Graph Algorithms for Modern Data Models

Ashish Goel

Stanford University

Joint work with Kamesh Munagala; Bahman Bahmani and Abdur Chowdhury

Modern Data Models

- Over the past decade, many commodity distributed computing platforms have emerged
- Two examples: Map-Reduce; Distributed Stream Processing
- Similar to PRAM models, but have several nuances
  - Carefully calibrated to take latencies of disks vs network vs memory into account
  - Cost of processing is often negligible compared to the cost of data transfer
  - Take advantage of aggregation in disk and network operations
  - Example: the cost of sending 100KB is about the same as sending 1 Byte over a network

Data Model #1: Map Reduce

- An immensely successful idea which transformed offline analytics and bulk-data processing. Hadoop (initially from Yahoo!) is the most popular implementation.

- **MAP:** Transforms a (key, value) pair into other (key, value) pairs using a UDF (User Defined Function) called Map. Many mappers can run in parallel on vast amounts of data in a distributed file system

- **SHUFFLE:** The infrastructure then transfers data from the mapper nodes to the "reducer" nodes so that all the (key, value) pairs with the same key go to the same reducer and get grouped into a single large (key, $\langle val_1, val_2, ..\rangle$) pair

- **REDUCE:** A UDF that processes this grouped (key, $\langle val_1, val_2, ..\rangle$) pair for a single key. Many reducers can run in parallel.

Complexity Measures

- Key-Complexity:
  - The maximum size of a key-value pair
  - The amount of time taken to process each key
  - The memory required to process each key
- Sequential Complexity:
  - The total time needed by all the mappers and reducers together
  - The total output produced by all the mappers and reducers together
- Number of MapReduce phases

[Goel, Munagala; 2012]

Comp

THE CURSE OF THE LAST REDUCER

- Key-Complexity:
  - The maximum size of a key-value pair
  - The amount of time taken to process each key
  - The memory required to process each key
- Sequential Complexity:
  - The total time needed by all the mappers and reducers together
  - The total output produced by all the mappers and reducers together

[Goel, Munagala; 2012]

Complexity Measures

- Key-Complexity:
  - The maximum size of a key-value pair
  - The amount of time taken to process each key
  - The memory required to process each key
- Sequential Complexity:
  - The total time needed by all the mapp
    together
  - The total output produced by all the mappers and reducers together

SHUFFLE SIZE

[Goel, Munagala; 2012]

Complexity Measures

- Key-Complexity:
  - The maximum
  - The amount of tim key
  - The memory required rocess each key
- Sequential Complexity:
  - The total time needed by all the mappers and reducers together
  - The total output produced by all the mappers and reducers together

THE AMOUNT OF WORK DONE TO AGGREGATE ALL THE VALUES FOR A SINGLE KEY (SORTING) IS NOT A COMPLEXITY MEASURE

[Goel, Munagala; 2012]

Complexity Measures

- Key-Complexity:
  - The maximum size of a key-value pair
  - The amount of time taken to process each key
  - The memory required to process each key
- Sequential Complexity:
  - The total time needed by all the mappers and reducers together
  - The total output produced by all the mappers and reducers together
- Number of MapReduce phases

[Goel, Munagala; 2012]

Densest Subgraph (DSG)

- Given: an undirected graph $G = (V,E)$, with N nodes, M edges, and maximum degree d_{MAX}
  - For a subset S of nodes, let $E(S)$ denote the set of edges between nodes in S
  - Goal: Find the set S that maximizes $|E(S)|/|S|$
  - Applications: Community detection
- Can be solved in polynomial time
- A $(2+\varepsilon)$-approximation known on MapReduce
  - $O((\log N)/\varepsilon)$-phases
  - Each phase has sequential complexity $O(M)$ and key complexity $O(d_{MAX})$

  [Bahmani, Kumar, Vassilvitskii; 2012]

0	1	0	1	1	0	0	0	0	0	1	0
1	0	0	1	1	0	0	0	0	1	1	0
0	0	0	0	0	1	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	1	1	0
0	1	0	1	0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	1	1
1	1	0	0	1	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0

