University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
Z. Parsaeian, G. Schmid

Distributed Graph Algorithms
Sample Solution Exercise Sheet 5

Exercise 1: Exact Maximum Matching in Bipartite Graphs *(20 Points)*

In this exercise, we want to design a distributed algorithm in the CONGEST model for computing an **exact maximum (unweighted) matching** that runs in $O(s^\star \log s^\star)$ rounds, where $s^\star$ is the size of a maximum matching. We first recall the notion of an *augmenting path*.

Augmenting Paths: For a given matching M of a graph $G = (V, E)$, an augmenting path $P = u_0, u_1, \ldots, u_{2t+1}$ is an odd-length path that starts and ends in an unmatched node and that alternates between edges not in M and edges in M. Given an augmenting path P, one can obtain a matching M' of size $|M'| = |M| + 1$ by flipping the role of each edge along the path (i.e., edges not in M are added to the matching and edges in M are removed from the matching). A matching M is a maximum matching if and only if there is no augmenting path w.r.t. M.

We will develop the maximum matching algorithm in several steps. If you are not able to prove the claim of one of the steps, you can just assume it is true and use it in the susequent steps.

(a) Let $G = (V, E)$ be a graph with maximum matching size $s^\star$. Prove that if a given matching M has size $|M| = s^\star - k$ for some integer $k \geq 1$ then there exists an augmenting path of length at most $\frac{2s^\star}{k}$.

Hint: Consider the symmetric difference between M and some maximum matching of size s^*.

(b) Given a bipartite graph $G = (V, E)$ with diameter D, give a CONGEST algorithm to compute the bipartition of the graph in $O(D)$ rounds (i.e., partition the nodes V in to V_L and V_R such that all edges are between V_L and V_R).

(c) Given a graph $G = (V, E)$ with maximum matching size $s^\star$, give a $O(D + s^\star)$-rounds CONGEST algorithm to determine an upper bound $\hat{s}$ on $s^\star$ such that $s^\star \leq \hat{s} \leq 2s^\star$.

(d) Given a graph $G = (V, E)$ with maximum matching size $s^\star$ and diameter D, show that it always holds that $s^\star = \Omega(D)$. As a consequence, all the steps up to here require at most $O(s^\star)$ rounds.

(e) Assume that that we are given a bipartite graph $G = (V_L \cup V_R, E)$, a matching M of G, and an integer parameter L. You can further assume that all nodes of G know if they are in V_L or in V_R. Give an $O(L)$-round CONGEST algorithm that computes a new matching M' with the following properties. If G has an augmenting path of length at most L w.r.t. M, then $|M'| > |M|$. Otherwise, the matching has to remain the same, i.e., $M' = M$.

(f) We now have all the necessary tools to build our algorithm. Starting from an empty matching $M = \emptyset$, the algorithm operates in $\log_2 \hat{s}$ phases $i = 1, 2, \ldots, \lceil \log_2 \hat{s} \rceil$. Phase i consists of $\lceil \hat{s}/2^i \rceil$ interations. In each of those iterations, we run the algorithm of (e) with parameter $L_i = 2^{i+1}$.

Show that this algorithm computes a maximum matching in time $O(s^\star \log s^\star)$.

Hint: Show that at the end of phase i, the matching has size at least $s^\star - s^\star/2^i$.

Sample Solution

(a) Augmenting Path Length Bound

Claim: Let s^* be the size of a maximum matching. If a matching M has size $|M| = s^* - k$ for some integer $k \geq 1$, then there exists an augmenting path of length at most $\frac{2s^*}{k}$.

Proof. Let M^* be a maximum matching ($|M^*| = s^*$). Consider the symmetric difference graph $G_\Delta = (V, M \oplus M^*)$, where edges are those present in exactly one of the matchings. The maximum degree in G_Δ is 2, so it consists of isolated vertices, alternating cycles, and alternating paths.

- **Cycles:** Must contain an equal number of edges from M and M^*. They do not contribute to the difference in size.
- **Paths:** Can be classified by their endpoints:
  1. *Even paths:* Equal number of edges from M and M^*.
  2. *Augmenting paths for M:* Start and end with edges from M^*. Length is odd. Contains x edges from M and $x+1$ edges from M^*.
  3. *Augmenting paths for M^*:* Start and end with edges from M.

