DISTRIBUTED MINIMUM SPANNING TREES

CME 323 Project

Swaroop Indra Ramaswamy & Rohit Patki

May 27, 2015

Stanford University

Problem Statement & Assumptions

Problem Statement

Given an undirected, weighted simple graph we attempt to find a Minimum Spanning Tree (MST) or Minimum Spanning Forest (MSF).

Assumptions

- The number of edges in the graph (m) is much larger than the number of vertices of the graph (n)
- The edges of the graph do not fit in the memory of a single machine
- The vertices of the graph do fit in the memory of a single machine

Applications

- Single linkage clustering
- Network Design
- Image Segmentation
- Taxonomy
- Broadcasting in computer networks
- Important primitive in many graph algorithms

Single machine algorithms

Classical Algorithms

- Kruskal's Algorithm - $O(m \log n)$
- Prim's Algorithm - $O(m \log n)$

Faster Algorithms

- Karger, Klein and Tarjan (1995) - Randomized $O(m)$
- Bernard Chazelle (2000) - $O(m\alpha(m, n))$
- Fredman and Willard (1994) - $O(m + n)$ for integer weights

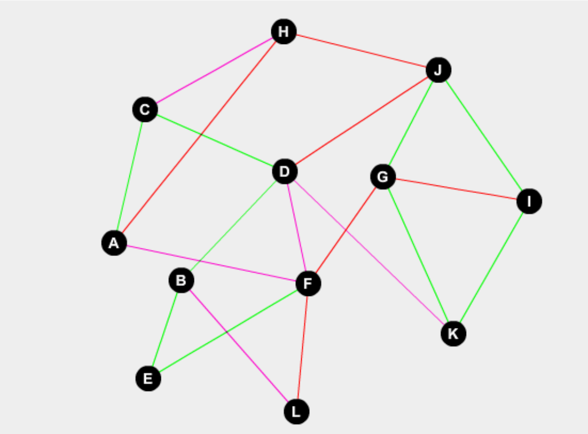
Edge partitioning

if $|E| < \eta$ **then**
  Compute $T^* = MST(V, E)$
  Return T^*
end if
$k = \Theta\left(\frac{|E|}{\eta}\right)$
Partition E into E_1, E_2, ... E_k where $|E_i| < \eta$ using a universal hash function
Compute $T_i^* = MST\left(G\left(V, E_i\right)\right)$ in parallel
Return $MST\left(G\left(V, \cup_i T_i^*\right)\right)$

Algorithm 1: Edge partitioning

Egde partitioning

Figure: Edge Partition



Edge partitioning : Analysis

- Memory of each machine, $\eta = O\left(n^{1+\epsilon}\right)$
- Number of edges, $m = O\left(n^{1+c}\right)$
- Number of machines, $k = O\left(n^{c-\epsilon}\right)$

Processing time

It can be shown that the algorithm takes $\lceil \frac{c}{\epsilon} \rceil$ iterations in expectation.

Processing time per iteration

$$\underbrace{O\left(\frac{m}{k}\log n\right)}_{Kruskal's\ on\ each\ machine} + \underbrace{O\left(\frac{m}{k}\right)}_{random\ partioning\ of\ edges}$$

Edge partitioning : Analysis

Communication Cost

- One all-to-all communication (shuffle) in each iteration
- Cost per iteration = m, $\frac{m}{n^{\epsilon}}$, $\frac{m}{n^{2\epsilon}}$

Total communication cost

$$\frac{n\left(n^{c}-1\right)}{1-n^{-\epsilon}}$$

Partition V into V_1, V_2, … V_k with $V_i \cap V_j = \Phi$ using a universal hash function
Denote the edges induced by V_i and V_j by $E_{i,j}$
Denote the induced subgraph by $G\left(V_i \cup V_j, E_{i,j}\right)$
Compute $T^*_{i,j} = MST\left(G\left(V_i \cup V_j, E_{i,j}\right)\right)$ in parallel
Return $MST\left(G\left(V, \cup_{i,j} T^*_{i,j}\right)\right)$

Algorithm 2: Vertex partitioning

Vertex partitioning : Analysis

- Number of edges, $m = n^{1+c}$
- Number of partitions, $k = n^{\frac{c}{2}}$
- Number of edges in each mapper = $n^{1+\frac{c}{2}}$, in expectation

Processing Time

Total processing time

$$\underbrace{O\left(\frac{m}{k}\log\frac{n}{k}\right)}_{Kruskal's\ on\ each\ machine} + \underbrace{O\left(n^{1+\frac{c}{2}}\log n\right)}_{Final\ Kruskal's}$$