0	**1**	0	**1**	**1**	0	0	0	0	**0**	**1**	0
1	**0**	0	**1**	**1**	0	0	0	0	**1**	**1**	0
0	0	0	0	0	1	0	0	0	0	0	0
1	**1**	0	**0**	**1**	0	0	0	0	**1**	**1**	0
0	1	0	1	0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	**1**	0	**1**	**1**	0	0	0	0	**0**	**1**	1
1	**1**	0	**0**	**1**	0	0	0	0	**1**	**0**	0
0	0	0	0	0	0	0	0	0	0	0	0

LP Formulation

Maximize $\Sigma_e\, y_e$

Subject to:

$\Sigma_v\, x_v \leq 1$

$y_e \leq x_v$ [for all nodes v, edges e, such that e is incident on v]

$x, y \geq 0$

LP Formulation

Maximize $\Sigma_e\, y_e$

Subject to:

$\Sigma_v\, x_v \leq 1$

x_v indicates whether node v
Is part of S

$y_e \leq x_v$ [for all nodes v, edges e, such that e is incident on v]

$x, y \geq 0$

LP Formulation

Maximize $\Sigma_e \, y_e$

y_e indicates whether edge e
Is part of E(S)

Subject to:

$\Sigma_v \, x_v \leq 1$

x_v indicates whether node v
Is part of S

$y_e \leq x_v$ [for all nodes v, edges e, such that e is incident on v]

$x, y \geq 0$

LP Formulation

Maximize $\Sigma_e\, y_e$

y_e indicates whether edge e
Is part of E(S)

Subject to:

$\Sigma_v\, x_v \le 1$

x_v indicates whether node v
Is part of S

$y_e \le x_v$ [for all nodes v, edges e, such that e is incident on v]

Edge e can be in E(S) only if its
endpoints are in S

$x, y \ge 0$

Maximizing $\Sigma_e y_e$ while setting $\Sigma_v x_v \leq 1$ maximizes density

LP Formulation

y_e indicates whether edge e
Is part of E(S)

Maximize $\Sigma_e y_e$

Subject to:

x_v indicates whether node v
Is part of S

$\Sigma_v x_v \leq 1$

$y_e \leq x_v$ [for all nodes v, edges e, such that e is incident on v]

$x, y \geq 0$

Edge e can be in E(S) only if its endpoints are in S

Maximizing $\Sigma_e\ y_e$ while setting $\Sigma_v\ x_v \leq 1$ maximizes density

LP Formulation

y_e indicates whether edge e
Is part of E(S)

Maximize $\Sigma_e\ y_e$

Subject to:

x_v indicates whether node v
Is part of S

$\Sigma_v\ x_v \leq 1$

$y_e \leq x_v$ [for all nodes v, edges e, such that e is incident on v]

$x, y \geq 0$

Edge e can be in E(S) only if its
endpoints are in S

The LP has NO INTEGRALITY GAP

General Direction for DSG

- Write the dual of the LP, and solve it on MapReduce
- PST type algorithms: Perform multiplicative updates of dual weights. Powerful primal-dual technique, with many applications in online, parallelized, and centralized algorithms.
- Approach: formulate the dual in a form suitable for PST; reduce width for efficiency; increase width for obtaining the primal back from the dual

[Plotkin, Shmoys, Tardos; 1995]

[General exposition: Arora, Hazan, Kale; 2010]

[Many updates, variants: eg. Garg, Konemann 1998]

The Primal and its Dual

Maximize $\Sigma_e \, y_e$

Subject to:

$\Sigma_v \, x_v \leq 1$ **[D]**

$y_e \leq x_v$ **[$\text{®}_{e,v}$]**

$x, y \geq 0$

Minimize D

Subject to:

$\text{®}_{e,v} + \text{®}_{e,w} \geq 1$ **[y_e]**

[for all edges e = (v,w)]

