Notes on MapReduce Algorithms

Barna Saha

1 Finding Minimum Spanning Tree of a Dense Graph in MapReduce

We are given a graph $G = (V, E)$ on $|V| = N$ vertices and $|E| = m \geq N^{1+c}$ edges for some constant $c > 0$. Our goal is to compute the minimum spanning tree of the graph.

Algorithm 1 MST computation in MapReduce

Set $k = N^{\frac{c}{2}}$.
Partition V into k parts of equal size: $V_1, V_2, ..., V_k$ with $V_i \cap V_j = \phi$ for $i \neq j$ and $|V_i| = \frac{N}{k}$ for all i.
▷ Let $E_{i,j} \subseteq E$ be the set of edges induced by the vertex set $V_i \cup V_j$, that is $E_{i,j} = \{(u, v) \in E \mid u, v \in V_i \cup V_j\}$.
Distribute $G_{i,j} = \{V_i \cup V_j, E_{i,j}\}$ to each server and compute its minimum spanning tree $M_{i,j}$.
Distribute $H = \cup M_{i,j}$ to a single server and compute the final MST M of H
return M

Theorem 1. *The tree M computed by Algorithm 1 is the minimum spanning tree of G.*

Proof. The algorithm works by sparsifying the input graph. We now show that H contains all the relevant edges that may appear in the MST of G.

Consider an edge $e = (u, v)$ that was discarded, that is $e \in E(G)$ but $e \notin E(H)$. Observe that any edge $e = (u, v)$ is present in at least one subgraph $G_{i,j}$. If $e \notin M_{i,j}$ then by the cycle property of minimum spanning trees, there must be some cycle $C \subseteq E_{i,j}$ such that e is the heaviest edge on the cycle. However, since $E_{i,j} \subseteq E$, we have now exhibited a cycle in G such that e is also the heaviest edge in that cycle in G. Therefore, e cannot be in the MST of G, and can be safely discarded. □

We now look at the memory usage at each machine.

Lemma 1. *Let $k = N^{\frac{c}{2}}$, then with high probability the size of every $E_{i,j}$ is $\tilde{O}(N^{1+\frac{c}{2}})$.*

Proof. We can bound the total number of edges in $E_{i,j}$ by bounding the total degree of the vertices in V_i and V_j. Thus,

$$|E_{i,j}| \leq \sum_{v \in V_i} deg(v) + \sum_{v \in V_j} deg(v).$$

For the purpose of this proof, let us partition the vertices into groups by their degree: let W_1 be the set of vertices of degree at most 2, W_2 be the set of vertices with degree 3 or 4, and in general, W_i be the set of vertices with degree in between $(2^{i-1}, 2^i]$. There are $\log N$ total groups.

Consider the number of vertices from group W_i that are mapped to V_j. If the group has a small number of vertices, that is $|W_i| < 24N^{\frac{c}{2}} \log N$, then

$$\sum_{v \in W_i} deg(v) \leq N * 24N^{\frac{c}{2}} \log N = \tilde{O}(N^{1+\frac{c}{2}})$$

If the group is large, define an indicator random variable, X_i^u which is 1 if $u \in W_i$ belongs to V_j. Then $|W_i \cap V_j| = \sum_{u \in W_i} X_i^u$.

$$\mathbb{E}[|W_i \cap V_j|] = \frac{|W_i|}{k} \geq 10 \log N$$

Then, by a simple application of Chernoff's bound,

$$Prob(|W_i \cap V_j| > \frac{3}{2}\frac{|W_i|}{k}) \leq exp(-\frac{24 \log N}{3.2^2}) = \frac{1}{n^2}$$

Therefore, by union bound

$$Prob(\exists i, |W_i \cap V_j| > \frac{3}{2}\frac{|W_i|}{k} \mid |W_i| \geq 24 N^{\frac{c}{2}} \log N) \leq \frac{\log n}{n^2}$$

Hence, with probability at least $1 - \frac{\log N}{N^2}$:

$$\begin{aligned}\sum_{v \in V_j} deg(v) &\leq \sum_i \sum_{v \in V_j \cap W_i} deg(v) \\ &\leq \frac{3}{2k} \sum_i \sum_{v \in W_i} deg(v) = O(N^{1+\frac{c}{2}})\end{aligned}$$

□

Lemma 1 tells us that with high probability each part has $\tilde{O}(N^{1+\frac{c}{2}})$ edges. Therefore, each reducer uses sub linear amount of memory space to compute $M_{i,j}$.

There are N^c total parts, again sublinear in the input size. Each part outputs $M_{i,j}$ which contain at most $N/k - 1$ edges. Hence size of H is at most $O(N^c \frac{N}{k}) = O(N^{1+\frac{c}{2}})$

2 An Alternate Algorithm for MST in MapReduce

Recall that we use n to denote the total input size that is $n = N + M$. Fix some $\epsilon > 0$.

