Efficient Primal-Dual Graph Algorithms for Map Reduce

Kamesh Munagala

Duke University

Joint work with
Bahman Bahmani
Ashish Goel, Stanford

Modern Data Models

- Over the past decade, many commodity distributed computing platforms have emerged
  - 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 often negligible compared to the cost of data transfer
  - Take advantage of aggregation in disk and network operations

Map Reduce (or Hadoop)

- Immensely successful idea that transformed offline analytics and bulk-data processing.
- **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, <val_1, val_2, ..>) pair
- **REDUCE:** A UDF that processes this grouped (key, <val_1, val_2, ..>) pair for a single key. Many reducers can run in parallel.

[Dean and Ghemawat; 2005]

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
- Number of MapReduce phases

[Goel, Munagala, 2012; Andoni, Nikolov, Onak, Yaroslavtsev, 2014]

Complexity

THE AMOUNT OF WORK DONE PER COMPUTER IF WE HAD INFINITELY MANY COMPUTERS

- 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
- Number of MapReduce phases

[Goel, Munagala, 2012; Andoni, Nikolov, Onak, Yaroslavtsev, 2014]

Complexity Measures

- Key-Complexity:
  - The maximum size
  - The amount of time
  - The memory required to pr

THE AMOUNT OF WORK DONE ON A SINGLE COMPUTER

- Sequential Complexity:
  - The total time needed by all the mappers and reducers together
  - The total output produced by all the mappers and reducers

- Number of MapReduce phases

[Goel, Munagala, 2012; Andoni, Nikolov, Onak, Yaroslavtsev, 2014]

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
- Number of MapReduce phases

SHUFFLE SIZE

[Goel, Munagala, 2012; Andoni, Nikolov, Onak, Yaroslavtsev, 2014]

Complexity Measures

- Key-Complexity
  - The maximum siz
  - The amount of tim
  - The memory required to
- Sequential Complexity:
  - The total time needed by all the mappers and reducers together
  - The total output produced by all the mappers and reducers
- Number of MapReduce phases

TOTAL MEMORY IS NOT A COMPLEXITY MEASURE SINCE THAT DEPENDS ON THE NUMBER OF REDUCERS AND MAPPERS

[Goel, Munagala, 2012; Andoni, Nikolov, Onak, Yaroslavtsev, 2014]

Complexity Measures

- Key-Complexity:
  - The maximum size of
  - The amount of time
  - The memory require
- Sequential Complexity:
  - The total time needed by all the mappers and reducers together
  - The total output produced by all the mappers and reducers
- Number of MapReduce phases

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

[Goel, Munagala, 2012; Andoni, Nikolov, Onak, Yaroslavtsev, 2014]

In this talk…

- Parallel algorithms for approximately solving linear programs
  - Exact model – MapReduce, PRAM – does not matter that much
- Several basic model-independent open questions

Densest Subgraph Problem (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, computational biology
- Can be solved in polynomial time
- $(2+\varepsilon)$-approximation known on MapReduce
  - $O((\log N)/\varepsilon)$-phases
  - Sequential complexity $O(M)$ per phase
  - Key complexity $O(d_{MAX})$

  [Bahmani, Kumar, Vassilvitskii; 2012]

LP Formulation

Maximize $\Sigma_e y_e$

Subject to:

$\Sigma_v x_v \leq 1$

$y_e \leq x_v$ [for all vertices v, edge 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

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

x_v indicates whether node v is part of S

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

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

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
- Dual is a "mixed packing/covering" problem
  - Maximum concurrent flow in a certain type of bipartite graph
- PST-type algorithms: Perform ***multiplicative updates*** of dual weights till constraints are satisfied
  - Powerful primal-dual technique
  - Also called "mirror descent" or "Frank-Wolfe" type algorithms

[Plotkin, Shmoys, Tardos; 1991, Grigoriadis, Khachiyan; 1993, Garg, Konemann; 1997, Freund, Schapire; 1997]