$\Sigma_{e \text{ incident on } v} \, \text{®}_{e,v} \leq D$ **[x_v]**

[for all nodes v]

$\text{®}, D \geq 0$

The Primal and its Dual

Maximize $\Sigma_e\, y_e$

Subject to:

$\Sigma_v\, x_v \leq 1$ **[D]**

$y_e \leq x_v$ **[$\circledR_{e,v}$]**

$x, y \geq 0$

USEFUL FACT: An approximate solution to **this** dual results in an approximate solution to the primal

Minimize D

Subject to:

$\circledR_{e,v} + \circledR_{e,w} \geq 1$ **[y_e]**

[for all edges e = (v,w)]

$\Sigma_{e \text{ incident on } v}\, \circledR_{e,v} \leq D$ **[x_v]**

[for all nodes v]

$\circledR, D \geq 0$

Solving the Dual

~~Minimize D~~ Guess D

~~Subject to:~~ Try to find ®, s.t.

$® _{e,v} + ® _{e,w} \geq 1$

[for all edges e = (v,w)]

$\Sigma_{e \text{ incident on } v}\, ® _{e,v} \leq D$

[for all nodes v]

$® \geq 0$

® 2 P

Solving the Dual

PST: Solve the dual using calls to the following **oracle**, for given y_e:

Maximize $\Sigma_e\, y_e(®_{e,u} + ®_{e,v})$

s.t. ® 2 **P**

Width, ½ = max $\{®_{e,v} + ®_{e,w}\}$
s.t. ® 2 **P**

Guarantee: We get a (1+²)-approximation in O((½ log N)/²²) steps

~~Minimize D~~ Guess D

~~Subject to:~~ Try to find ®, s.t.

$®_{e,v} + ®_{e,w} \geq 1$

[for all edges e = (v,w)]

$\Sigma_{e \text{ incident on } v}\, ®_{e,v} \leq D$

[for all nodes v]

® ≥ 0

® 2 **P**

The Dual Oracle on MapReduce

- Need to compute the **oracle** in each iteration:

  Maximize $\Sigma_e\, y_e(\circledR_{e,u} + \circledR_{e,v})$, subject to:

  $$\Sigma_{e \text{ incident on } v}\, \circledR_{e,v} \le D;\ \circledR \ge 0$$

- Maps well to MapReduce
  - Map(edge e = (u,v), y_e):
    EMIT(u, (e, y_e)); Emit(v, (e, y_e))
  - Reduce(node u, <(e$_1$, y_{e1}), …>):
    Find the largest y_e in the values list, and output $\circledR_{e,u}$ = D and everything else is implicitly 0
  - Key complexity: $O(d_{MAX})$; sequential complexity: $O(M)$

Solving the Dual

PST: Solve the dual using calls to the following **oracle**, for given y_e:

Maximize $\Sigma_e\, y_e$(®$_{e,u}$ + ®$_{e,v}$)

s.t. ® 2 **P**

Width, ½ = max {®$_{e,v}$ + ®$_{e,w}$}
s.t. ® 2 **P**

Guarantee: We get a (1+²)-approximation in O((½ log N)/²²) steps

~~Minimize D~~ Guess D

~~Subject to:~~ Try to find ®, s.t.

®$_{e,v}$ + ®$_{e,w}$ ≥ 1
[for all edges e = (v,w)]

$\Sigma_{e \text{ incident on } v}$ ®$_{e,v}$ ≤ D
[for all nodes v]

® ≥ 0

® 2 **P**

Solving the Dual

PST: Solve the dual using calls to the following **oracle**, for given y_e:

Maximize $\Sigma_e\ y_e$(®$_{e,u}$ + ®$_{e,v}$)

s.t. ® 2 **P**

Width, ½ = max {®$_{e,v}$ + ®$_{e,w}$}
s.t. ® 2 **P**

Guarantee: We get a (1+²)-approximation in O((½ log N)/²²) steps

First Problem: ½ is too large (as large as D)

~~Minimize D~~ Guess D

~~Subject to:~~ Try to find ®, s.t.

