Lecture 19

Massively Parallel Algorithms: Maximal Matching

Course: Algorithms for Big Data

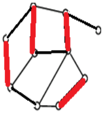
Instructor: Hossein Jowhari

Department of Computer Science and Statistics
Faculty of Mathematics
K. N. Toosi University of Technology

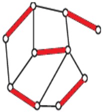
Spring 2021

Maximum Matching in Graphs

Graph G with m edges on n vertices.
The size of maximum matching = $m^*(G)$



maximal matching



maximum matching

bipartite matching

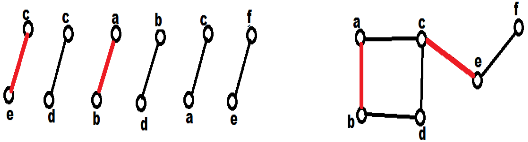
Applications:

- Building block in designing algorithms
- Job scheduling
- Transportation
- Image Recognition

Algorithms:

- A $O(\sqrt{n}m)$ time algorithm for finding a maximum cardinality matching. Hopcroft–Karp–Karzanov 1973, Micali-Vazirani 1980

Greedy algorithm for maximal matching



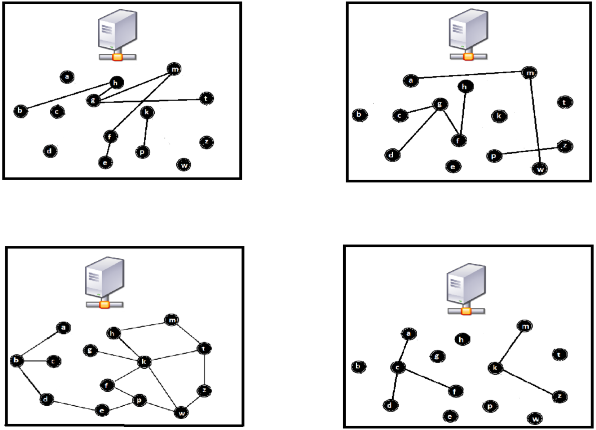
Input: sequence of edges $e_1, \ldots, e_m$
Let $T = \varnothing$ be an empty matching.
For $i \leftarrow 1$ to n:
    $e_i \leftarrow$ next edge
    If e_i intersects with an endpoint of a matching edge in T
        Ignore e_i
    Otherwise
        Add e_i to T

Space Usage $|T| = O(n)$

$|T| \geq \frac{1}{2} m^*(G)$

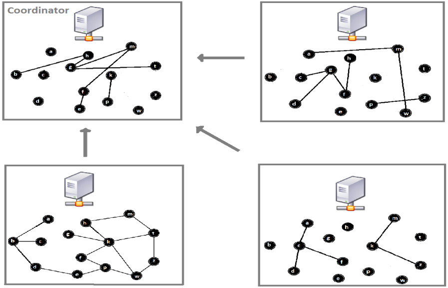
Maximum Matching: MPC

- Every machine gets a subset of the edges.
- Memory size of each machine $S \leq n^{1+\epsilon}$ words.
- Each machine sends/receives at most S words in each round.



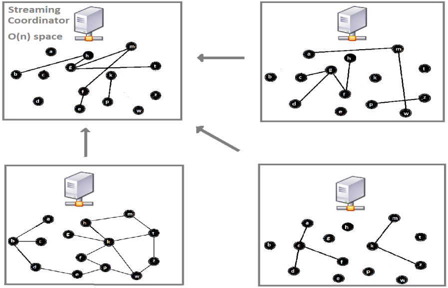
Lets examine a few strategies:

- Strategy A: All machines send their input to a single machine (coordinator). The coordinator collects all data and solves the problem locally.



Not feasible. Input size N could be $\Omega(n^2)$ while memory capacity of the coordinator is $n^{1+\epsilon}$.

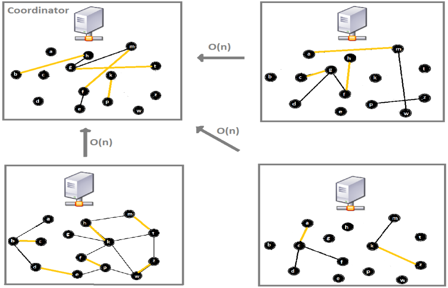
- Strategy B: We fix the strategy A by using the greedy algorithm. Since we are interested in the maximal matching, the coordinator does not need to store the entire data. Instead, he uses the greedy algorithm that takes $O(n)$ space.



How many rounds does it take?

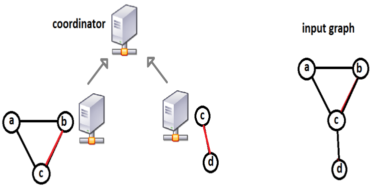
$\frac{m}{n^{1+\epsilon}} = O(n^{1-\epsilon})$. (Since the coordinator cannot receive more than $n^{1+\epsilon}$ words in a single round.)

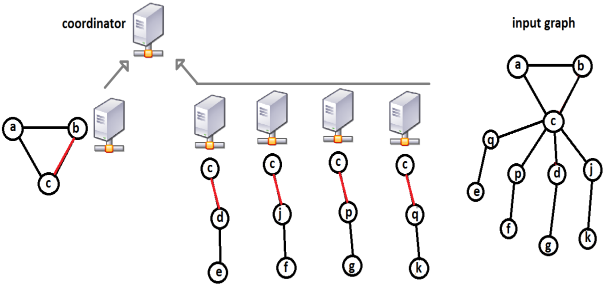
- Strategy C: What if (instead of transmitting all edges) each machine compute a local maximal matching and send its maximal matching to the coordinator?



