Network Computing and Efficient Algorithms

Vertex Coloring

Xiang-Yang Li and Xiaohua Xu

School of Computer Science and Technology
University of Science and Technology of China (USTC)

September 1, 2021

Some Basic Concepts

- Distributed Systems
  - Many nodes: Many processors or entities are active in the system at any moment.
  - Certain degrees of freedom: they have their own hard- and software.
  - Share resources and information: nodes may communicate by exchanging messages (point-point communication, wireless broadcast communication), or by means of shared memory.
  - Coordination: nodes often work together to solve a global task; or they are autonomous agents that have their own agenda and compete for common resources.

Some Basic Concepts

- Some of the fundamental issues
  - Communication: Often communication cost dominates the cost of local processing or storage.
  - Collaborative: How to coordinate a distributed system so that it performs some task efficiently?
  - Fault-tolerance: A advantage of a distributed system is that even in the presence of failures the system as a whole may survive.
  - Locality: Global information is not always needed to solve a task, often it is sufficient if nodes talk to their neighbors.

Some Basic Concepts

- Some of the fundamental issues
  - Parallelism: How fast can you solve a task if you increase your computational power?
  - Symmetry breaking: Sometimes some nodes need to be selected to orchestrate computation or communication.
  - Synchronization: How to implement a synchronous algorithm in an asynchronous environment?
  - Uncertainty: The nodes cannot know what other nodes are doing at this exact moment, and the nodes are required to solve the tasks at hand despite the lack of global knowledge.

Some Basic Concepts

- How to measure the performance?
    - Optimality of the solution (Efficacy)
    - Cost needed to find the solution (Efficiency)
        - Computation Complexity
        - Communication complexity
    - Robustness (with respect to failure)
    - Convergence speed (system changes)
    - Fairness?

Outline

Vertex Coloring

Problem 1.1 (Vertex Coloring)

Given an undirected Graph $G = (V, E)$, assign a color c_v to each vertex $v \in V$ such that the following holds: $e = (v, w) \in E \Rightarrow c_v \neq c_w$.

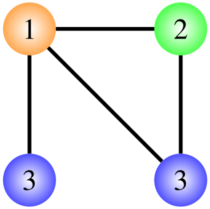


Figure: 3-colorable graph with a valid coloring.

The application often asks us to use few colors!

Vertex Coloring

Assumption 1.3 (Node Identifiers)

Each node has a unique identifier, e.g., its IP address. We usually assume that each identifier consists of only $log n$ bits if the system has n nodes.

Remarks:

- Sometimes we might even assume that the nodes exactly have identifiers $1, ..., n$.
- It is easy to see that node identifiers (as defined in Assumption 1.3) solve the coloring problem 1.1, but using n colors is not exciting. How many colors are needed is a well-studied problem.

Some Notations

Definition 1.4 (Chromatic Number)

Given an undirected Graph $G = (V, E)$, the chromatic number $\chi(G)$ is the minimum number of colors to solve Problem 1.1.

Definition 1.6 (Degree)

The number of neighbors of a vertex v, denoted by $\delta(v)$, is called the degree of v. The maximum degree vertex in a graph G defines the graph degree $\Delta(G) = \Delta$.

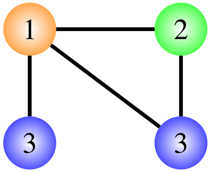
Some Notations

Definition 1.4 (Chromatic Number)

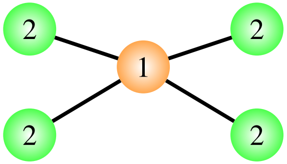
Given an undirected Graph $G = (V, E)$, the chromatic number $\chi(G)$ is the minimum number of colors to solve Problem 1.1.

Definition 1.6 (Degree)

The number of neighbors of a vertex v, denoted by $\delta(v)$, is called the degree of v. The maximum degree vertex in a graph G defines the graph degree $\Delta(G) = \Delta$.



