MST Construction in $O(\log \log n)$ Communication Rounds

Haneen Dabain

Department of Computer Science and Engineering
York University

February 11, 2010

Table of contents

Table of contents

The Problem

- The MST can be trivially constructed in a single round of communication, if messages are not restricted in size : each process sends all its information to all its neighbors, allowing each node to locally compute the MST.
- Note that the previous algorithm requires messages of size $O(n \log n)$.
- Message size could be limited.

Table of contents

The Model

- A complete weighted undirected graph $G = (V, E, \omega)$ where $\omega(e)$ denotes the weight of edge $e \in E$ and $|V| = n$.
- All edge weights are different (w.l.o.g.).
- Each node knows all its edges weights.
- Each node knows about all the other nodes.
- Each message contains at most $O(\log n)$ bits.
- Each edge has a weight of $O(\log n)$ bits.
- Each node has a distinct ID of $O(\log n)$ bits.
- Model is reliable: messages are never lost or corrupted
- The synchronous model is used.
  Note: The algorithm works in the asynchronous model using the simple synchronizer

Related Work

- An algorithm by Gallagher, Humbles, and Spira.
- Works in phases: for each phase, each cluster will add the Minimum Weight Outgoing Edge (MWOE) connecting that cluster to a node outside the cluster.
- Requires $\log n$ phases.
- Can this be achieved in a less number of phases?

Table of contents

The General Idea

The Main Goal

- In the previous algorithm , the minimum cluster size doubled in each phase. Let β_k denotes the minimum cluster size in phase k, then $\beta_{k+1} = 2\beta_k$
- Clusters have to grow faster by merging clusters, such that $\beta_{k+1} = \beta_k(\beta_k + 1)$
- Since $\beta_k \leq n$, it follows that $k = \log(\log n) + 1$ and the time complexity is $O(\log(\log n))$.
- To achieve such a rate, information has to be spread faster.

The General Idea

Message Size Limitation

- To overcome the limitation of message size:

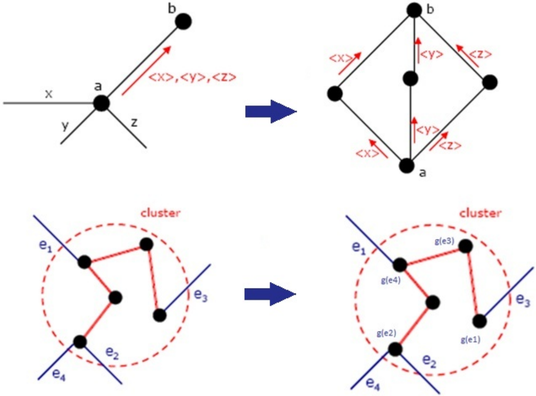


Table of contents

Notations

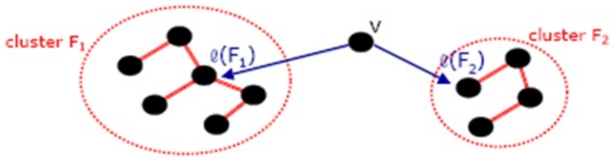
- V_0: A special node in the graph, e.g., the node with the smallest ID in the graph.
- $l(F)$: A leader of cluster F, e.g. the node with the smallest ID in the cluster.
- $g(e)$: A guardian node assigned to each minimum weight edge e.

Table of contents

Step by Step

Finding the β Lightest Edges

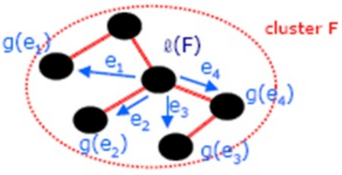
- Step 1:
  1. Each node computes the minimum-weight edge that connects it to any node in cluster F other than the own cluster.
  2. Each node sends the edge to the leader $l(F)$ of that cluster F.



Step by Step

Finding the β Lightest Edges

- Step 2:
  1. Each leader $l(F)$ of a cluster F computes the lightest edge between F and every other cluster.
  2. Each leader $l(F)$ selects the β lightest outgoing edges and appoint them to its nodes (guardians).



Step by Step

Procedure Cheap_Out

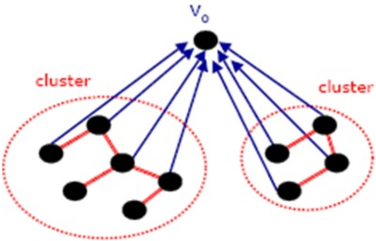
Input: Lightest edge $e(F, \grave{F})$ for every other cluster $\grave{F}$.

1. Sort the input edges in increasing order of weight
2. Define β = minimum size of all clusters
3. Choose the first β edges of the sorted list
4. Appoint the node with the i^{th} largest ID as the guardian of the i^{th} edge, $i = 1, ..., \beta$
5. Send a message about the edge to the node it is appointed to.

