that has a $(1/3, 2)$–fractional independent set. But in the following we show that also planar graphs have such a fractional independent set for a constant $c$. This makes it appealing for algorithm design for planar graphs. First we recall Euler’s theorem about planar graphs.

Theorem 7.2.2 (Euler’s Theorem) For any planar graph $G = (V, E)$ with $|V| \geq 3$, it holds that $|E| \leq 3|V| - 6$.

An immediate application of Euler’s theorem about planar graphs gives the following lemma.

Lemma 7.2.3 Let $d \geq 6$ and $G = (V, E)$ be a planar graph. Let $V_d$ be the set of vertices of degree at most $d$. Then, $|V_d| \geq |V|/c$ for some constant $c$.

Proof. Let $V_h$ denote the set of high degree vertices, i.e., vertices of degree at least $d + 1$. Then,

$$\sum_{v \in V} \deg(v) \geq \sum_{v \in V_h} \deg(v) \geq (d + 1)|V_h|$$

Using Euler’s theorem we have thus:

$$2 \cdot (3|V| - 6) \geq (d + 1)|V_h|$$

Equivalently,

$$|V_d| = |V| - |V_h| \geq \frac{d - 5}{d + 1}|V|$$
With \( d = 6 \) in the above lemma, it shows that there are at least \( |V|/7 \) vertices of degree at most 6. However, we seek to find such a large low-degree independent set. For this, we can consider each vertex of degree at most \( d \), add this vertex to a set \( I \), delete its neighbors, and repeat till all vertices of degree at most \( d \) are considered. Proceeding thus, it can be shown that:

**Claim 7.2.4** \(|I| \geq |V_d|/(d+1)\)

Thus, at least in the sequential setting, one can construct a fractional independent set for planar graphs. But, to arrive at such a construction in the parallel setting is possible only when one uses randomization. The following algorithm shows the way.

**Algorithm FractionalIndependentSet** \((G)\)

- For each vertex \( v \in V \) do in parallel
  - if \( \deg(v) \leq 6 \) set lowdegree\((v) = 1 \) else
    - set lowdegree\((v) = 0 \), label\((v) = 0 \).
  - For each vertex \( v \) such that lowdegree\((v) = 1 \) do in parallel
    - set label\((v) = 1 \) with probability \( 1/2 \) and set label\((v) = 0 \) with probability \( 1/2 \)
  - For each vertex \( v \) with label\((v) = 1 \) do in parallel
    - if \( uv \in E \) and label\((v) = 1 = \text{label}(u) \) then
      - set label\((v) = 0 \). End Algorithm.

In the following we show that the vertices that survive this algorithm form a fractional independent set, with high probability.

**Lemma 7.2.5** With \( d = 6 \), with high probability, the set \( I = \{v \in V|\text{label}(v) = 1\} \) forms a fractional independent set of a planar graph.

**Proof.** Clearly, \( I \) is an independent set and for every vertex in \( I \), its degree is bounded by 6. We only have to show that \( I \) has at least a constant fraction of the vertices in \( V \). For this, notice that for any vertex \( v \in V_6 \), \( \Pr(v \in I) = (1/2)^7 \). However, we cannot apply chernoff bounds directly on these indicator random variables as they are not independent. To manufacture independence, we proceed as follows.

Observe that if we somehow make a subset of the nodes in \( V_6 \) be in \( I \) independent of each other then Chernoff bounds can be applied. Further, if the neighbors of nodes are not overlapping then the events of such nodes being in \( I \) are independent of each other. Thus, we should look at nodes at a distance of at least 3 to each other. Let us call this set \( V' \). Thus, \( V' = \{v \in V_6|\text{dist}(v, w) \geq 3, w \in V_6\} \). Importantly, we have:

**Claim 7.2.6** \(|V'| \geq |V_6|/36\).

So, we can apply Chernoff bounds on the sum \(|V'| \) indicator random variables each with success probability of \( 1/2^7 \) so that the value of this sum is a lower bound on \(|I|\).

Letting \( X = \sum_{v \in V'} X_v \), with \( E(X) = |V'|/2^7 \), we have:

\[
\Pr[X \leq (1 - \epsilon)E(X)] \leq e^{-E(X)\epsilon^2/3}
\]

With \( \epsilon = 1/2 \), we can find a constant \( c \) so that with high probability, \(|I| \geq |V|/c\).
Bibliography


