MapReduce Algorithms

Barna Saha

March 28, 2016

Complexity Model for MapReduce

Minimum Spanning Tree in MapReduce

Computing Dense Subgraph in MapReduce

Complexity Model for MapReduce: $\mathcal{MRC}^i$

- Input: finite sequence of pairs: $(k_i, v_i) : (key, value)$.

Complexity Model for MapReduce: $\mathcal{MRC}^i$

- Input: finite sequence of pairs: $(k_i, v_i) : (key, value)$.
- Total Input length $= \sum_i k_i + v_i = n$

Complexity Model for MapReduce:$\mathcal{MRC}^i$

- Input: finite sequence of pairs: $(k_i, v_i) : (key, value)$.
- Total Input length$= \sum_i k_i + v_i = n$
- The algorithm executes a sequence of map and reduce tasks $(\mu_1, \rho_1, \mu_2, \rho_2, ..., \mu_R, \rho_R)$

Complexity Model for MapReduce: $\mathcal{MRC}^i$

Consider an $\epsilon > 0$.

- Each map and reduce task requires $n^{1-\epsilon}$ space

Complexity Model for MapReduce: $\mathcal{MRC}^i$

Consider an $\epsilon > 0$.

- Each map and reduce task requires $n^{1-\epsilon}$ space
- Thus the space available in each machine is sublinear in input size.

Complexity Model for MapReduce:$\mathcal{MRC}^i$

Consider an $\epsilon > 0$.

- Each map and reduce task requires $n^{1-\epsilon}$ space
- Thus the space available in each machine is sublinear in input size.
- Total number of machines used is sublinear as well, $n^{1-\epsilon}$

Complexity Model for MapReduce:$\mathcal{MRC}^i$

Consider an $\epsilon > 0$.

- Each map and reduce task requires $n^{1-\epsilon}$ space
- Thus the space available in each machine is sublinear in input size.
- Total number of machines used is sublinear as well, $n^{1-\epsilon}$
- The number of rounds $R = O((\log n)^i)$

Complexity Model for MapReduce

- Therefore $\mathcal{MRC}^0$ is the class of problems that requires only constant number of rounds with sublinear amount of memory in each machine, and sublinear number of machines altogether.

Complexity Model for MapReduce

- Therefore $\mathcal{MRC}^0$ is the class of problems that requires only constant number of rounds with sublinear amount of memory in each machine, and sublinear number of machines altogether.
- There are other classes defined such as $\mathcal{MR}$ model, where more explicit time+communication complexity of a problem is accounted for.

Minimum Spanning Tree (MST) in MapReduce

- Given a graph $G = (V, E)$ on $|V| = N$ vertices and $|E| = M \geq N^{1+c}$ edges for some constant $c > 0$ (n still denotes the length of the input and not the number of vertices)
- Compute Minimum Spanning Tree of the graph.

Minimum Spanning Tree (MST) in MapReduce

- Fix a number k

Minimum Spanning Tree (MST) in MapReduce

- Fix a number k
- Randomly partition the set of vertices into k equally sized subsets, $V = V_1 \cup V_2 \cup ... \cup V_k$, with $V_i \cap V_j = \phi$ for $i \neq j$ and $|V_i| = N/k$ for all i.

Minimum Spanning Tree (MST) in MapReduce

- Fix a number k
- Randomly partition the set of vertices into k equally sized subsets, $V = V_1 \cup V_2 \cup \ldots \cup V_k$, with $V_i \cap V_j = \phi$ for $i \neq j$ and $|V_i| = N/k$ for all i.
- For every pair $\{i, j\}$, let $E_{i,j} \subseteq E$ be the set of edges induced by the vertex set $V_i \cup V_j$.

$$E_{i,j} = \{(u, v) \in E \mid u, v \in V_i \cup V_j\}$$

Minimum Spanning Tree (MST) in MapReduce

- Fix a number k
- Randomly partition the set of vertices into k equally sized subsets, $V = V_1 \cup V_2 \cup \ldots \cup V_k$, with $V_i \cap V_j = \phi$ for $i \neq j$ and $|V_i| = N/k$ for all i.
- For every pair $\{i, j\}$, let $E_{i,j} \subseteq E$ be the set of edges induced by the vertex set $V_i \cup V_j$.

$$E_{i,j} = \{(u, v) \in E \mid u, v \in V_i \cup V_j\}$$

- Denote the resulting subgraph by $G_{i,j} = (V_i \cup V_j, E_{i,j})$.

Minimum Spanning Tree (MST) in MapReduce

- Place each $G_{i,j}$ in a single machine

Minimum Spanning Tree (MST) in MapReduce

- Place each $G_{i,j}$ in a single machine
- Compute MST $M_{i,j}$ of $G_{i,j}$

Minimum Spanning Tree (MST) in MapReduce

- Place each $G_{i,j}$ in a single machine
- Compute MST $M_{i,j}$ of $G_{i,j}$
- Let $H = (V, \bigcup_{i,j} M_{i,j})$

Minimum Spanning Tree (MST) in MapReduce

- Place each $G_{i,j}$ in a single machine
- Compute MST $M_{i,j}$ of $G_{i,j}$
- Let $H = (V, \bigcup_{i,j} M_{i,j})$
- Compute MST of H in a single machine

Minimum Spanning Tree (MST) in MapReduce

Theorem

The algorithm computes MST correctly.

