ECS-189A Special Topics in TCS April 11, 2023

Lecture 3

Lecturer: Slobodan Mitrović

1 Introduction

In the last lecture, we discussed an algorithm for computing connected components (CC) in MPC when the memory per machine is n^{1+c} for a constant $c > 0$. That algorithm solves the problem in $O(1/c) = O(1)$ many MPC rounds. However, when the memory per machine is $2n$, one can show that the same algorithm takes $O(\log n)$ rounds. On the other hand, when the memory per machine is sublinear in n, the algorithm we discussed last lecture might not even terminate.

In this lecture, we will design a different algorithm that counts CC in $O(\log n)$ rounds even when the memory per machine is less than n. The algorithm we design can easily be turned into one finding CC as well.

2 Preliminaries

2.1 Notation

We will use $G = (V, E)$ to denote a graph on vertex set V and edge set E. If not stated otherwise, we will use n to denote $|V|$ and m to denote $|E|$.

We will use notation $\tilde{O}(f)$ to hide poly-logarithmic factors in f, i.e., $\tilde{O}(f) = O(f \cdot \text{poly} \log f)$.

By saying that an event $\mathcal{E}$ happens *with high probability* (whp), we refer that $\Pr[\mathcal{E}] \geq 1 - n^{-c}$ for some constant $c \geq 1$.

2.2 Probability tools

Theorem 1 (Markov's inequality). *If X is a nonnegative random variable and $a > 0$, then*

$$\Pr[X \geq a] \leq \frac{\mathbb{E}[X]}{a}.$$

3 Leader-Contraction

In this section, we will present an algorithm for *counting* connected components in the sub-linear MPC memory regime. We point out that this algorithm can easily be extended to outputting the corresponding connected components as well.

Observe that counting the number of connected components when there is no edge is trivial – the number of CC equals the number of vertices. Inspired by this, given an input graph G, our algorithm will produce a graph H such that H has fewer edges than G, but the number of CC in G and H are identical. More specifically, we will obtain connected components by iteratively contracting some neighboring vertices. That algorithm is called Leader-Contraction, and the complete pseudo-code is outlined as Algorithm 1. The algorithm itself does not dive into details of MPC implementation; instead, an outline of MPC implementation is given in Section 3.4.

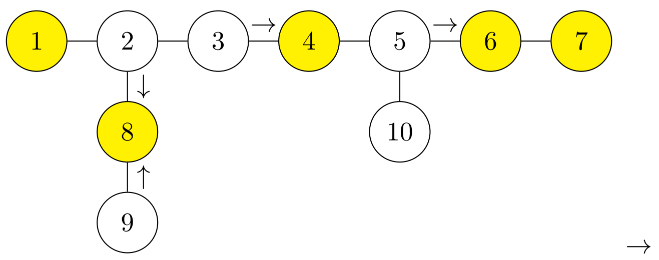
```
    Input   : Graph $G = (V, E)$
              Space per machine $S$
1   Function Leader-Contraction($G = (V, E)$):
2       $H \leftarrow G$
3       Independently of other vertices, each vertex of $H$ is marked as a leader with probability
          $1/2$.
4       Contract each non-leader vertex of $H$ to its adjacent leader, if there is any. In case of
          ties, choose the leader with the largest label.
5       return $H$
6   $\tilde{G} \leftarrow G$
7   while there is an edge in $\tilde{G}$ do
8       $\tilde{G} \leftarrow$ Leader-Contraction($\tilde{G}$)
9   return the number of vertices in $\tilde{G}$
```

Algorithm 1: An algorithm for computing connected components. In Section 3.4 we outline its implementation in the sub-linear memory regime of MPC.

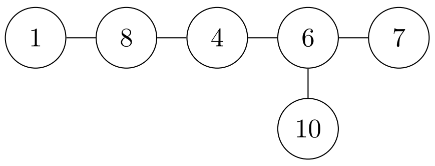
Before we proceed with the analysis of Algorithm 1, we provide several examples.

3.1 Example 1

- Round 1 (leaders are marked in yellow)

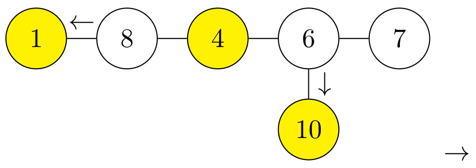


(a) Before contraction.

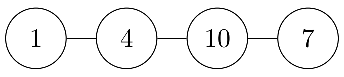


(b) After contraction.