®$_{e,v}$ + ®$_{e,w}$ ≥ 1
[for all edges e = (v,w)]

® 2 **P**

$\Sigma_{e \text{ incident on } v}$ ®$_{e,v}$ ≤ D
[for all nodes v]

® ≥ 0

Solving the Dual: Reducing Width

~~Minimize D~~ Guess D

~~Subject to:~~ Try to find ®, s.t.

$®_{e,v} + ®_{e,w} \geq 1$

[for all edges e = (v,w)]

® 2 P

$\Sigma_{e \text{ incident on } v}\, ®_{e,v} \leq D$

[for all nodes v]

$® \geq 0$; $\boxed{® \leq 1}$

Solving the Dual: Reducing Width

Width ½ = max $\{®_{e,v} + ®_{e,w}\}$
s.t. ® 2 **P**

The optimum solution to the dual LP never sets any $®_{e,u}$ to be larger than 1, and hence, adding the "® ≤ 1" constraints does not change the dual solution

Next problem: It no longer holds that an approximate dual leads to an approximate primal

~~Minimize D~~ Guess D

~~Subject to:~~ Try to find ®, s.t.

$®_{e,v} + ®_{e,w} \geq 1$

[for all edges e = (v,w)]

® 2 **P**

$\Sigma_{e \text{ incident on } v}\, ®_{e,v} \leq D$

[for all nodes v]

® ≥ 0; ® ≤ 1

Preserving Approximation

Replace "® ≤ 1" with "® ≤ 2"

The width increases by only O(1), but:

Technical Lemma: A $(1+{}^{2})$-approximate solution to the dual results in a $(1+O({}^{2}))$-approximate solution to the primal

~~Minimize D~~ Guess D

~~Subject to:~~ Try to find ®, s.t.

®$_{e,v}$ + ®$_{e,w}$ ≥ 1

[for all edges e = (v,w)]

$\Sigma_{\text{e incident on v}}$ ®$_{e,v}$ ≤ D

[for all nodes v]

® ≥ 0; $\boxed{\text{® ≤ 2}}$

® 2 P

Performance

- $O((\log N)/^{22})$ iterations
- Each iteration:
  - Reduce-key complexity: $O(d_{MAX})$
  - Sequential complexity: $O(M)$
- The greedy algorithm takes $O((\log N)/^{2})$ iterations, but gives a $(2+^{2})$-approximation
- Extends to fractional matchings, and directed graphs

[Goel, Munagala; 2013]

Data Model #2: Active DHT

- DHT (Distributed Hash Table): Stores key-value pairs in main memory on a cluster such that machine H(key) is responsible for storing the pair (key, val)
- Active DHT: In addition to lookups and insertions, the DHT also supports running user-specified code on the (key, val) pair at node H(key)
- Like Continuous Map Reduce, but reducers can talk to each other

Example 2: PageRank

- An early and famous search ranking rule, from Brin et al. Given a directed graph $G=(V,E)$, with N nodes, M edges, $d(w)$ = number of edges going out of node w, ^{2} = teleport probability, $¼(v)$ = PageRank of node v.

$$¼(v) = {}^2/N + (1-{}^2)\ §_{(w,v)\ 2\ E}\ (¼(w)/d(w))$$

- Equivalently: Stationary distribution of random walk that teleports to a random node with probability ^{2}
- Consequence: The Monte Carlo method. It is sufficient to do

$$R = O(\log N)$$

random walks starting at every node , where each random walk terminates upon teleport

PageRank in Social Networks

- Interpretation in a social network: You are highly reputed if other highly reputed individuals follow you (or are your friends)
- Updates to social graph are made in real-time
  - As opposed to a batched crawl process for web search
- Real-time updates to PageRank are important to capture trending events
- Goal: Design an algorithm to update PageRank incrementally (i.e. upon an edge arrival) in an Active DHT

t-th edge arrival: Let (u_t, v_t) denote the arriving edge, $d_t(v)$ denote the out-degree of node v, and ¼$_t(v)$ its PageRank