$\chi(G) = 3, \Delta(G) = 3$



$\chi(G) = 2, \Delta(G) = 4$

Centralized Greedy Algorithm

We can assume that colors are numbered as $1, 2, \ldots$.
Initially, all nodes are not colored (marked as color 0)

```
ALGORITHM 1.5: GREEDY SEQUENTIAL()
1: while there is an uncolored vertex $v$ do
2:     color $v$ with the minimal color (number) that does not conflict with the already colored neighbors
```

Theorem 1.7

Algorithm 1.5 is correct and terminates in n steps. The algorithm uses at most $\Delta + 1$ colors.

Centralized Greedy Algorithm

We can assume that colors are numbered as $1, 2, \ldots$.
Initially, all nodes are not colored (marked as color 0)

```
ALGORITHM 1.5: GREEDY SEQUENTIAL()
1: while there is an uncolored vertex $v$ do
2:     color $v$ with the minimal color (number) that does not conflict with the already colored neighbors
```

Theorem 1.7

Algorithm 1.5 is correct and terminates in n steps. The algorithm uses at most $\underline{\Delta + 1}$ colors.

Upper Bound! Sometimes $\chi(G) \ll \Delta + 1$

Synchronous Distributed Algorithm

Definition 1.8 (Synchronous Distributed Algorithm)

In a synchronous distributed algorithm, nodes operate in synchronous rounds. In each round, each node executes the following steps:

- *Send messages to neighbors in graph (of reasonable size).*
- *Receive messages (that were sent by neighbors in step 1 of the same round).*
- *Do some local computation (of reasonable complexity).*

Centralized ⇒ Distributed

One fundamental problem is Symmetry Breaking

- When some nodes need to be selected to conduct computation or communication, how to distinguish one vertex form another (or a set of vertices from another set).
    - node IDs, but in some cases, nodes do not have unique IDs (anonymous systems).
    - Random numbers
    - Vertex coloring, Maximal Independent Set...

Distributed Version for Algorithm 1.5

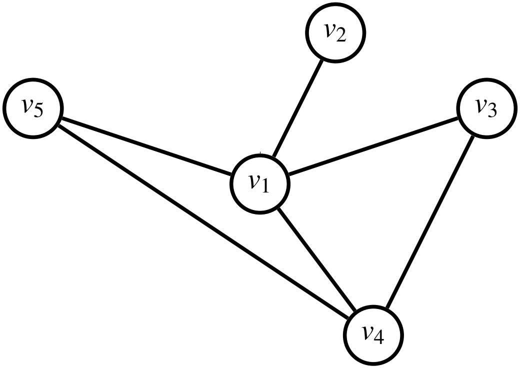
```
ALGORITHM 1.9: REDUCE()
1: Assume that initially all nodes have IDs
2: Each node $v$ executes the following code:
3: node $v$ sends its ID to all neighbors
4: node $v$ receives IDs of neighbors
5: while node $v$ has an uncolored neighbor with higher ID do
6:     node $v$ sends undecided to all neighbors
7:     node $v$ receives new decisions from neighbors
8: node $v$ chooses the smallest admissible free color
9: node $v$ informs all its neighbors about its choice
```

Distributed Version for Algorithm 1.5

```
ALGORITHM 1.9: REDUCE()
1: Assume that initially all nodes have IDs
2: Each node $v$ executes the following code:
3: node $v$ sends its ID to all neighbors
4: node $v$ receives IDs of neighbors
5: while node $v$ has an uncolored neighbor with higher ID do
6:     node $v$ sends undecided to all neighbors
7:     node $v$ receives new decisions from neighbors
8: node $v$ chooses the smallest admissible free color
9: node $v$ informs all its neighbors about its choice
```

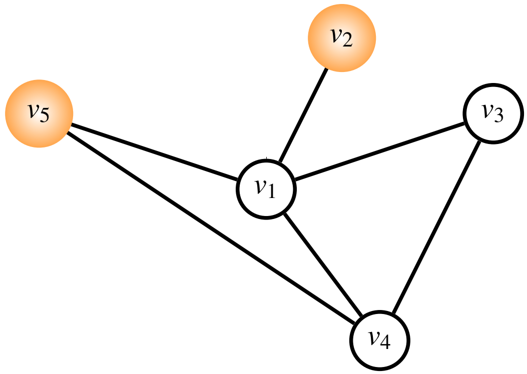
Color the node with the highest ID in its one-hop uncolored neighborhood.