- Round 2

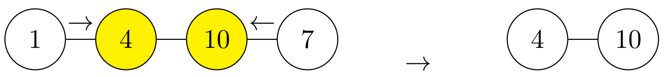


(a)



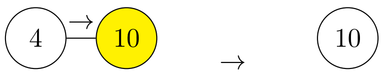
(b)

- Round 3



(a) (b)

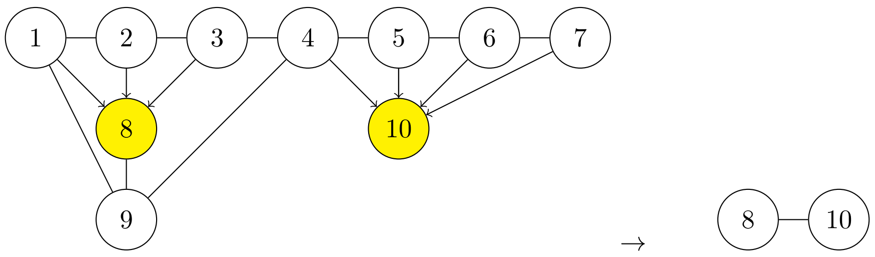
- Round 4



(a) (b)

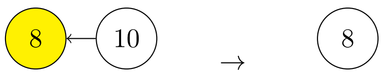
3.2 Example 2

- Round 1 (leaders are marked in yellow)



(a) (b)

- Round 2



(a) (b)

3.3 Correctness

Showing that Algorithm 1 outputs the number of CC is almost direct from the algorithm design. Observe that the final version of $\tilde{G}$ has no edge, so outputting the number of vertices it has is indeed the number of CC that graph has. To see that it corresponds to the number of CC in the input graph G, observe that contracting a non-leader vertex to u to a leader vertex v can be seen as re-labeling *all* the vertices with label u to label v. This re-labeling process is then performed while at least two or more different labels are in a CC of G.

3.4 Implementation in MPC

The only non-trivial part needed to implement Algorithm 1 is a method that contracts a non-leader vertex to a leader one. We will skip those technicalities in this lecture and only provide a high-level idea.

To contract vertex u to vertex v, we would like to take every edge $\{u, w\}$ and re-label it to $\{v, w\}$. In MPC, this can be done by sorting all the edges with respect to one of the endpoints and then applying all needed re-labels simultaneously using a data structure similar to an interval/segment tree. The same process is then applied to the other endpoint of each edge.

Procedures for sorting and performing search/updates in the corresponding data structure in $O(1/c)$ rounds can be found in [GSZ11, Section 4].

3.5 Round complexity

Theorem 2. *Algorithm 1 runs in $O(\log n)$ rounds whp.*

To show Theorem 2, we first prove the following claim.

Lemma 3. *Let G be an input to* Leader-Contraction *and $\tilde{G}$ be its output. Let z and $\tilde{z}$ be the number of non-isolated vertices in G and $\tilde{G}$, respectively. Then,*

$$\mathbb{E}\left[\tilde{z}\right] \le \frac{3}{4} z.$$

Proof. Let A be the set of z non-isolated vertices in G. Let X_v be an indicator variable equal 1 if a non-isolated vertex $v \in A$ is not contracted. Then, we have

$$\mathbb{E}\left[\tilde{z}\right] = \mathbb{E}\left[\sum_{v \in A} X_v\right] = \sum_{v \in A} \mathbb{E}\left[X_v\right].$$

A vertex v is contracted if (1) v is **not** chosen as a leader, and (2) at least one of its neighbors is chosen as a leader. We have $\Pr\left[v \text{ is chosen as a leader}\right] = 1/2$. Since all leader/non-leader choices are made independently across vertices, we also have $\Pr\left[\text{at least one neighbor of } v \text{ is chosen as a leader}\right] = 1 - (1/2)^{d(v)} \ge 1/2$; recall that $d(v) \ge 1$ as v is not isolated. This now implies

$$\Pr\left[v \text{ is contracted}\right] \ge \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}.$$

Hence,

$$\Pr\left[v \text{ is not contracted}\right] = 1 - \Pr\left[v \text{ is contracted}\right] \le \frac{3}{4}.$$

This implies that

$$\mathbb{E}\left[X_v\right] \le \frac{3}{4},$$

and hence our claim follows. □

We want to use Lemma 3 to prove Theorem 2. Let $\tilde{z}_t$ be the number of non-isolated vertices in $\tilde{G}$ after t invocations of Leader-Contraction. By repeatedly applying Lemma 3, for $k = 2\frac{\log_2 n}{\log_2 (4/3)}$ we have $\mathbb{E}\left[\tilde{z}_k\right] \le \left(\frac{3}{4}\right)^k n = n^{-1}$.

By applying Markov's inequality, i.e., Theorem 1, we have that

$$\Pr\left[\tilde{z}_k \ge 1\right] \le \frac{1}{n}.$$

So, with high probability, we have that $\tilde{z}_k < 1$, which proves Theorem 2. **Remark:** By replacing the multiplicative factor 2 by d we would get the probability of failure to be $n^{-(d-1)}$ instead of n^{-1}.

References

[GSZ11] Michael T Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the mapreduce framework. In *ISAAC*, volume 7074, pages 374–383. Springer, 2011.