Graph Coloring with $\Delta + 1$ Colors: Algorithms, Proofs, and Complexity

Introduction

An overview of an graph coloring algorithm for a graph with a maximum degree Δ can colors using at most $\Delta + 1$ colors, assuming the EREW PRAM model.

Algorithms

Coloring Rooted Trees

The `6-Color-Rooted-Tree` algorithm reduces the number of colors in a rooted tree iteratively:

1. **Initialization**:
   - Assign each node in the tree a unique initial color equal to their IDs. It is essential for the IDs to be distinct numbers from 0 to N - 1, where N is the size of the graph
   - Each node knows the size of the graph, it's ID and Δ
2. **Color Refinement**:
   - Repeat the following steps until the number of distinct colors at each node is reduced to at most 6:
     - Each node communicates with its parent (the node to which it is connected in the tree).
     - The node compares its color with its parent's color bit by bit to find the smallest bit position where they differ.
     - It creates a new color by concatenating this bit position and the value of the differing bit.
   - This step ensures that no two adjacent nodes have the same color because the new color explicitly encodes a distinction between each node and its parent.
3. **Termination**:
   - When the number of colors is reduced to 6, stop the iterative process.
   - The tree is now 6-colored.

The above can be refined further into a `3-Color-Rooted-Tree` algorithm:

(a) **Reduce to Three Colors**:

   - Take the 6-colored tree and perform three sequential steps to reduce the number of colors to 3.
   - In each step, recolor each node by checking the colors of its parent and children and choosing a new color that avoids conflict.

Coloring Constant-Degree Graphs

The `Color-Constant-Degree-Graph` algorithm extends tree-coloring to general graphs with degree Δ:

1. **Partition into Pseudoforests**:
   - Identify a maximal pseudoforest within the graph. A pseudoforest is a subset of the graph where each connected component is a tree or contains a single cycle.
   - This can be achieved by assigning a direction to one edge per node in the graph (if possible) and ensuring that no node has more than one outgoing edge.
   - Remove the edges of the pseudoforest from the graph.
   - Repeat this process until all edges are removed, partitioning the graph into Δ or fewer pseudoforests.
2. **Color the Pseudoforests**:
   - Use the `3-Color-tree` algorithm to color each pseudoforest in parallel.
   - Since pseudoforests are edge-disjoint, they can be colored independently without conflict. The tree-coloring algorithm works for pseudoforests as well, since each node can treat it's outgoing edge like an edge to parent, and the ones without it act as root(s). Each step of an algorithm is determined for each node, and a node in a tree only compares itself to it's neibours, which results in valid coloring in a pseudoforest as well.
3. **Combine Colors**:
   - Reintroduce the edges of each pseudoforest one at a time.
   - For each pseudoforest, recolor the nodes by choosing new colors from the palette $\{0, \ldots, \Delta\}$ to ensure consistency with adjacent nodes from previously processed pseudoforests.

Theorems

Theorem 1: 6-Color-Rooted-Tree Algorithm Correctness The `6-Color-Rooted-Tree` algorithm produces a valid 6-coloring of a tree in $O(\log^* n)$ time on a CREW PRAM using a linear number of processors.

Proof:

- If node i after one step of reassigning colors would have the same color as its parent, then that would mean that in their previous color, they differed on the same i-th bit, and were the same on i - 1 previous bits. But, since the last bit of the new color is the value of the differing bit, then since they were different on the i'th bit, they are different on the 0-th bit now.

- **Time Complexity:** Each iteration involves constant-time communication between a node and its parent, resulting in $O(\log^* n)$ total time, because each iteration reduces the amount of colors logarythmically, since the maximum new color is bounded by $2 \log^* c$, where c is the previous max color, because we take the bit length of c and then bitwise shift left by one

Theorem 2: 3-Color-Rooted-Tree Algorithm Correctness Given a rooted tree, the `3-Color-Rooted-Tree` algorithm constructs a valid 3-coloring of the tree using $O(n)$ processors and $O(\log^* n)$ time on a CREW PRAM.

Proof:

- After obtaining a 6-coloring using the `6-Color-Rooted-Tree` algorithm, the number of colors is reduced to 3 by successive stages. Each stage shifts colors downwards and eliminates one color by recoloring nodes to avoid conflicts. After said shift, each child has the color of it's parent. Node v then can be colored in one of 3 colors, either 0,1 or 2, since all children have the same color and it's parent has the second, which means only two of them can be taken, leaving the third one for v

- Each stage takes constant time.

Theorem 3: Coloring Constant-Degree Graphs The `Color-Constant-Degree-Graph` algorithm colors a constant-degree graph with $\Delta + 1$ colors in $O((\log \Delta)(\Delta^2 + \log^* n))$ time on an EREW PRAM using $O(n)$ processors.

Proof:

- The graph is partitioned into Δ or fewer pseudoforests, taking (O(Δ) time, not including the time of partition. On the distributed system where each node has to send and receive from each of it's neighbours, said multiplier is (O(Δ). In EREW PARAM, An unused edge can be selected in logarythmic time. Each pseudoforest is 3-colored in $O(\log^* n)$ time using the `3-Color-Pseudoforest` algorithm, times the same multiplier as before.

- Combining the colors requires $O(\log \Delta^2)$ iterations, where we iterate through each pseudoforest and each color, giving $O(\Delta^2)$ time, disregarding local computation time. In EREW PRAM, counting the optimal local computation time gives an additional $O(\log \Delta)$ multiplier .

- The total time complexity on EREW PRAM is $O((\log \Delta)(\Delta^2 + \log^* n))$.

Complexity Analysis

- **Rooted Tree Coloring:** $O(\log^* n)$ time using $O(n)$ processors on a CREW PRAM.

- **Constant-Degree Graphs:** $O((\log \Delta)(\Delta^2 + \log^* n))$ time using $O(n)$ processors.

References

- Goldberg, Plotkin, Shannon - Parallel symmetry-breaking in sparse graphs