Example



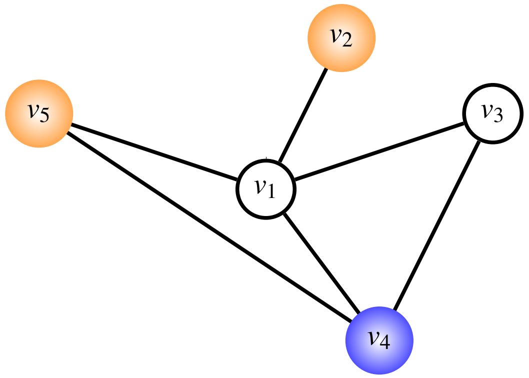
Round 0 (Initially)

Example



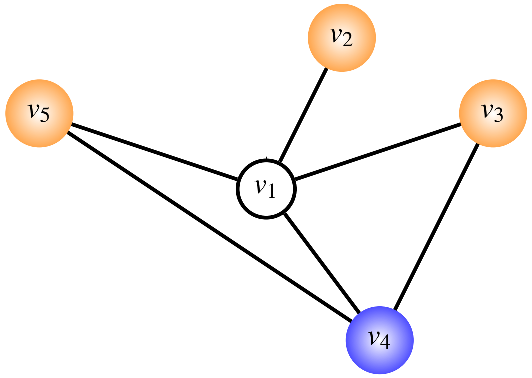
Round 1

Example



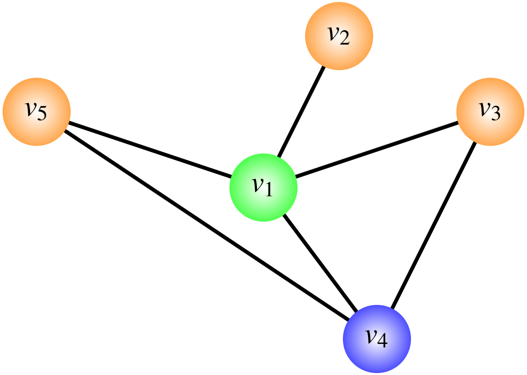
Round 2

Example



Round 3

Example



Round 4

Performance Analysis

Definition 1.11 (Time Complexity)

For synchronous algorithms (as defined in 1.8) the time complexity is the number of rounds until the algorithm terminates. The algorithm terminates when the last node terminates.

Theorem 1.12

Algorithm 1.9 is correct and has time complexity n. The algorithm uses at most $\Delta + 1$ colors.

Performance Analysis

Definition 1.11 (Time Complexity)

For synchronous algorithms (as defined in 1.8) the time complexity is the number of rounds until the algorithm terminates. The algorithm terminates when the last node terminates.

Theorem 1.12

Algorithm 1.9 is correct and has time complexity n. The algorithm uses at most $\Delta + 1$ colors.

Proof. Nodes choose colors that are different from their neighbors, and no two neighbors choose concurrently. In each round at least one node chooses a color, so we are done after at most n rounds.

Hardness of Vertex Coloring

- For general graphs, it is hard to compute the chromatic number.
  - approximating the chromatic number within $n^{1-\varepsilon}$
- For a planar graph,
  - whether it can be colored by 2 colors is polynomial time decidable.
  - whether it can be colored by 3 colors is NP-complete.
  - it can be definitely colored by 4 colors (Four Color Theorem).

Coloring Trees

Lemma 1.13

$\chi(Tree) \leq 2$

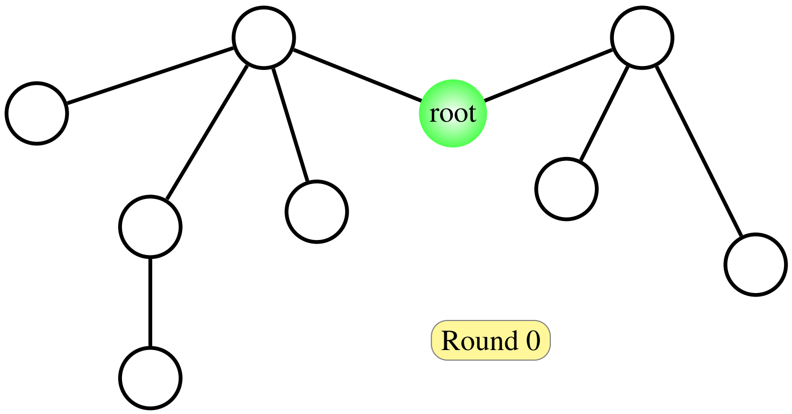
Proof. Call some node the root of the tree. If the distance of a node to the root is odd (even), color it 1 (0). An odd node has only even neighbors and vice versa.

