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, <val₁, val₂, ..]*) pair.

- **REDUCE**: A UDF that processes this grouped *(key, <val₁, val₂, ..]*) 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]
• **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 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 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 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 $d_{\text{MAX}}$
  - 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 $(2+\epsilon)$-approximation known on MapReduce
  - $O((\log N)/\epsilon)$-phases
  - Each phase has sequential complexity $O(M)$ and key complexity $O(d_{\text{MAX}})$

[Bahmani, Kumar, Vassilvitskii; 2012]
<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
LP Formulation

Maximize $\sum_e y_e$

Subject to:

$\sum_v x_v \leq 1$

$y_e \leq x_v$ [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:
$\Sigma_v x_v \leq 1$

$y_e \leq x_v$ [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:

$\Sigma_v x_v \leq 1$

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

$x, y \geq 0$

$y_e$ indicates whether edge $e$ is part of $E(S)$

$x_v$ indicates whether node $v$ is part of $S$
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 $v$]

$x, y \geq 0$

$y_e$ indicates whether edge $e$ is part of $E(S)$

$x_v$ indicates whether node $v$ is part of $S$

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

Maximize $\sum_e y_e$

Subject to:

$\sum_v x_v \leq 1$

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

$x, y \geq 0$

$y_e$ indicates whether edge $e$ is part of $E(S)$

$x_v$ indicates whether node $v$ is part of $S$

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

Maximizing $\sum_e y_e$ while setting $\sum_v x_v \leq 1$ maximizes density
Maximize $\sum_e y_e$

Subject to:

$\sum_v x_v \leq 1$

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

$x, y \geq 0$

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]
The Primal and its Dual

Maximize $\sum_e y_e$
Subject to:
$\sum_v x_v \leq 1$ [D]
y_e \leq x_v [\theta e,v]
x, y \geq 0

Minimize $D$
Subject to:
$\theta e,v + \theta e,w \geq 1$ [y_e] [for all edges e = (v,w)]
$\sum_e \text{incident on } v \theta e,v \leq D$ [x_v] [for all nodes v]
${\theta}, D \geq 0$
The Primal and its Dual

Maximize $\Sigma_v y_v$
Subject to:
$\Sigma_v x_v \leq 1$ \hfill $[D]$ \hfill $[y_e]$
$y_e \leq x_v$ \hfill $[\mathbb{R}_{e,v}]$
$x, y \geq 0$

Minimize $D$
Subject to:
$\mathbb{R}_{e,v} + \mathbb{R}_{e,w} \geq 1$ \hfill $[y_e]$
\hfill $[\text{for all edges } e = (v,w)]$
$\Sigma_e \text{ incident on } v \mathbb{R}_{e,v} \leq D$ \hfill $[x_v]$
\hfill $[\text{for all nodes } v]$

$\mathbb{R}, D \geq 0$

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

Minimize $D$

Guess $D$

Subject to: Try to find $\mathcal{R}$, s.t.

$\mathcal{R}_{e,v} + \mathcal{R}_{e,w} \geq 1$

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

$\sum_{e \text{ incident on } v} \mathcal{R}_{e,v} \leq D$

[for all nodes $v$]

$\mathcal{R} \geq 0$
Solving the Dual

PST: Solve the dual using calls to the following \textbf{oracle}, for given $y_e$:

Maximize $\sum e \ y_e(\mathcal{R}_{e,u} + \mathcal{R}_{e,v})$

s.t. $\mathcal{R} \in \mathbb{P}$

Width, $\frac{1}{2} = \max \{\mathcal{R}_{e,v} + \mathcal{R}_{e,w}\}$

s.t. $\mathcal{R} \in \mathbb{P}$

 Guarantee: We get a $(1+\epsilon)$-approximation in $O((\frac{1}{2} \log N)/\epsilon^2)$ steps

\begin{align*}
\text{Minimize} & \quad D \\
\text{Guess} & \quad D \\
\text{Subject to:} & \quad \text{Try to find } \mathcal{R}, \text{ s.t.} \\
\mathcal{R}_{e,v} + \mathcal{R}_{e,w} & \geq 1 \\
& \quad \text{[for all edges } e = (v,w)\text{]} \\
\sum e \text{ incident on } v \quad \mathcal{R}_{e,v} & \leq D \\
& \quad \text{[for all nodes } v\text{]} \\
\mathcal{R} & \geq 0
\end{align*}
The Dual Oracle on MapReduce