Algorithm 2 MST computation in MapReduce

if $|E| \leq N^{1+\epsilon}$ **then**
  Compute $T^* = MST(E)$
return T^*
end if
$l \leftarrow \Theta(|E|/N^{1+\epsilon})$
Partition E into E_1, E_2,...,E_l where $|E_i| \leq N^{1+\epsilon}$ using a universal hash function $h : E \rightarrow \{1, 2, ..., l\}$.
In parallel: Compute T_i, the minimum spanning tree on $G = (V, E_i)$

Recursively return the minimum spanning tree on $G = (V, E = \cup_i T_i)$

Theorem 2. *Algorithm 2 terminates after $\lceil \frac{c}{\epsilon} \rceil$ iterations, and returns the minimum spanning tree of the original graph G correctly.*

Proof. To show correctness, note that any edge that is not part of the MST on a subgraph of G is also not part of the MST of G by the cycle property of the minimum spanning trees.

It remains to show that the memory constraints are not violated, and the total number of rounds is bounded.

First, since the partition is done randomly, applying the Chernoff bound, no machine gets more than $2N^{1+\epsilon}$ edges with high probability.

Also, note that after first round,

$$|\cup_i T_i| \leq l(N-1) = \frac{N^{1+c}}{N^{1+\epsilon}}(N-1) = N^{1+c-\epsilon}$$

After the second round,

$$|\cup_i T_i| \leq l(N-1) = \frac{N^{1+c-\epsilon}}{N^{1+\epsilon}}(N-1) = N^{1+c-2\epsilon}$$

Continuing, after $\lceil \frac{c}{\epsilon} \rceil -$ iterations, the input is small enough to fit in the memory of a single machine, and the overall the algorithm terminates in $\lceil \frac{c}{\epsilon} \rceil$ rounds. □

3 Densest Subgraph Computation

Definition. Let $G = (V, E)$ be an undirected graph. Given $S \subseteq V$, its density $\rho(S)$ is defined as

$$\rho(S) = \frac{|E(S)|}{|S|}$$

The maximum density $\rho^*(G)$ of the graph is then

$$\rho^*(G) = \max_{S \subseteq V} \{\rho(S)\}.$$

The above definition can be naturally extended to weighted graphs.

A simple greedy algorithm First, let us look into a very simple greedy algorithm. The algorithm repeatedly removes the minimum degree vertex among the remaining vertices, and compute the resultant density. At the end, it returns the density of the subgraph that is maximum among all the iterations. The algorithm is naturally sequential, as it removes one vertex at a time based on minimum degree.

This simple algorithm gives a $\frac{1}{2}$-approximation to the densest subgraph as follows. Suppose the optimum subgraph is S_{OPT} with density ρ^*. Then each vertex in S_{OPT} must have degree at least ρ^*, otherwise, a simple calculation shows that removing that vertex improves the density of the subgraph contradicting its optimality.

Consider the first iteration, at which our greedy algorithm tries to remove a vertex from S_{OPT}. At that time, suppose the vertices still under consideration is $T \supseteq S_{OPT}$. Every vertex in T must have degree at least ρ^*. Thus the total number of edges in T is at least $\rho^* T/2$, or its density is at least $\rho^*/2$. Since we return the subgraph with maximum density, the returned subgraph will have density at least $\frac{\rho^*}{2}$. □

We now give a simple streaming algorithm, which can also be implemented in the map reduce framework.

A simple streaming algorithm Starting with the given graph G, the algorithm computes the current density, $\rho(G)$, and removes all of the nodes (and their incident edges) whose degree is less than $(2 + 2\epsilon)\rho(G)$ for some fixed $\epsilon > 0$. If the resulting graph is non-empty, then the algorithm recurses on the remaining graph, with node set denoted by S, again computing its density and removing all of the nodes whose degree is lower than the specified threshold; we denote these nodes by $A(S)$. Then, the node set reduces to $S \setminus A(S)$, and the recursion continues in the same way.

Lemma 2. *The algorithm above obtains a $\frac{1}{2+2\epsilon}$-approximation to the densest subgraph problem.*

Proof. Fix some optimal solution S^*. For each $i \in S$, $deg_{S^*}(i) \geq \rho(S^*)$.

