Course 2.18.1: Distributed Algorithms for Networks Sleeping Model

Mikaël Rabie

1 The Awaken Complexity

1.1 $\Delta + 1$-Coloring in $O(\Delta)$ awaken rounds - Slide 3

Each node wakes up $\Delta + 1$ times. First at round 0 to learn the identifiers of the neighbors. Then at each round corresponding to its own identifier and the one of their neighbors. At the round corresponding to its own round, it learns the new color of its neighbors that have smaller identifier. It then chooses the smallest available color. After this round, each times it wakes up, it provides its new color to its awaken neighbor.

Note that it is actually not necessary to wake up in rounds of smaller identifiers, however local minimas will have to do it, keeping the $\Delta + 1$ complexity. The total number of rounds will be 1+M, where M is the maximal identifier in the system.

1.2 Reduce K colors in $\log K$ awake rounds - Slide 4

Assume we have a K-coloring of the graph. We build a perfectly balanced binary research tree containing the integers from 1 to K (adding nodes to reach the next power of 2 after K). At round 0, each node wakes up, learning about the colors in their neighborhood. After that, at round i, all nodes that are in the subtree of node with value i (i.e. node i and the ones below in the binary tree) wake up twice (i.e. each round is doubled).

At round i, nodes of color i will compute their final color. To do so, awaken nodes of color $\geq i$ learn the new color that nodes of color $< i$ have computed. Nodes of color i choose the smallest color they know is not in their neighborhood, and provides it to the other nodes (this is why we double each round).

To prove that this algorithm works, we need to be sure that when node of color i chooses its final color, it knows the final color of its neighbor who had a color $< i$. Let $j < i$.

- If j is below i in the tree, i learned the final colors of nodes with starting color j in round i.
- Otherwise, j is in another part of the tree. Let k be the lowest common ancestor of i and j (k can be j). If $k = j$, i learned the final color of j during the round j.

  Otherwise, we must have $j < k < i$, as we are on a research binary tree. Hence, during round k, nodes of initial color j have transmitted their final color while nodes of color i were awaken.

Each node is awaken as many times as its depth in the binary tree, which is $\log K$.

2 The $\log n$ Complexity

2.1 Full Knowledge of the Graph- Slide 5

In this section, I will also call DLT the intermediate trees, as long as they respect the increasing labels from root to leafs.

We assume that each node knows the maximal possible label (that actually depends of the maximal identifier and size of the graph). To perform a broadcast in a DLT, each node wakes up at two rounds, the one corresponding to its label x, and the one corresponding to its parent's label $p(x)$. Hence, as its parent as a smaller label, it will get information from it, and will be able to transmit the information to its own children at the round corresponding to its own label.

For convergecast (that consists in gathering the information from all nodes to some node), we will gather everything to the root of the DLT. Let M be an upper bound on the maximal label, nodes wake up at rounds $M-x$ and $M-p(x)$. Hence, at its round, it will get information gathered by its children, and it will transmit this information (plus its own information) to its parent at round $M-p(x)$.

2.2 Building a DLT- Slides 6

Here are some more details on the steps:

1. With a convergecast, the root will know all the neighboring DLTs (and each vertex is neighbor to what). It can know that way if there is one with smaller label.

2. During the convergecast, the root got also the full structure of the graph. With a broadcast, it will give the new distance to compute for every node according to the choice of the neighboring DLT and the corresponding node u.

3. A new convergecast allows roots to know if there tree got selected or not. In the second case, if no neighboring DLT had smaller identifier, it chooses any neighbor. To merge, the convergecast also gave the distance to the node v, hence it can broadcast the new identifier and distance to the nodes in its current DLT.

   We are sure that if a DLT u_1 chooses a DLT u_2, DLT did not choose any other DLT (as it had at least one neighbor with smaller identifier).

5. + 6. For the merges, we see that we maintain the structure of increasing labels through each new DLTs. Hence, a convergecast can be done for the root to learn the new structure, and then broadcast it.

3 $\Delta + 1$ Coloring

3.1 Trade-Off- Slide 8

We can observe that even though we can find awaken rounds that are lower than the usual number of rounds in the LOCAL model, we get the drawback that nodes run for a longer time. For example, the $(\Delta+1)$-coloring in $\Delta+1$ rounds use the maximal identifier total rounds.

On paths, we can notice that each communication round allows to perform two simplification steps of the Cole-Vishkin Algorithm, reducing twice by log the maximal identifier. One can then

wonder what are all the possible trade-offs between awaken complexity and round complexity. For example, by doing Linial's algorithm to reach $O(\Delta^2)$ colors in $O(\log^* n)$ awaken and regular rounds, we can then reduce the number of colors in $O(\log \Delta)$ awaken rounds to reach a $\Delta + 1$-coloring.

3.2 Coloring in $O(\sqrt{\log n} \log^* n)$ awaken rounds - Slides 9-14

The full paper for this construction is [1] (it solves more general problems, including $(\Delta + 1)$-coloring). Feel free to contact me for more details.

About **Slide 13**: We repeat the process as many times as needed to have given colors to clusters. When a cluster becomes colored, it ends with final color being $(i, \gamma'_{i-1}(l_{i-1})$, i.e. they also remember in which level they got their color. It ensures that at the end, two neighboring clusters do not have the same γ. Indeed, either they left on two different levels, or they left on the same level and had different colors. As they are uniquely labeled, we repeat the algorithm on the other clusters, using the property from Slide 12.

References

[1] Alkida Balliu, Mohsen Ghaffari, Fabian Kuhn, Augusto Modanese, Dennis Olivetti, Mikaël Rabie, Jukka Suomela, and Jara Uitto. Shared randomness helps with local distributed problems. *CoRR*, abs/2407.05445, 2024.