• Need to compute the oracle in each iteration:

\[
\text{Maximize } \sum_e y_e (\mathcal{R}_{e,u} + \mathcal{R}_{e,v}), \text{ subject to:}
\]

\[
\sum_e \text{incident on } v \mathcal{R}_{e,v} \leq D; \mathcal{R} \geq 0
\]

• Maps well to MapReduce
  – Map(edge \( e = (u,v), y_e \)):
    \text{EMIT}(u, (e, y_e)); \text{Emit}(v, (e, y_e))
  – Reduce(node \( u, \langle (e_1, y_{e_1}), ... \rangle \)):
    Find the largest \( y_e \) in the values list, and output \( \mathcal{R}_{e,u} = D \) and everything else is implicitly 0
  – Key complexity: \( O(d_{\text{MAX}}) \); sequential complexity: \( O(M) \)
Solving the Dual

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

Maximize $\sum_e y_e (\mathcal{R}_{e,u} + \mathcal{R}_{e,v})$

s.t. $\mathcal{R}_2 \mathcal{P}$

Width, $\frac{1}{2} = \max \{\mathcal{R}_{e,v} + \mathcal{R}_{e,w}\}$

s.t. $\mathcal{R}_2 \mathcal{P}$

Guarantee: We get a $(1+\sqrt{2})$-approximation in $O((\frac{1}{2} \log N)/\mathcal{R}^2)$ steps

Minimize $D$ Guess $D$

Subject to: Try to find $\mathcal{R}$, s.t.

$\mathcal{R}_{e,v} + \mathcal{R}_{e,w} \geq 1$

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

$\sum_e$ incident on $v$ $\mathcal{R}_{e,v} \leq D$

[for all nodes $v$]

$\mathcal{R} \geq 0$
Solving the Dual

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

Maximize $\sum e y_e (\mathcal{R}_{e,u} + \mathcal{R}_{e,v})$

s.t. $2 \mathcal{P}$

Width, $\frac{1}{2} = \max \{ \mathcal{R}_{e,v} + \mathcal{R}_{e,w} \}$

s.t. $2 \mathcal{P}$

Guarantee: We get a $(1+\varepsilon^2)$-approximation in $O((\frac{1}{2} \log N)/\varepsilon^2)$ steps

First Problem: $\frac{1}{2}$ is too large (as large as $D$)

Minimize $D$

Guess $D$

Subject to: Try to find $\mathcal{R}$, s.t.

$\mathcal{R}_{e,v} + \mathcal{R}_{e,w} \geq 1$

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

$\sum e$ incident on $v$ $\mathcal{R}_{e,v} \leq D$

[for all nodes $v$]

$\mathcal{R} \geq 0$
Solving the Dual: Reducing Width

Minimize $D$

Guess $D$

Subject to: Try to find $\mathcal{R}$, s.t.

$\mathcal{R}_{e,v} + \mathcal{R}_{e,w} \geq 1$

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

$\sum_{e \text{ incident on } v} \mathcal{R}_{e,v} \leq D$

[for all nodes $v$]

$\mathcal{R} \geq 0; \quad \mathcal{R} \leq 1$
Solving the Dual: Reducing Width

Width $\frac{1}{2} = \max \{\mathbb{R}_{e,v} + \mathbb{R}_{e,w}\}$

s.t. $\mathbb{R} \leq 1$

The optimum solution to the dual LP never sets any $\mathbb{R}_{e,u}$ to be larger than 1, and hence, adding the “$\mathbb{R} \leq 1$” 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 $\mathbb{R}$, s.t.

$\mathbb{R}_{e,v} + \mathbb{R}_{e,w} \geq 1$

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

$\sum_{e \text{ incident on } v} \mathbb{R}_{e,v} \leq D$

[for all nodes $v$]

$\mathbb{R} \geq 0; \mathbb{R} \leq 1$
Preserving Approximation

Replace “$\mathcal{R} \leq 1$” with “$\mathcal{R} \leq 2$”

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

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

\[
\text{Minimize } D \quad \text{Guess } D
\]

\[
\text{Subject to: } \text{Try to find } \mathcal{R}, \text{ s.t.}
\]

\[
\mathcal{R}_{e,v} + \mathcal{R}_{e,w} \geq 1
\]

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

\[
\sum_{e \text{ incident on } v} \mathcal{R}_{e,v} \leq D
\]

[for all nodes $v$]