[General exposition: Arora, Hazan, Kale; 2012]

Parallel Implementations

- [Young 2001] $O((\log^3 N)/\varepsilon^4)$ phases for $(1+\varepsilon)$ approximation

- Other parallel variants known for *pure* packing/ covering problems

  [Luby, Nisan 1993; Awerbuch, Khandekar 2009; Allen-Zhu, Orecchia 2015]

  - Not really applicable to our mixed packing/covering problem

- Algorithms prior to ours needed $\Omega(1/\varepsilon^4)$ phases to achieve $(1+\varepsilon)$ approximation
  - Same dependence holds even for pure packing/covering problems

Our Approach

- Formulate dual in a form suitable for applying multiplicative weights (or PST)
- Reduce "width" for efficiency
  - Width is roughly the magnitude of the gradient of a potential function that measures violation in constraints
  - Mirror descent algorithms are more efficient when gradient has small magnitude
- Reducing width leads to infeasible primal!
- Increase width for obtaining the primal back from the dual

The Primal and its Dual

Maximize $\Sigma_e y_e$

Subject to:

$\Sigma_v x_v \leq 1$

[D]

$y_e \leq x_v$

$x, y \geq 0$

USEFUL FACT: [illegible]
solution to this dual [illegible]
approximate solution [illegible]

The Primal and its Dual

Maximize $\Sigma_e y_e$

Subject to:

$\Sigma_v x_v \leq 1$

[D]

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

$x, y \geq 0$

USEFUL FACT: [illegible]
solution to this dual [illegible]
approximate solution [illegible]

The Primal and its Dual

Maximize $\Sigma_e y_e$

Subject to:

$\Sigma_v x_v \leq 1$

[D]

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

$x, y \geq 0$

Minimize D

Subject to:

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

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

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

[for all nodes v]

$\alpha, D \geq 0$

USEFUL FACT: ...
solution to this dual ...
approximate solution ...

The Primal and its Dual

Maximize $\Sigma_e y_e$

Subject to:

$\Sigma_v x_v \leq 1$ [D]

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

$x, y \geq 0$

Minimize D

Subject to:

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

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

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

[for all nodes v]

$\alpha, D \geq 0$

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 α, s.t.

$\alpha_{e,v} + \alpha_{e,w} \geq 1$

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

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

[for all nodes v]

$\alpha \geq 0$

$\alpha \in P$

Solving the Dual

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

Maximize $\sum y_e(\alpha_{e,U} + \alpha_{e,v})$
s.t. $\alpha \in P$

Width, $\rho = \max \{\alpha_{e,v} + \alpha_{e,w}\}$ s.t.
$\alpha \in P$

Guarantee:
We get $(1+\varepsilon)$-approx. in $O(\rho (\log N)/\varepsilon^2)$ steps

~~Minimize D~~ Guess D

~~Subject to:~~ Try to find α, s.t.

$\alpha_{e,v} + \alpha_{e,w} \geq 1$
[for all edges e = (v,w)]

$\Sigma_{e \text{ incident on } v} \alpha_{e,v} \leq D$
[for all nodes v]

$\alpha \geq 0$

$\alpha \in P$

The Dual Oracle on MapReduce

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

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

  $\Sigma_{\text{e incident on v}} \alpha_{e,v} \leq D; \alpha \geq 0$

- 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
    Output $\alpha_{e,u}$ = 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 $\sum y_e(\alpha_{e,U} + \alpha_{e,v})$
s.t. $\alpha \in P$

Width, $\rho = \max \{\alpha_{e,v} + \alpha_{e,w}\}$ s.t.
$\alpha \in P$

Guarantee:
We get $(1+\varepsilon)$-approx. in $O(\rho (\log N)/\varepsilon^2)$ steps

~~Minimize D~~ Guess D

~~Subject to:~~ Try to find α, s.t.