Slow Tree Coloring

If we assume that each node knows its parent and children in a tree, this constructive proof gives a very simple algorithm:

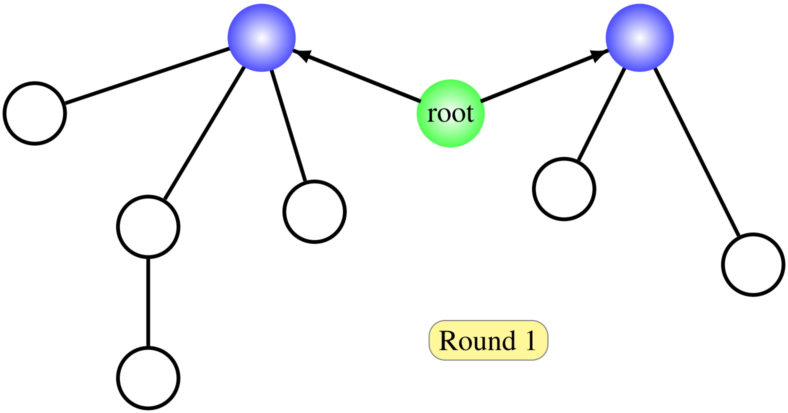
```
ALGORITHM 1.14: SLOW TREE COLORING()
1: Color the root 0, root sends 0 to its children
2: Each node $v$ concurrently executes the following code:
3: if node $v$ receives a message $c_p$ (from parent) then
4:     node $v$ chooses color $c_v = 1 - c_p$
5:     node $v$ sends $c_v$ to its children (all neighbors except parent)
```

Example

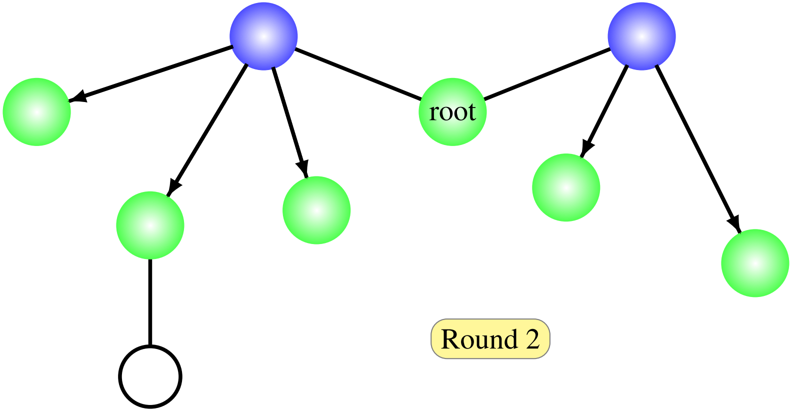


Round 0

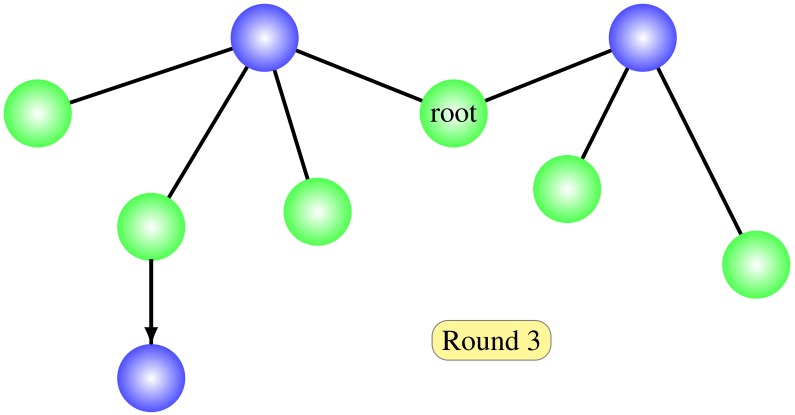
Example



Example



Example



Slow Tree Coloring

```
ALGORITHM 1.14: SLOW TREE COLORING()
1: Color the root 0, root sends 0 to its children
2: Each node $v$ concurrently executes the following code:
3: if node $v$ receives a message $c_p$ (from parent) then
4:     node $v$ chooses color $c_v = 1 - c_p$
5:     node $v$ sends $c_v$ to its children (all neighbors except parent)
```