Since $\sum_{i \in S} deg_S(i) = 2|S|\rho(S)$, at least one node must be removed in every pass. Now, consider the first time in the pass when a node i from the optimal solution S^* is removed. That is $A(S) \cap S^* \neq \phi$. This moment is guaranteed to exist, since S eventually becomes empty. Clearly, $S \supseteq S^*$. Let $i \in A(S) \cap S^*$. We have

$$\begin{aligned} \rho(S^*) &\leq deg_{S^*}(i) \\ &\leq \deg_S(i) \leq (2+2\epsilon)\rho(S) \end{aligned}$$

This implies $\rho(S) \geq \frac{\rho(S^*)}{(2+2\epsilon)}$. □

Lemma 3. *The algorithm above terminates in $O(\log(1+\epsilon)n)$ passes.*

Proof. At each step of the pass, we have

$$\begin{aligned} 2|E(S)| &= \sum_{i \in A(S)} deg_S(i) + \sum_{i \in S \setminus A(S)} deg_S(i) \\ &> 2(1+\epsilon)(|S| - |A(S)|)\rho(S) = 2(1+\epsilon)(|S| - |A(S)|)\frac{|E(S)|}{|S|} \end{aligned}$$

Thus,

$$|A(S)| > \frac{\epsilon}{1+\epsilon}|S|.$$

Or,

$$|S \setminus A(S)| < \frac{1}{1+\epsilon}|S|$$

Therefore, the cardinality of the remaining set S decreases by a factor of at least $1/(1+\epsilon)$ during each pass. Hence, the algorithm terminates in $O(\log_{(1+\epsilon)} n)$ passes. □

4 Estimating Density via Sampling

Let G' be the graph formed by sampling each edge in G independently with probability p where

$$p = \frac{c}{\epsilon^2} \log N \frac{N}{M}$$

where c is a large enough constant, $\frac{1}{2} > \epsilon > 0$ is the desired accuracy that we want, $N = |V(G)|$ and $M = |E(G)|$. We may assume that M is sufficiently large such that $p < 1$, otherwise the entire graph can be kept in a single machine.

Let U be an arbitrary set of k nodes. We use ρ_G and $\rho_{G'}$ to denote the density of a set of nodes in G and G' respectively. We first relate the density of U in G' with that in G.

Lemma 4. *For all $U \subseteq V(G)$,*

$$\begin{aligned} &Pr(\rho_{G'(U)} \geq pOPT/10) \leq N^{-8} && \text{if } \rho_G(U) \leq OPT/60 \\ &Pr(|\rho_{G'(U)} - p\rho_G(U)| \geq \epsilon p \rho_G(U)) \leq N^{-8} && \text{if } \rho_G(U) \geq OPT/60 \end{aligned}$$

Proof. Let X denote the number of induced edges in U in G'. Then

$$\mathbb{E}X = \rho_G(U)|U|p$$

Let OPT denote the optimum density, then $OPT \geq \frac{M}{N}$. Hence $p \geq \frac{c}{\epsilon^2} \log N/OPT$.

First, let us consider the case when U does not have density close to OPT. Let $\rho_G(U) \leq OPT/60$. Then,

$$\begin{aligned}
&Pr(\rho_{G'}(U) \geq pOPT/10) \\
&= Pr(X \geq p|U|OPT/10) \\
&= Pr(X \geq p|U|\rho_G(U)(1 + \frac{OPT/12}{\rho_G(U)})) \\
&\leq exp(-p|U|\rho_G(U)\frac{OPT/12}{\rho_G(U)}) \\
&= exp(-\frac{c|U|\log N}{12\epsilon^2}) \leq \frac{1}{N^{10|U|}}
\end{aligned}$$

On the other hand, if $\rho_G(U) \geq OPT/60$,then

$$\begin{aligned}
&Pr(|\rho_{G'}(U) - p\rho_G(U)| \geq \epsilon p\rho_G(U)) \\
&= Pr(|X - p|U|\rho_G(U)| \geq \epsilon p|U|\rho_G(U)) \\
&\leq 2exp(-\epsilon^2 p|U|\rho_G(U)|/3) \\
&\leq 2exp(-c|U|\log N/180) \leq \frac{1}{N^{10|U|}}
\end{aligned}$$

Here we have used c to be a sufficiently large constant.

Now there are at most $n^{|U|}$ subgraphs of size $|U|$. Hence with probability at least $1 - \frac{1}{N^{9|U|}}$ the above concentration result holds for all subgraphs of size $|U|$.

There are N possible values of $|U|$. Thus, overall, again by apply a union bound, the above concentration result holds for all subgraphs of G with probability at least $1 - \frac{1}{N^8}$ □

Theorem 3. *Let $U' = argmax_U\{\rho_{G'}(U)\}$. Then with high probability,*

$$(1-\epsilon)^2 OPT \leq \rho_G(U') \leq OPT$$

Proof. Consider the subgraph S^* with optimum density OPT in G. By the above lemma, in G', its density is at least $(1-\epsilon)pOPT$. Thus $\rho_{G'}(U) \geq (1-\epsilon)pOPT$.

Assuming $(1-\epsilon) \geq 1/10$, it must hold that $\rho_G(U) \geq OPT/60$, but then

$$\rho_G(U) \geq (1-\epsilon)\frac{1}{p}\rho_{G'}(U) \geq (1-\epsilon)^2 OPT.$$

□