Step by Step

Sending Edges to the Special Node

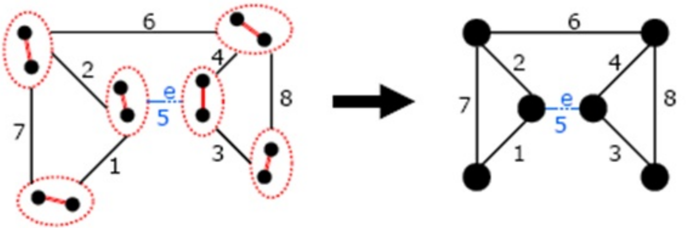
- Step 3:
  1. Each guardian node $g(e)$ sends the edge assigned to it to a specific node V_0.
- V_0 knows the β lightest outgoing edges of each cluster.
- V_0 needs to know which edges can be added without creating a cycle.



Step by Step

Example

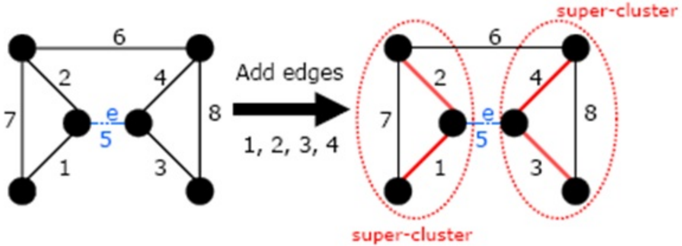
- Assume we have 12 nodes, and $\beta = 2$ (minimum cluster size)
- This is the picture V_0 has after receiving the $\beta = 2$ lightest outgoing edges of each cluster.
- V_0 can construct a logical graph.



Step by Step

Example

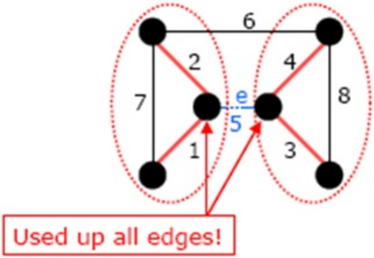
- V_0 can locally merge nodes of the logical graph into clusters.



Step by Step

Major Issue

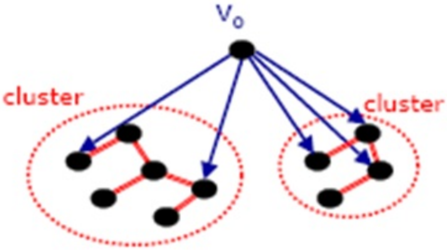
- Can we add the edge 6?
- When all β outgoing edges of a cluster are used up, It is not safe to add any other edge.
- Therefore 6 is rejected.
- The remaining edges will be rejected because they create cycles.



Step by Step

Making the Decision

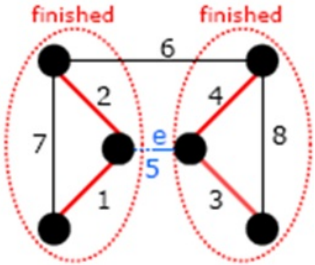
- Step 4:
    1. V_0 locally performs procedure Const_Frags to computes the edges to be added.
    2. V_0 sends messages to the guardians of all edges that have been added.



Step by Step

How Does Const_Frags Work?

- We call a super-cluster a finished cluster if it contains a cluster that used up all of its β edges.
- If an edge is the lightest outgoing edge of one super-cluster that is not finished, then it is still safe to add it.



Step by Step

Procedure Const_Frags

Input: the β lightest outgoing edges of each cluster

1. Construct the logical graph
2. Sort the input edges in increasing order of weight
3. Go through the list, starting with the lightest edge
   If the edge can be added without creating a cycle and is safe to add
   Then: add it
   Else: drop it

Step by Step

Adding Edges

- Step 5
  1. All nodes, that received a message from V_0, broadcast their edge to all other nodes.
- Step 6
  1. Each node adds all edges and computes the new clusters.
  2. If the number of resulting clusters is greater than one, then the next phase starts.

Step by Step

Review of the Algorithm for Node v in Cluster F

- Compute the minimum-weight edge $e(v, \grave{F})$ that connects v to cluster $\grave{F}$ and send it to $l(\grave{F})$ for all clusters $\grave{F} \neq F$.
- If $v = l(F)$, Compute the β lightest edge between F and every other cluster. Perform Cheap_Out
- If $v = g(e)$ for some edge e: Send e to V_0
- If $v = V_0$: Perform Const_Frags. Send message to $g(e)$ for each added edge e
- If v received a message from V_0 : Broadcast it
- Add all received edges and compute the new clusters

Table of contents

Extension to Large Messages

Procedure Cheap_Out:

- Input: Lightest edge $e(F, \grave{F})$ for every other cluster $\grave{F}$.
- Sort the input edges in increasing order of weight
- Define $\beta =$ minimum size of all clusters
- Choose the first $l \cdot \beta$ edges of the sorted list
- Appoint the node with the i^{th} largest ID as the guardian of the j^{th} edge if $j \ mod \ (l \cdot \beta) = i$.
- Send a message about the edge to the node it is appointed to.

Step 3:

- Each guardian node g(e) sends all the edge assigned to it to a specific node V_0.

Conclusion

- This algorithm solves the MST problem in the given model in $O(\log \log n)$ rounds.
- $O(\log \log n)$ is a good result , faster than $O(\log n)$.
- The algorithm sends $O(n^2 . \log n)$ bits, which is optimal.
- Questions:
  Is there a faster algorithm? Is $O(\log \log n)$ a lower bound?