$\alpha_{e,v} + \alpha_{e,w} \geq 1$
[for all edges e = (v,w)]

$\Sigma_{e \text{ incident on } v} \alpha_{e,v} \leq D$
[for all nodes v]

$\alpha \geq 0$

$\alpha \in P$

Solving the Dual

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

Maximize $\sum y_e(\alpha_{e,U} + \alpha_{e,v})$
s.t. $\alpha \in P$

Width, $\rho = \max \{\alpha_{e,v} + \alpha_{e,w}\}$ s.t.
$\alpha \in P$

Guarantee:
We get $(1+\varepsilon)$-approx. in $O(\rho (\log N)/\varepsilon^2)$ steps

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

~~Minimize D~~ Guess D

~~Subject to:~~ Try to find α, s.t.

$\alpha_{e,v} + \alpha_{e,w} \geq 1$
[for all edges e = (v,w)]

$\Sigma_{e \text{ incident on } v} \alpha_{e,v} \leq D$
[for all nodes v]

$\alpha \geq 0$

$\alpha \in P$

Solving the Dual: Reducing Width

~~Minimize D~~ Guess D

~~Subject to:~~ Try to find α, s.t.

$\alpha_{e,v} + \alpha_{e,w} \geq 1$

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

$\alpha \in P$

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

[for all nodes v]

$\alpha \geq 0$; $\alpha \leq 1$

Solving the Dual: Reducing Width

Width, $\rho = \max \{\alpha_{e,v} + \alpha_{e,w}\}$ s.t.

$\alpha \in P$

The optimum solution to the dual LP never sets any $\alpha_{e,u}$ to be larger than 1, and hence, adding the "$\alpha \leq 1$" 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 α, s.t.

$\alpha_{e,v} + \alpha_{e,w} \geq 1$

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

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

[for all nodes v]

$\alpha \geq 0$; $\alpha \leq 1$

$\alpha \in P$

Preserving Approximation

Replace "$\alpha \le 1$" with "$\alpha \le 2$"

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

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

~~Minimize D~~ Guess D

~~Subject to:~~ Try to find α, s.t.

$$\alpha_{e,v} + \alpha_{e,w} \ge 1 \quad \text{[for all edges } e = (v,w)\text{]}$$

$$\Sigma_{e \text{ incident on } v} \alpha_{e,v} \le D \quad \text{[for all nodes v]}$$

$$\alpha \ge 0; \alpha \le 2$$

$\alpha \in P$

Performance

- $O((\log N)/\varepsilon^2)$ iterations
- Each iteration:
  - Reduce-key complexity: $O(d_{MAX})$
  - Sequential complexity: $O(M)$
- In contrast, GREEDY takes $O((\log N)/\varepsilon)$ iterations, but gives a $(2+\varepsilon)$-approximation

  Recent O(1) phase algorithms need O(N) reduce-key complexity [Bhattacharya et al. '15]
- **Other Results:** $(1+\varepsilon)$ approximate max multi-commodity flow when paths are at most L hops
  - Shave an ε from $1/\varepsilon^4$ dependence in [Awerbuch, Khandekar; '07]

Open Questions: I

- [Allen-Zhu, Orecchia; 2015]
  - $1/\varepsilon^3$ dependence for any pure packing and covering problem
  - Clever thresholding of dual oracle to reduce the width
  - Use a combination of gradient descent and mirror descent analysis
- Can we improve these bounds further?
  - Beat $1/\varepsilon^4$ for general mixed packing/covering problems
  - Beat $1/\varepsilon^3$ for pure packing/covering problems
- Efficient distributed algorithm for max concurrent flow with bounded path lengths?

Open Questions: II

- Cautionary Note:
  - In experiments on real networks, GREEDY is actually superior
  - GREEDY needs very few rounds; finds almost optimal solutions
  - PRIMAL-DUAL does not take advantage of graph structure
  - "Warm start" does not seem to speed things up much
- Can we combine combinatorial methods with convex programming techniques?
- Constant approximations with small number of rounds for larger classes of flow/cut problems?