This takes $\frac{Mn}{n^{1+\epsilon}} = \frac{M}{n^{\epsilon}}$ rounds but the output might NOT be a maximal matching.

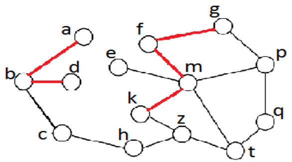
Bad example for strategy C:





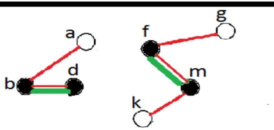
Filtering strategy

Lattanzi, Moseley, Suri, Vassilvitskii. Filtering: a method for solving graph problems in mapreduce. SPAA 2011.



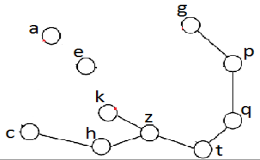
a random subset of edges
are selected in each round

the marked edges are
transmitted to the coordinator



the coordinator computes a
maximal matching of the
random subset

then announces the matched
vertices to all machines



the machines remove the
matched vertices from
the graph

the process repeats with
the remaining edges

Algorithm in full detail:

Let $G_r = (V, E_r)$ be the graph at round r.
$G_0 = (V, E) = \text{input graph}$. We assume the coordinator's input is empty. Let m_r be the number of edges in E_r. $m_0 = m$

In the r-th round:

- The machines marks their local edges with probability $p = \frac{n^{1+\epsilon}}{2m_r}$ (independently).
- The machines send their marked edges to the coordinator.
- The coordinator computes a maximal matching of the marked edges and announced the matched vertices to all.
- The machines discard the edges on newly announced matched vertices.

The process stops when $m_r \leq n^{1+\epsilon}$. At this point all remaining edges are sent to the coordinator.

Analysis

Lemma 1: With high probability, in each round the number of marked edges does not exceed $n^{1+\epsilon}$.

Proof: Each edge is marked independently with probability $p = \frac{n^{1+\epsilon}}{2m_r}$. Therefore in expectation number of marked edges is

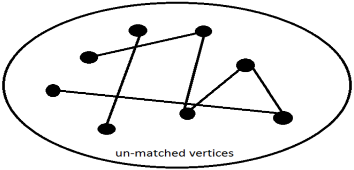
$$m_r \times \frac{n^{1+\epsilon}}{2m_r} = \frac{n^{1+\epsilon}}{2}$$

Using Chernoff bound, one can show that

$$Pr(\text{ number of marked edges } > n^{1+\epsilon}) \leq n^{-c}$$

Lemma 2: With high probability, the number of remaining edges in the end of round r is at most $\frac{10m_r}{n^\epsilon}$.

Proof: Let U be the un-matched vertices in the end of round r.



Observation: There cannot be any marked edge between the vertices in U.

Let F be a subset of V where the number of edges between the vertices in F is at least $\frac{10m_r}{n^\epsilon}$.

We show there is a marked edge between the vertices in F with exponentially high probability.

$$Pr(\text{all edges in F are unmarked }) \leq (1-p)^{\frac{10m_r}{n^\epsilon}} \leq e^{-p\frac{10m_r}{n^\epsilon}} \leq 2^{-5n}$$

Using the union bound, the probability that there is such a subset like F is at most $2^n \times 2^{-5} = 2^{-4n}$.

Therefore there will be at most $\frac{10m_r}{n^\epsilon}$ edges remained in the end of round r. $\square$

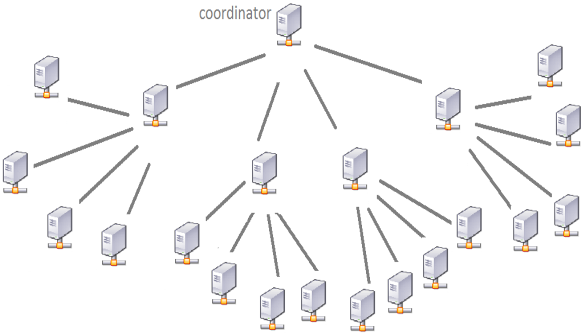
Lemma 3: Number of rounds is bounded by $O(\frac{1}{\epsilon})$.

Proof: In each round, the number of edges drops by factor of $\frac{10}{n^\epsilon}$. Therefore after $\log_{(n^\epsilon/10)} n^2 = O(\frac{1}{\epsilon})$ rounds, the number of edges goes below $n^{1+\epsilon}$.

Communication details

In the end of each round, the machines need to know m_r (the number of remaining edges in the graph). Also the newly matched vertices need to announced.

This can be done using a broadcast tree in constant number of rounds.



Related Results and References

Theorem There is a massively parallel algorithm for finding a maximal matching in the $S = O(n^{1+\epsilon})$ regime that finishes in $O(\frac{1}{\epsilon})$ number of rounds.

Theorem There is a massively parallel algorithm for finding a $O(1)$-approximate maximum matching in the $S = \tilde{O}(n)$ regime that finishes in $O(\log \log n)$ number of rounds.

Reference Mohsen Ghaffari. Massively Parallel Algorithms (book draft). Computer Science, ETH Zurich. 2019