- If an edge e is discarded, that is $e \in E(G)$ but $e \notin E(H)$: show that e is not part of a MST.

Minimum Spanning Tree (MST) in MapReduce

Theorem

The algorithm computes MST correctly.

- If an edge e is discarded, that is $e \in E(G)$ but $e \notin E(H)$: show that e is not part of a MST.
- Every edge is present in at least one $G_{i,j}$
- If an edge does not appear in $M_{i,j}$, then there exists a cycle in $G_{i,j}$ such that e is the heaviest weight edge in that cycle. This implies e cannot be part of the MST of G.

Minimum Spanning Tree (MST) in MapReduce

Lemma

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

- With high probability each part has $\tilde{O}(N^{1+c/2})$ edges. Therefore, the total input size to any reducer is $O(n^{1-\epsilon})$.

Minimum Spanning Tree (MST) in MapReduce

- There are N^c total parts, each producing a spanning tree with $2N/k - 1 = O(N^{1-c/2})$ edges.
- Thus the size of H is bounded by $\tilde{O}(N^{1+c/2}) = O(n^{1-\epsilon})$, again small enough to fit into the memory of a single machine.

Minimum Spanning Tree (MST) in MapReduce

Lemma

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

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

Minimum Spanning Tree (MST) in MapReduce

Lemma

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

- $|E_{i,j}| \leq \sum_{v \in V_i} deg(v) + \sum_{v \in V_j} deg(v)$.
- Let $W_i = \{v \in V : 2^{i-1} < deg(v) \leq 2^i\}$. Hence W_1 is the set of vertices with degree at most 2, W_2 is the set of vertices with degrees 3 and 4, and so on.

Minimum Spanning Tree (MST) in MapReduce

Lemma

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

- $|E_{i,j}| \leq \sum_{v \in V_i} deg(v) + \sum_{v \in V_j} deg(v)$.
- Let $W_i = \{v \in V : 2^{i-1} < deg(v) \leq 2^i\}$. Hence W_1 is the set of vertices with degree at most 2, W_2 is the set of vertices with degrees 3 and 4, and so on.
- There are $\log N$ total groups.

Minimum Spanning Tree (MST) in MapReduce

Lemma

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

- How many vertices from W_i are mapped to V_j?

Minimum Spanning Tree (MST) in MapReduce

Lemma

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

- How many vertices from W_i are mapped to V_j?
- If $|W_i| < 2N^{c/2} \log N$ then $\sum_{v \in W_i} deg(v) \leq 2N^{1+c/2} \log N = \tilde{O}(N^{1+c/2})$.

Minimum Spanning Tree (MST) in MapReduce

Lemma

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

- How many vertices from W_i are mapped to V_j?
- If $|W_i| < 2N^{c/2} \log N$ then $\sum_{v \in W_i} deg(v) \leq 2N^{1+c/2} \log N = \tilde{O}(N^{1+c/2})$.
- If the group is large, using concentration inequality, we can show the number of vertices mapped from any particular group to V_j is small.
- Overall, the total degree in any part remains bounded by $\tilde{O}(N^{1+c/2})$

Computing Dense Subgraph in MapReduce

Given an undirected graph $G = (V, E)$, compute a subset of nodes $S \subseteq V$ such that $\frac{|E(S)|}{|V(S)|}$ is maximized.

- Community Mining
- Computational Biology
- Link Spam Detection
- Efficient Indexing for Reachability Queries

Computing Dense Subgraph in Streaming Setting

- We will show an algorithm in that computes the dense subgraph in multiple passes, but in a single machine, and use $O(n \log n)$ memory at any pass.

Computing Dense Subgraph in Streaming Setting

- We will show an algorithm in that computes the dense subgraph in multiple passes, but in a single machine, and use $O(n \log n)$ memory at any pass.
- Exercise convert the algorithm into MapReduce framework.

Computing Dense Subgraph in Streaming Setting

- Let $\epsilon > 0$ be a parameter.
- We start with the given graph G, compute the current density $\rho(G)$ and remove all nodes whose degree is less than $(2 + 2\epsilon)\rho(G)$.
- If the remaining graph is nonempty, recurse on the remaining graph.
- Return the graph from the round which has highest density.

Computing Dense Subgraph in Streaming Setting

Lemma

Algorithm obtain s $(2 + 2\epsilon)$-approximation to the densest subgraph problem.

- Consider the round in which a vertex from the optimum subgraph S^* is removed for the first time.
- Consider a $i \in S^*$ that is removed.
- We have

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

Computing Dense Subgraph in Streaming Setting

Lemma

Algorithm terminates in $\log_{1+\epsilon}(n)$ *rounds,* $n = |V|$.

- Exercise: show that after each round, the number of vertices reduce by a factor of $\frac{1}{(1+\epsilon)}$.
- Exercise: show how to convert the algorithm into a $\mathcal{MRC}^1$ algorithm.