

Improved Distributed Approximate Matching

Zvi Lotker Boaz Patt-Shamir Seth Pettie

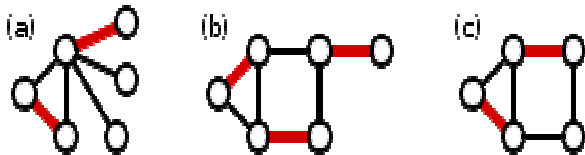
מוצג ע"י
אייל רונן

Overview

- What is Matching?
- Sample of real life uses for Matching
- Generic graph Matching Algorithm
- Bipartite graphs case
- Using the bipartite result for generic graph
- Summary

Matching

- Computing a set of disjoint links in a graph



- We achieve Maximal Matching when there are no disjoint links that can be added
- Maximum Cardinality Matching is the largest possible Maximal Matching

Sample of real life uses for Matching

- Matching Examples
 - Communication In Wireless Networks
- Bipartite Examples:
 - Optimizing data flow between ports in switches
 - Assigning end users to Cells in cellular networks / Wireless Infrastructure networks .

Some basic concepts

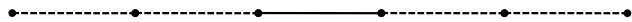
- Augmenting paths



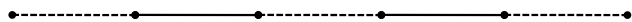
- Xoring Augmenting paths to Matching



XOR



EQUALS



Our Model For Generic Graph

- Input $G = (V, E)$
- Synchronous algorithm
- Maximum Message size $O(|V| + |E|)$
- Each node can send a message to each of her neighbors each turn
- Output M

Generic Graph Algorithm

- Start with a empty Matching M
- At phase ℓ we find a maximal set of augmenting paths of length $2\ell - 1$
- Xoring the set of augmenting paths to the current matching gives the matching of the next phase
- After ℓ phase we get a Matching of size at least $(1 - \frac{1}{\ell})|M^*|$

Why Does It Work?

- We can use Lemmas from Hopcroft and Karp article
- If the shortest augmenting path w.r.t M has length ℓ and P is a maximal set of augmenting paths w.r.t M . $M \text{ Xor } P$ has length greater than ℓ
- If the shortest augmenting path w.r.t M has length $2k - 1 \geq 1$ then $|M| \geq (1 - \frac{1}{k})|M^*|$

Finding the Maximal Set

- We Build the Maximal Set using ℓ -conflict graph
- Each node in the graph is an augmenting path in G w.r.t M with length up to ℓ
- Two nodes are connected iff their augmenting paths intersect in G
- We Run distributed MIS algorithm on the conflict graph

Building the ℓ -conflict graph

- At time $i = 1..2\ell$ v sends to all its neighbors a description of its neighborhood to distance $i-1$
- v checks if its an endpoint of an augmenting path of length ℓ and if so if its ID is lower then the other end point
- If so v is a leader of the path, and it knows all of the leaders intersecting her path

Analyzing the Algorithm

- The correctness is trivial with induction
- The messages used in building the conflict graph describe parts of the graph. Therefore they bounded by $O(|G|) = O(|V| + |E|)$

Analyzing the Algorithm cont.

- The algorithm runs for $2k-1$ iterations when $k = \epsilon^{-1}$
- The number of nodes in ℓ -conflict graph is $n^{O(\ell)}$
- The MIS alg. Complexity is $\log(N) = \log(n^{O(\ell)}) = O(\ell) \log n$
- Each routing step in the MIS alg. Takes $O(\ell)$ to route between leaders on G
- Total $O(\ell^3 \log(N)) = O(\epsilon^{-3} \log(N))$

Bi Partite Case

- $V = X \cup Y$ $E \subseteq X \times Y$
- For the Bi Partite Case we can avoid constructing all of the I-Conflict Graph
- We combine the construction process with the MIS calculation
- To find the Maximal Set each node needs only to know how many augmenting path pass through it.