$\mathcal{R} \geq 0; \quad \mathcal{R} \leq 2$
Performance

- $O((\log N)^2)$ iterations
- Each iteration:
  - Reduce-key complexity: $O(d_{\text{MAX}})$
  - 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 $H(\text{key})$ is responsible for storing the pair ($\text{key}, \text{val}$)

- Active DHT: In addition to lookups and insertions, the DHT also supports running user-specified code on the ($\text{key}, \text{val}$) pair at node $H(\text{key})$

- 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 $w$, $\beta$ = teleport probability, $\frac{\beta}{4}(v)$ = PageRank of node $v$.

$$\frac{\beta}{4}(v) = \frac{\beta}{N} + (1-\beta) \sum_{(w,v) \in E} \left( \frac{\frac{\beta}{4}(w)}{d(w)} \right)$$

• Equivalently: Stationary distribution of random walk that teleports to a random node with probability $\beta$

• 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 $\frac{1}{4}_t(v)$ its PageRank
Incremental PageRank in Social Networks

- Two Naïve approaches for updating PageRank:
  - Run the power iteration method from scratch. Set $\frac{1}{4_0}(v) = \frac{1}{N}$ for every node $v$, and then compute
    \[
    \frac{1}{4_r}(v) = \frac{2}{N} + (1 - 2 \sum_{(w,v)} \frac{1}{2} (\frac{1}{4_{r-1}}(w)/d(w)))
    \]
    $R$ times, where $R \frac{1}{4} (\log N)^2$
  - Run the Monte Carlo method from scratch each time

- Running time $\mathcal{O}(\frac{M}{\Delta^2} \log N)$ and $\mathcal{O}(\frac{N}{\Delta^2} \log N)$, 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: $O(1/\epsilon)$.
- 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: \[ E[\frac{1}{4_t(u_t)}/d_t(u_t)] = \frac{1}{t} \]

• Expected # of random walks rerouted at time \( t \)
  \[ = (\text{Expected # of Random Walks through node } u_t)/d_t(u_t) \]
  \[ = E[\frac{(\frac{1}{4_t(u_t)} (RN/2))/d_t(u_t)}{d_t(u_t)}] \]
  \[ = \frac{(RN/2)}{t} \]

) Number of network calls made = \( O(RN/(2^2t)) \)

• Amount of extra work done(*) per edge arrival goes to 0 !!

• Work done over all \( M \) edge arrivals goes to \( O((N/2^2) \log^2 N) \)

• Compare to \( \mathcal{L} ((N/2) \log N) \) 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: \( E[\frac{1}{4_t(u_t)/d_t(u_t)}] = \frac{1}{t} \)

\[
E[\frac{1}{4_t(u_t)/d_t(u_t)}] = \frac{E\left(\frac{1}{4_t(u_t)} \right)}{d_t(u_t)} = \frac{RN}{t^2} \\
\text{Number of network calls made} = O\left(\frac{RN}{t^2}\right)
\]

• Amount of extra work done (*) per edge arrival goes to 0 !!
• Work done over all \( M \) edge arrivals goes to \( O\left(\frac{N}{2t} \log_2 N\right) \)
• Compare to \( \Omega\left(\frac{N}{2t} \log N\right) \) per edge arrival for Naïve Monte Carlo

ROUGH INTUITION

We “expect” \( \frac{1}{4_t(u_t)} \) to be around \( \frac{1}{N} \)
We “expect” \( \frac{1}{d_t(u_t)} \) to be around \( \frac{N}{t} \)

The ratio of “expectations” is \( \frac{1}{t} \)

The random order ensures that the expectation of the ratios is also \( \frac{1}{t} \)
An Average Case Analysis

- Assume that the edges of the graph are chosen by an adversary, but presented in random order.
- Technical consequence: \( E\left[\frac{1}{4_t(u_t)}/d_t(u_t)\right] = 1/t \)
- Expected # of random walks rerouted at time \( t \)
  \[= \left(\text{Expected # of random walks through node } u_t\right)/d_t(u_t)\]
  \[= E\left[\frac{1}{4_t(u_t)\phi(RN/2)}/d_t(u_t)\right]\]
  \[= (RN/2)/t\]

  ) Number of network calls made = \( O(RN/(2^2 t)) \)
- Amount of extra work done (*) per edge arrival goes to 0 !!
- Work done over all \( M \) edge arrivals goes to \( O((N/2^2) \log^2 N) \)
- Compare to \( E ((N/2) \log N) \) 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