- **Time Complexity**: height of the tree
    - Nice trees, e.g., balanced binary trees, have logarithmic height, that is we have a logarithmic time complexity
    - However, the height can reach n.
- Can we do better than the height of the tree, or logarithmic?

Log-Star

Definition 1.16 (Log-Star)

$\forall x \leq 2 : log^{*}x := 1$
$\forall x > 2 : log^{*}x := 1 + log^{*}(logx)$

- Log-star is an amazingly slowly growing function
    - Log-star of all the atoms in the observable universe (estimated to be 10^{80}) is 5.

6-Color Algorithm

- Idea:
  - Interpret vertex ID as color (we can assume the IDs are $1, 2, , n$)
  - Reduce the number of colors based on vertex ID manipulations

6-Color Algorithm

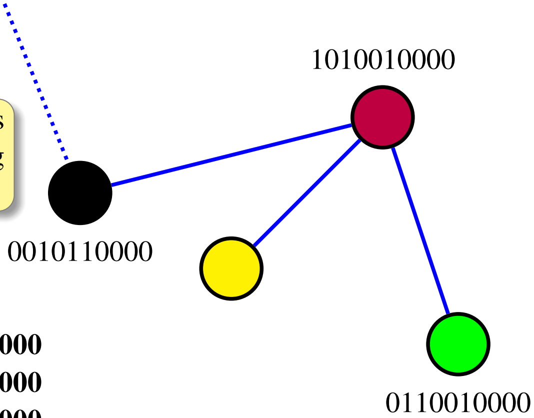
- Idea:
    - Interpret vertex ID as color (we can assume the IDs are $1, 2, , n$)
    - Reduce the number of colors based on vertex ID manipulations

```
ALGORITHM 1.17: 6-COLOR()
1:  Assume that initially the nodes have IDs of size $\log n$ bits
2:  The root assigns itself the label 0
3:  Each other node $v$ executes the following code
4:  send own color $c_v$ to all children
5:  repeat
6:      receive color $c_p$ from parent
7:      interpret $c_v$ and $c_p$ as bit-strings
8:      let $i$ be the index of the smallest bit where $c_v$ and $c_p$ differ
9:      the new label is $i$ (as bitstring) followed by the $i^{th}$ bit of $c_v$
10:     send $c_v$ to all children
11: until $c_w \in \{0, ..., 5\}$ for all nodes $w$
```

Example

Round 1

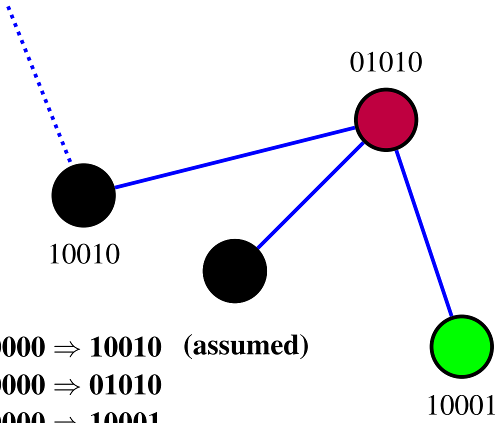
We count the bit positions from right to left, starting with 0



Grand-parent	**0010110000**
Parent	**1010010000**
Child	**0110010000**

Example

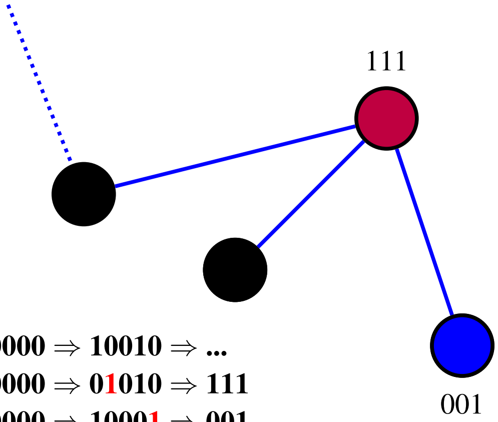
Round 2



Grand-parent	**0010110000 $\Rightarrow$ 10010**	**(assumed)**
Parent	**1010010000 $\Rightarrow$ 01010**	
Child	**0110010000 $\Rightarrow$ 10001**	