Vertex partitioning : Analysis

Communication Cost

- One one-to-all communication (broadcast) = $O\left(nk^2\right)$
- One all-to-all (groupByKey) = $O\left(m\right)$
- One all-to-one to compute the final MST = $O\left(n^{1+\frac{c}{2}}\right)$

Total communication cost

$$O\left(nk^2\right) + O\left(m\right) + O\left(n^{1+\frac{c}{2}}\right)$$

Communication Time

Total communication time

$$O\left(n \log k\right) + O\left(m\right) + O\left(n^{1-\frac{c}{2}}\right)$$

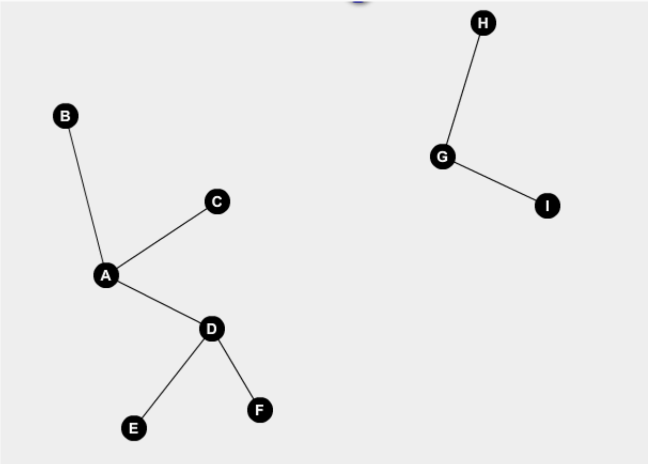
Parallel Prim's Algorithm

A = DISJOINTSET()
for i in V **do**
   A.MAKE-SET(i)
end for
Broadcast A
Find the minimum edge leaving each disjoint set using a reduce operation, denote this by the list of edges, $\hat{E}$
while $|\hat{E}| > 0$ **do**
   for e in $\hat{E}$ **do**
      A.UNION(u, v)
   end for
   Broadcast A
   $\hat{E}$ = Minimum edges leaving the disjoint sets
end while

Algorithm 3: Parallel Prim's Algorithm

Parallel Prim's

Figure: Connected components



Parallel Prim's

Figure: Potential new edges

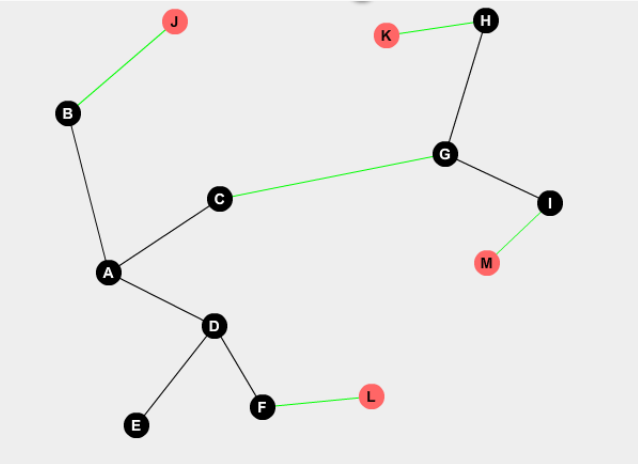


Figure: Final new edge after reduce

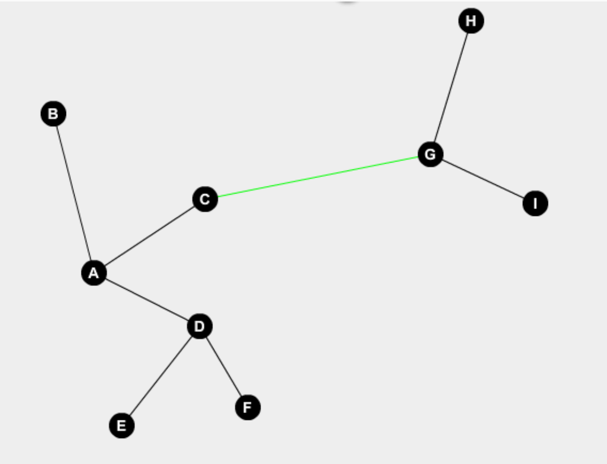
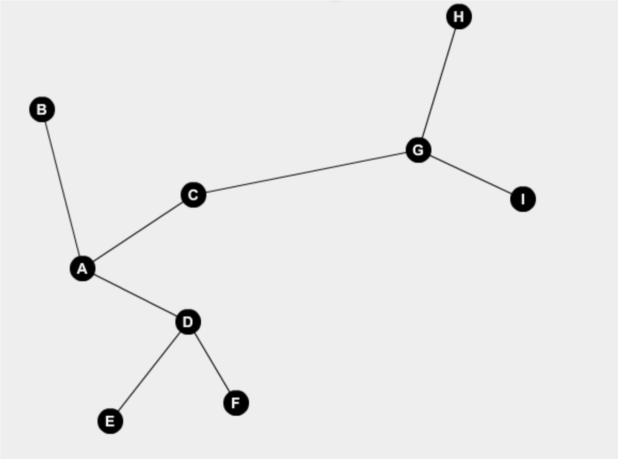