Since M^* is maximum, it admits no augmenting paths. Thus, there are no paths of type 3. The deficiency $k = |M^*| - |M|$ is exactly equal to the number of paths of type 2. Let these k augmenting paths be $P_1, \ldots, P_k$.
These paths are vertex-disjoint. The total number of edges from M^* in these paths is $\sum_{i=1}^{k} \frac{|P_i|+1}{2}$. This sum cannot exceed $|M^*| = s^*$.

$$\sum_{i=1}^{k} \frac{|P_i| + 1}{2} \leq s^* \implies \sum_{i=1}^{k} (|P_i| + 1) \leq 2s^*$$

$$\sum_{i=1}^{k} |P_i| \leq 2s^* - k < 2s^*$$

The sum of the lengths of the k paths is less than $2s^*$. By the pigeonhole principle, the minimum length path must be strictly less than the average:

$$\min_i |P_i| < \frac{2s^*}{k}$$

□

(b) Distributed Bipartition in $O(D)$

Algorithm:

1. **Root Selection:** If a leader is not given, elect a root r (e.g., node with minimum ID) via flood/echo in $O(D)$.
2. **BFS Construction:** Root r initiates a BFS.
   - Round 0: r sets $L(r) = 0$ and broadcasts.
   - Round t: Unvisited nodes receiving a message set $L(v) = t$.
3. **2-Coloring:**
   - If $L(v)$ is even, v joins set V_L.
   - If $L(v)$ is odd, v joins set V_R.

Since the graph is bipartite, edges only exist between levels i and $i+1$, ensuring proper coloring. The BFS completes in exactly D rounds.

(c) Approximation of s^* in $O(D + s^*)$

To schedule the algorithm, we need an estimate $\hat{s}$ such that $s^* \leq \hat{s} \leq 2s^*$.
Algorithm:

1. **Compute Maximal Matching:** Run a distributed maximal matching algorithm (e.g., Fischer's or Luby's algorithm). A maximal matching M_{max} satisfies $|M_{max}| \geq s^*/2$. This typically takes polylogarithmic time, which is well within $O(s^*)$.
2. **Aggregate Size:** Count $|M_{max}|$ by aggregating values up the BFS tree constructed in part (b). This takes $O(D)$ rounds.
3. **Broadcast Estimate:** Root sets $\hat{s} = 2|M_{max}|$ and broadcasts it.

Correctness: Since $s^*/2 \leq |M_{max}| \leq s^*$, it follows that $s^* \leq 2|M_{max}| \leq 2s^*$.

(d) Lower Bound $s^* = \Omega(D)$

$s^* \geq D/2$.

Proof. Let $P = (v_0, v_1, \ldots, v_D)$ be a shortest path between two nodes at distance D. Since it is a shortest path, it is simple. Consider the set of edges $M_P = \{(v_{2i}, v_{2i+1}) \mid 0 \leq 2i < D\}$. These edges are non-adjacent. Thus M_P is a valid matching. The size of this matching is $\lceil D/2 \rceil$. Since s^* is the size of the maximum matching, $s^* \geq |M_P| \geq D/2$. Therefore, $O(D) \subseteq O(s^*)$. □

(e) The `SetupPath(L)` Subroutine

Goal: Find an augmenting path of length $\leq L$ in $O(L)$ rounds.
Algorithm:

1. **Alternating BFS (L rounds):**
   - All free nodes start a BFS token containing their ID.
   - Tokens traverse edges alternatingly: from V_L free nodes, they traverse Unmatched $\rightarrow$ Matched $\rightarrow$ Unmatched...
   - Nodes store the `SourceID` and `Parent` pointer of the first token they receive.
2. **Collision Detection:**
   - If a node receives tokens from different sources (implying a path Free $\rightarrow \cdots \rightarrow u{-}v \leftarrow \cdots \leftarrow$ Free), an augmenting path is detected.
   - The collision generates a Path ID: pair of Source IDs.
3. **Conflict Resolution (Locking):**
   - To ensure disjointness, collisions propagate a "Lock" message back to the sources.
   - If a node lies on multiple detected paths, it only forwards the Lock for the path with the lexicographically smallest Path ID.
4. **Augmentation:**
   - Sources that receive a Lock confirm the path by sending a "Flip" message.
   - Edges on the path invert their status in M.

Total runtime is $O(L)$ for search + $O(L)$ for locking + $O(L)$ for flipping.

(f) The Complete Algorithm

Schedule:

1. Initialize $M = \emptyset$. Compute $\hat{s}$ (from part c).
2. Run Phases $i = 1$ to $\lceil \log_2 \hat{s} \rceil$:
   - Parameter: Max search depth $L_i = 2^{i+1}$.
   - Repetitions: Run `SetupPath`(L_i) for $K_i = \lceil \frac{\hat{s}}{2^i} \rceil$ iterations.

Complexity Analysis: Total rounds $T = \sum_i (\text{Iterations}_i \times \text{Cost}_i)$: Since $\hat{s} \leq 2s^*$, the total complexity is $O(s^* \log s^*)$.
Correctness: At phase i, we aim to reduce the deficiency to $s^*/2^i$. As long as the deficiency is larger than this, Part (a) guarantees an augmenting path exists of length $\leq \frac{2s^*}{s^*/2^i} = 2^{i+1}$. The subroutine is guaranteed to find such paths. We allocate enough iterations to reduce the deficiency to the target of the next phase.