Distributed Laplacian Solver

Rahul V

November 25, 2019

Problem Statement: Solve $Lx = b$

- $x^T L = b^T$
- Laplacian matrix: $L = D - A$
- Adjacency matrix: $A_{uv} = w_{uv}$
- Diagonal matrix: $D_{uu} = \sum_{v=1}^{v=n} A_{uv}$
- One sink: $b_n = -\sum_{i=1}^{n-1} b_i$

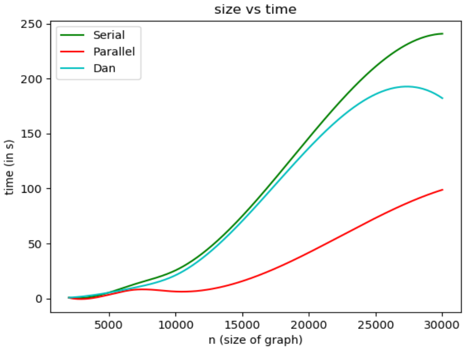
Algorithm

- Solve Data Collection Problem (DCP)
  - Each node generates a packet with probability βJ_u
  - Transmit a packet to a random neighbor v with probability w_{uv}
  - Packets sunk at the sink immediately
- Let η be the queue occupancy probability at stationarity
- $\eta^T(I - P) = \beta J^T$ or $(\eta^T D^{-1})L = \beta J^T$

Results

Dense graphs

- $N\_PROC = 4$
- Randomly generated graphs
- Spielman's solver employs Cholesky factorization



(a) Performance

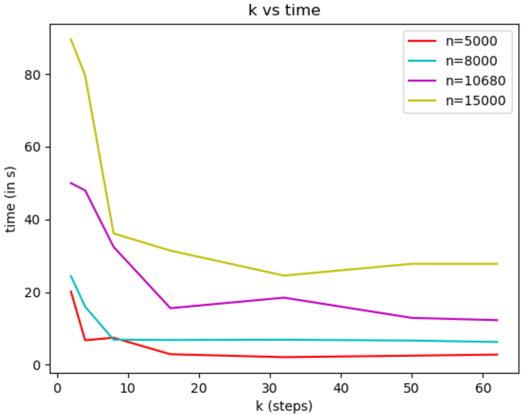
Results

Sparse graphs

- Terribly slow, in contrast to the Spielman's blazing fast solver (27s vs 0.34s)
- Reason:
  - Most solvers exploit sparsity; running time is linear in $\#edges$
  - ALG relies on fraction of packets sunk (C)
  - Random path to sink is longer in sparse graphs
- Solution:
  - Move the packets k steps at a time (Densify)
  - Each packet can move to any node within k hops

Result of k-step speed-up

Sparse graphs



(b) Speed up

Approximation of an approximate solution

- No speedup on dense graphs
- Satisfies a slightly different relation at stationarity
  - $\eta^T(I - P^k) = k\beta J^T$
- Retreiving original solution, α, without speed up:
  - $\eta^T(I - P^k) = k\alpha^T(I - P)$
  - $\alpha^T = \eta^T(I + P + \cdots + P^{k-1})/k$
- ALG performs just as good for sparse graphs now

Other algorithms of interest

- Becchetti's algorithm:
  - Send out *all* packets in each round
  - The $|Q_u|$ (not η_u) itself would be the solution.
  - $Q(t+1) = Q(t)P + \beta J$
  - $\#rounds$ to converge is lesser but time taken is more
  - Overall, performs worse than ALG
  - Unfortunately, the k step speed up doesn't help in this case.

Thank you