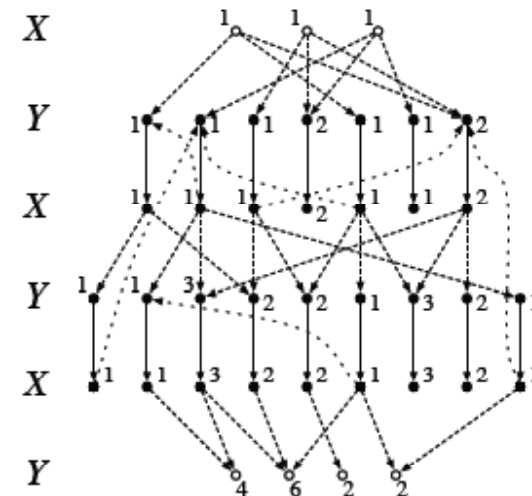
Counting augmenting path

- The graph is divide to X and Y
- On the first turn
 - All free nodes in X sends 1 on all edges and halt
- At the first turn a message is received
 - The node record the number received from each edge
 - The node sums all of the messages it received

Counting augmenting path cont.

- If the node is X
 - Send sum to all neighbors and halt
- If the node is Y
 - If matched send to matched X and halt
 - If not matched and time = /
 - record sum, path leader
 - Else halt

Counting Example



Computing Maximal Set

- We use a distributed parallel randomized algorithm written by M. Luby to find the MIS
- Each node chooses a number uniformly at random from $[1, N^4]$, $N = n\Delta^{(l+1)/2}$ number of nodes
- Node gets added to the set iff it's number is the highest of all its neighbors
- After $\log(N)$ iterations we get MIS w.h.p

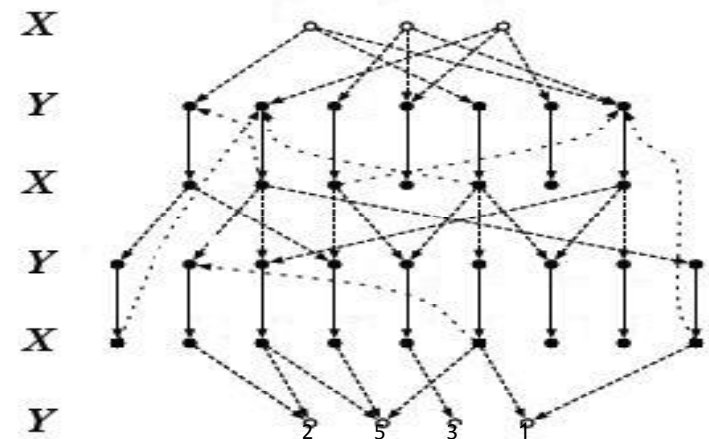
Implementing an Iteration

- Each leader node chooses a edge with a distribution relative to the number received on the edge.
- The leader chooses a number from a distribution relative to the number of paths it "leads"
- The node send the number token to chosen edge

Implementing an Iteration

- Each X node on the way sends up only the highest token received and record the source
- Each Y node on the way chooses an X node randomly relative to the number received on the edge
- At the end the path is rebuild and augmented

MIS Iteration Example



Reducing the Size of Messages

- The upper limit on number of augmenting path is $\Delta^{(l+1)/2}$ or $O(l \log(\Delta)) \leq O(l \log(n))$ bits
- We can pipeline the token in l parts MSB first, so each nodes send only the highest MSB it receives
- Each iteration of MIS takes $O(l)$ turns, and total for the MIS algorithm is:
 $O(l \log N) = o(l^2 \log(\Delta) + l \log(n))$

Back to general graphs

- We can achieve the message size of the bipartite case in a general graph
- Each iteration we randomly divide the graph into a bipartite G'
- We find maximal set of augmenting path on the bipartite graph and add it to the matching

Running time

- In each iteration we need to find a maximal set of augmenting paths with length at most $2k-1$
- The probability of a path in G appearing in G' is 2^{-2k+1}
- We can prove that w.h.p after $2^{2k+1}(k+1)\ln(k)$ iteration we get a $(1-\frac{1}{k})MCM$ approximation

Summary

- We have seen an efficient distributed algorithm to achieve $(1-\frac{1}{l})MCM$ approximation
- We have seen how to reduce the message size in the bipartite case using random algorithm
- We have seen a method to use randomization to achieve the same message size on general graphs