Course 2.18.1: Distributed Algorithms for Networks LOCAL Variants

Mikaël Rabie

In this course, a problem is said greedy if we can solve it sequentially by looking at the already computed outputs at constant distance.

1 Volume Complexity - Slides 2-4

To solve any greedy problem in time $O(f(\Delta) \times \log^* n)$:

- We compute the Δ^2-coloring for the nodes at distance at most Δ^2. We have up to Δ^{Δ^2} nodes to consider.
- We orient the edges from bigger color to smaller color (each path if of length at most Δ^2).
- We only consider nodes accessible from u with the orientation. We pick each node one after another, making sure to only pick a node if all its neighbors with smaller color already has an output.

To simulate the algorithm with the new identifiers, it is the same technique as in the toroidal grid with $o(n)$ algorithms: we compute some distance-N coloring, and use the new colors as fake identifiers, and the algorithm will not see any contradiction.

2 Mendability - Slide 6

To 4-color a grid, a greedy algorithm does not work, as we can get a situation where a node has all 4 possible colors in its neighborhood. However, in that case, we can choose a cycle of length 4, and remove the output. Each node has at least 2 possible colors to select. There is always a choice such that all 4 nodes get colored properly.

2.1 From mendable to LOCAL - Slide 7-8

We can compute the output of each node one color after another. We have $O(\Delta^{2T})$ colors. To compute the output of nodes of color i, we need to have computed the output (and do the eventual mendability changes) needed for the $i-1$ previous colors, and then spend T rounds (to compute the output and do eventual changes).

A distance-$(2T+1)$ coloring can be computed in $O(T \log^* n)$ rounds on bounded degree graphs.

Consider the problem to 3-color with $\{1, 2, 3\}$ or 2-color with $\{A, B\}$ a path. This problem has $O(\log^* n)$ complexity, as it suffices to 3-color the path. However, given a partial solution, if there is a A at distance $T+1$ on the left and a B at distance $T+1$ on the right, there is no way to mend in the middle.

2.2 Mendability on Trees - Slide 9

We can build a tree and select the colors on the leafs such that there is only one way to color everything else (in particular, if your two children have different color, you must take the 3rd one).

The variant where we force not to have too long mixed nodes connected components ensure that, when a node needs to choose its output, it can recolor nodes at distance up to $2k$ to separate the mixed components. In particular, it ensures that nodes at the same level have the same color, "killing" that way mixed nodes.

3 Landscape of LOCAL Variants - Slide 10

The slide this image was taken from can be found in https://jukkasuomela.fi/doc/dagstuhl-2024-11-20.pdf.

We see several models. $M_1 \to M_2$ means that a problem with complexity $O(f(n))$ in M_1 has complexity $O(f(n))$ in M_2. The goal is to find model gaps, i.e. two models M_1 and M_2, and a problem Π such that the complexity of Π is $\Omega(f(n))$ in M_1 and $o(f(n))$ in M_2.

3.1 SLOCAL Model - Slide 11

In the SLOCAL model, we assume also that the algorithm can put more than the output on each node. In particular, it can give information such as output pre-computed for other nodes. Greedy problems like $(\Delta+1)$-coloring gets complexity $O(1)$, which gives a separation between LOCAL and SLOCAL. With Sinkless Orientation, we give a stronger separation, from $\Theta(\log n)$ to $O(\log\log n)$.

3.2 Randomized Sinkless Orientation - Slide 13

The 3 first steps are done in constant time. Note that edges that get unoriented are only between nodes of Type I or II. The proof that each connected component is of polylog size can be found in [2]. The key elements of this proof are:

- The probability that a node is of type I or III is very low
- Two nodes of at distance at least 2 have independent probabilities to be type I or III
- Build an virtual graph of Type I and III nodes where they are connected if their distance is between 2 and 4, you can make a spanning tree in each connected component
- A connected component of bad nodes is in the closed neighborhood of these spanning trees
- The probability of a spanning tree to be of size $\Omega(\log n)$ is $1/poly(n)$

The deterministic algorithm running on time $O(\log N)$, N being the size of the connected component, running it on the bad nodes will have running time $O(\log\log N)$ with high probability.

It is explained in [2] how to deal with the case where the maximal degree is lower, and when the graph is not Δ-regular.

3.3 SLOCAL Sinkless Orientation - Slide 14-16

For the composition, here is more details on the $T_{\mathcal{A}} + 2T_{\mathcal{B}}$ complexity:

- To compute $\mathcal{B}(v)$, we need to compute the output of $\mathcal{A}$ on $N_{T_{\mathcal{B}}}(v)$.
- Hence, $\mathcal{C}$ will compute in advance the outputs of $\mathcal{A}$ for some nodes that will be sequentially treated later on. When it computes outputs for some nodes while processing u, it will store those outputs on u.
- It implies that, if the output $\mathcal{A}(u)$ has been pre-computed, it was by a node at distance at most $T_{\mathcal{B}}$ from u. $\mathcal{C}$ needs to look at distance $T_{\mathcal{B}}$ of $N_{T_{\mathcal{B}}}(v)$ to check for pre-computed outputs.
- if we want to be more precise, the complexity is $T_{\mathcal{B}} + \max(T_{\mathcal{A}}, T_{\mathcal{B}})$

In the proof by induction, as u and v are in different components, and both have at least b marked edge toward themselves, their components are of size $\geq 2^b$. As those components must be different (otherwise, you have a cycle where an edge could not have been marked), we indeed have a component of size $\geq 2^{b+1}$ after the addition of edge $v \to u$.

Note that this algorithm has complexity $O(1)$, as each edge only needs to know the status and indegrees of both endpoints.

For the composition of the 3 algorithms, we have the following complexities:

1. $O(T) = O(\log\log n)$ to compute I, as it is a greedy algorithm, where we check if a node at distance $2T + 1$ is in I before deciding to add or not the current node in I.
2. The algorithm has complexity $O(1)$, but it is ran on a virtual graph where each node represents a component of diameter $O(T)$. Hence, this round has complexity $O(T) = O(\log\log n)$
3. The algorithm runs inside each cluster, which is of size $O(T) = O(\log\log n)$

By composition, we get complexity $O(\log\log n)$.

3.4 Shared Randomness Separation - Slide 17

The exact construction can be found in [1]. In particular, their construction works on any graph, instead of having to ensure that there is a grid structure with a binary tree on each column. This construction separates different models.

A node accepts the output if:

- It is not on the rightmost column, and its output is the same as the output on its right.
- It is on the rightmost column, and its output is the same as its input.
- It is on the rightmost column, points up, and either its up neighbor points up or its output is the same as its input.
- It is on the rightmost column, points down, and either its down neighbor points down or its output is the same as its input.

If we do not have binary tree on each column, randomness does not help. The full row needs to provide the same output, and the output must depend on the rightmost column. This gives a direct $\Omega(\sqrt{n})$ lower bound.

When we add the binary trees, each node in a same column can know its position. This does not help without shared randomness. All nodes can coordinate to choose deterministically (or with high probability) the same output depending on your position on the column. However, by looking at what is the high probability output on the leftmost column, we give the reverse input on the rightmost column, and the output is not valid.

However, with shared randomness, each column can use the $\sqrt{n}$ first bits to put the corresponding output on the column. The probability that the output is valid on a row is $\frac{1}{2}$, hence the probability of success is $1 - \frac{1}{2^{\sqrt{n}}}$.

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.

[2] Mohsen Ghaffari and Hsin-Hao Su. Distributed degree splitting, edge coloring, and orientations. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2505–2523. SIAM, 2017.