MCMD L18 : MapReduce | filtering for MST

MapReduce

D = Massive Data

Mapper(D): d in D -> {(key,value)}

Shuffle({(key,value)}) -> group by "key"

Reducer ({"key,value_i}) -> ("key, f(value_i))

Can repeat, constant # of rounds

______

MRC Model:
N = size of data
$O(N^{1-eps})$ memory on single machine (eps>0 constant)
so can’t fit all on one machine
at most N^{1-eps} machines total
Shuffle = $O(N^{2-2eps})$
so can’t shuffle more data than memory
Constant # rounds

"Filtering" idea:
consider subproblems -> drop many data points
recur until fits in memory, solve in-core

[Lattanzi, Moseley, Suri, Vassilvitskii 2011]

Given graph G=(V,E)

Assume $|V|=n$ and $|E| = m = n^{1+c}$
typical large graphs have c in [0.08, 0.5]
size of input is $N = O(n^{1+c})$

Find MST: (minimum spanning tree)

<MSF = minimum spanning forest, may not be connected>

each machine has memory $M = 2 * n^{1+eps} = O(N^{1-gamma})$
for $0 < eps < c$ and gamma > 0
(otherwise $|G| <= M$)

$P = Theta(n^{c-eps})$ so data just fits on machines

Map:
Partition E -> {E1,E2,...Ek}
 so E_i = Theta(M)
 k = 2*(|E|/M)
 (each edge e a random number i in [k]) -> (i,e)

Reduce:
 compute MSF(V,Ei) -> (V,Ei')
 E' = Union_i Ei'

If |E'| < M, solve on 1 machine
else : repeat M+R

Proof:
  3 parts (A) gives correct MST
          (B) finishes in constant number of rounds
          (C) no node has more than 2 * n^{1+eps} whp.

(A) Correctness:
Each edge thrown out was part of cycle, and was longer than all other edges.
  -> not in MST
  -> no edges in full MST thrown out.

(B): Constant number of rounds:
Each round decreases the size by a factor about n^{eps}.
  m_1 = |E'| <= k(n-1) = O(n^{1+c-eps})
  m_r = m_{r-1} / n^eps
-> requires c/eps iterations

Another view: If n^{1+c} = N, and n^{1+eps} = M,
 then requires R = log_M N rounds.

R = log_M N seems to be the goal in the number of rounds needed for hard problems...

(C) no Memory overflow:
Lemma. No machine has |Ei| > M = 2 * n^{1+eps} wp > 1/2
  (follows from Markov bound)

++++++++++++++++++++++++++++
Chernoff Inequality

```
Let {X_1, X_2, ..., X_r} be independent RVs
Let Delta_i =  max(X_i) - min(X_i)
Let S = sum_i X_i

Pr[ | S - sum_i E[X_i] | > alpha ] < 2 exp(- 2 alpha^2 / sum_i
(Delta_i)^2)

often:  Delta = max_i Delta_i then:
Pr[ |S - sum_i E[X_i]| > alpha ] < 2 exp(- 2 alpha^2/ r Delta^2)
++++++++++++++++++++++++++++

Let X_i represent edge i is in node j
Delta_i = 1-0 = 1; Delta = 1
S = number of edges on node j
sum_i E[X_i] = n^{1+eps}
Let alpha = n^{1+eps}
Pr[ S > 2 * n^{1+eps}] <=
 Pr[ |S - n^{1+eps}| > n^{1+eps}] <
 2 exp( -2 (n^{1+eps})^2 / n^{1+c} (1)^2)
 <= 2 exp(-2 n^{1+2eps-c})    let beta = 1+eps-c be a constant, beta >
0

with high probability (whp)  (probability <= e^{-poly(n)}):
   any node j has fewer than 2 * n^{1+eps} edges

to show for all k = n^{1+eps} nodes, we need to use union bound:
  no node has probability greater than e^{-n^{beta+eps}}/k
  easy to show that n^{beta+eps}/log(n^{1+eps}) > n^beta
  all nodes j has fewer than 2 * n^{1+eps} edge whp

------------------------------------------
Also solves # connected components
  Or assign component id to each vertex

Also w/ "filtering"
 - maximal matchings
 - approximate maximal weighted matchings
 - minimum cut
```