Example

Round 3



Grand-parent	**0010110000**	$\Rightarrow$	**10010**	$\Rightarrow$	**...**
Parent	**1010010000**	$\Rightarrow$	**01010**	$\Rightarrow$	**111**
Child	**0110010000**	$\Rightarrow$	**10001**	$\Rightarrow$	**001**

6-Color Algorithm

```
ALGORITHM 1.17: 6-COLOR()
 1: Assume that initially the nodes have IDs of size $\log n$ bits
 2: The root assigns itself the label 0
 3: **Each** other **node** $v$ executes the following code
 4: send own color $c_v$ to all children
 5: **repeat**
 6:     receive color $c_p$ from parent
 7:     interpret $c_v$ and $c_p$ as bit-strings
 8:     let $i$ be the index of the smallest bit where $c_v$ and $c_p$ differ
 9:     the new label is $i$ (as bitstring) followed by the $i^{th}$ bit of $c_v$
10:     send $c_v$ to all children
11: **until** $c_w \in \{0, ..., 5\}$ for all nodes $w$
```

- In each round, the parent and child have different colors (why?).
- The largest color used shrinks dramatically in each round:
  - the length of a nodes ID: not less than $i \to \log i + 1$

6-Color Algorithm

Theorem 1.18

Algorithm 1.17 terminates in $\log^* n + k$ time, when k is a constant independent of n.

- How do a node knows that all nodes have colors in range $[0-5]$?
- Why colors $6, 7 \ldots$ will not appear in the final solution?
- What happens if we do not know the root of the tree?
- Why using colors from $0-5$? Can we have a smaller # of colors?

6-Color Algorithm

Theorem 1.18

Algorithm 1.17 terminates in $\log^* n + k$ time, when k is a constant independent of n.

- How do a node knows that all nodes have colors in range $[0-5]$?
- Why colors $6, 7 \ldots$ will not appear in the final solution?
- What happens if we do not know the root of the tree?
- Why using colors from $0-5$? Can we have a smaller # of colors?

- Actually, with more tricks, we can color a tree with 3 colors in time $O(\log^* n)$
- For general graphs, we will study another distributed coloring technique later.

Summary

- For Vertex Coloring:
  - We can color it in time $O(n)$ using $\Delta + 1$ colors.
  - For Ring graph, the running time could be further improved to $O(\log^* n)$.
  - For general graph, the running time could be improved to $O(\log(n))$.
- For Tree Coloring:
  - We can color it using 3 colors in time about $O(\log^* n)$.
  - There is a simple method that can color it using 2 colors in time $O(h)$, height of the tree —- assuming we know the root of the tree.
- For Other Special Graphs
  - Growth bounded graphs: time complexity $O(\log^* n)$.
    - A graph where for each r-hop neighborhood of any node, the size of MIS is bounded by a function f(r). Function f is a given polynomial function.

Reference

- Book:
  - Leonid Barenboim and Michael Elkin, Distributed Graph Coloring: Fundamentals and Recent Developments, Synthesis Lectures on Distributed Computing Theory, July 2013.
- Basic technique of the log-star algorithm
  - R. Cole and U. Vishkin. Deterministic coin tossing and accelerating cascades: micro and macro techniques for designing parallel algorithms. In STOC, 1986.
- Some results on other special graphs, e.g., graphs with a constant degree, growth bounded graphs
  - Andrew V. Goldberg and Serge A. Plotkin. Parallel (+1)-coloring of constant-degree graphs. Inf. Process. Lett., 25(4):241-245, June 1987.
  - Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. On the Locality of Bounded Growth. In PODC, 2005
  - N. Linial. Locality in Distributed Graph Algorithms. SIAM Journal on Computing, February 1992.