Figure: New connected component



Parallel Prim's Algorithm : Analysis

After each iteration of the while loop, the number of edges left to find reduces by at least $\frac{1}{2}$. Therefore, at most $\log n$ iterations are required.

Processing Time

Total Processing Time,

$$\underbrace{\log n}_{iterations} \times \underbrace{O\left(\frac{m}{k}\right)}_{per\ iteration} + \underbrace{O(m)}_{Total\ cost\ of\ all\ the\ reduces}$$

Parallel Prim's Algorithm : Analysis

Communication Cost

- One one-to-all broadcast of the disjoint set data-structure, per iteration = $O(nk)$
- One reduce to find minimum edges, per iteration = $O\left(\frac{n}{2^i}\right)$

Total communication cost,

$$O(nk \log n) + O(n)$$

Communication Time

Total communication time,

$$O(n \log k \log n) + O(n \log n)$$

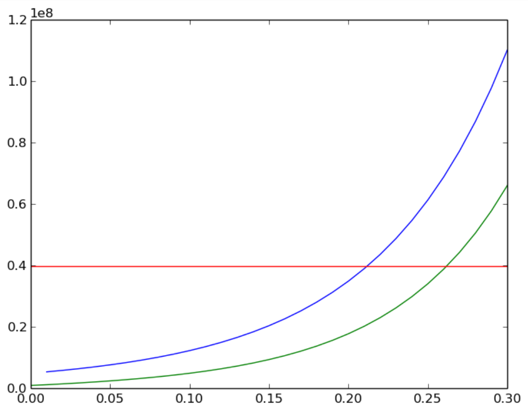
Theoretical Comparison

	Edge Partitioning	Vertex Partitioning	Parallel Prim's
Proc. Time	$O\left(\frac{m}{\epsilon k}\log n\right)$	$O\left(\left(\frac{m}{n^{\frac{c}{2}}}+n^{1+\frac{c}{2}}\right)\log n\right)$	$O\left(\frac{m}{k}\log n+m\right)$
Comm. Time	$O\left(m\frac{1-n^{-c}}{1-n^{-\epsilon}}\right)$	$O\left(m+cn\log n\right)$	$O\left(n\log k\log n\right)$

- Number of edges $m = n^{1+c}$
- Memory per machine $= n^{1+\epsilon}$

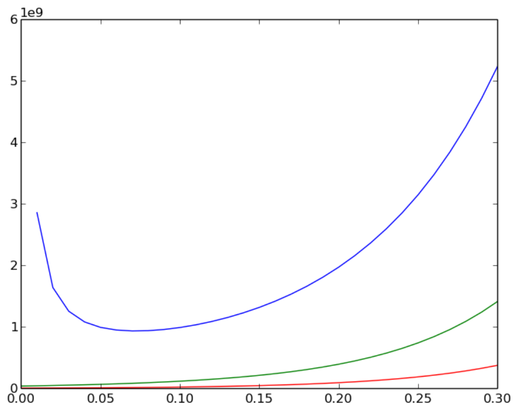
Plots: Communication Time

Figure: Communication Time vs vs c for $n = 1,000,000$



Plots: Processing Time

Figure: Processing Time vs c for $n = 1,000,000$



Experimental Comparison

	No.of Vertices	No.of Edges	Edge Partitioning	Vertex Partitioning	Parallel Prim's
1	281903	2312497	791 s	316 s	92 s
2	875713	5105039	7384 s	3733 s	229 s
3	685230	7600595	3670 s	1569 s	313 s
4	1696415	11095298	> 7200 s	> 3600 s	335 s
5	1088092	1541898	> 7200 s	> 3600 s	394 s

1. Stanford web graph
2. Google web graph
3. BerkStan graph
4. as-skitter
5. Road-net PA

Summary

- 3 algorithms for distributed MST : **Vertex Partitioning**, **Egde Partitioning** and **Parallel Prim's**
- Communication time for Parallel Prim's is independent of the number of edges
- For sparse graphs, if large number of machines are available, use **Vertex Partitioning**
- In general, **Parallel Prim's** has a better processing time than the other two algorithms
- As the density of the graph increases, **Parallel Prim's** wins out