Incremental PageRank in Social Networks

- Two Naïve approaches for updating PageRank:
  - Run the power iteration method from scratch. Set $¼_0(v) = 1/N$ for every node v, and then compute

$$¼_r(v) = {}^2/N + (1-{}^2\ §_{(w,v)\ 2\ E}\ (¼_{r-1}(w)/d(w))$$

    R times, where R ¼ $(\log N)/{}^2$
  - Run the Monte Carlo method from scratch each time
- Running time £((M/²) log N) and £((N/²) log N), respectively, per edge arrival
- Heuristic improvements are known, but nothing that provably gives significantly better running time

Incremental Monte Carlo using DHTs

- Initialize the Active DHT: Store the social graph and $R = \log N$ random walks starting at each node
- At time t, for every random walk passing through node u_t, shift it to use the new edge (u_t, v_t) with probability $1/d_t(u_t)$
- Time/number of network-calls for each re-routing: $O(1/\varepsilon)$
- Claim: This faithfully maintains R random walks after arbitrary edge arrivals.
- Observe that we need the graph and the stored random walks to be available in fast distributed memory; this is a reasonable assumption for social networks, though not necessarily for the web-graph.

An Average Case Analysis

- Assume that the edges of the graph are chosen by an adversary, but presented in random order
- Technical consequence: E[¼$_t$(u_t)/d_t(u_t)] = 1/t
- Expected # of random walks rerouted at time t

  = (Expected # of Random Walks through node u_t)/d_t(u_t)

  = E[(¼$_t$(u_t) (RN/^{2}))/d_t(u_t)]

  = (RN/^{2})/t

) Number of network calls made = O(RN/(^{22}t))
- Amount of extra work done$^{(*)}$ per edge arrival goes to 0 !!
- Work done over all M edge arrivals goes to O((N/^{22}) $\log^2$ N)
- Compare to £ ((N/^{2}) log N) per edge arrival for Naïve Monte Carlo

An Average Case Analysis

- Assume that the edges of the graph are chosen by an adversary, but presented in random order
- Technical consequence: $E[¼_t(u_t)/d_t(u_t)] = 1/t$

ROUGH INTUITION

We "expect" $¼_t(u_t)$ to be around $1/N$
We "expect" $1/d_t(u_t)$ to be around N/t

The ratio of "expectations" is $1/t$

The random order ensures that the expectation of the ratios is also $1/t$

An Average Case Analysis

- Assume that the edges of the graph are chosen by an adversary, but presented in random order
- Technical consequence: E[¼$_t$(u_t)/d_t(u_t)] = 1/t
- Expected # of random walks rerouted at time t
  = (Expected # of random walks through node u_t)/d_t(u_t)
  = E[(¼$_t$(u_t)¢(RN/²))/d_t(u_t)]
  = (RN/²)/t
) Number of network calls made = O(RN/(²²t))
- Amount of extra work done(*) per edge arrival goes to 0 !!
- Work done over all M edge arrivals goes to O((N/²²) log^2 N)
- Compare to £ ((N/²) log N) per edge arrival for Naïve Monte Carlo

Incremental PageRank: Summary

- The random order assumption is much weaker than assuming generative models such as Preferential Attachment, which also satisfy the random order assumption
- The technical consequence can be verified empirically
- The result does not hold for adversarial arrival order
- The analysis carefully pairs an appropriate computation/data model (Active DHTs) with minimal assumptions on the social network

[Bahmani, Chowdhury, Goel; 2011]

Directions and Open Problems

- An Oracle for Personalized PageRank
- Real-time Social Search
  - Partial progress for distance based relevance measures [Bahmani, Goel; 2012]
- Mixed Algorithms: Algorithms that can run on either MapReduce or Active DHTs (or a combination) seamlessly
- Additional research interests: Randomized Algorithms; Collaboration, trust, and mistrust in social networks; Internet Commerce