Parallel Local Graph Clustering

Kimon Fountoulakis, joint work with J. Shun, X. Cheng, F. Roosta-Khorasani, M. Mahoney, D. Gleich
University of California Berkeley and Purdue University

Based on
Local graph clustering: motivation
Facebook social network: colour denotes class year

Normalized cuts: finds 20% of the graph

Local graph clustering: finds 3% of the graph

Local graph clustering: finds 17% of the graph

Current algorithms and running time
Global, weakly and strongly local methods

**Global methods:** $O(\text{volume of graph})$
- The workload depends on the size of the graph

**Weakly local methods:** $O(\text{volume of graph})$
- A seed set of nodes is given
- The solution is locally biased to the input seed set
- The workload depends on the size of the graph

**Strongly local methods:** $O(\text{volume of output cluster})$
- A seed set of nodes is given
- The solution is locally biased to the input seed set
Global, weakly and strongly local methods

Cluster quality

We measure cluster quality using

\[
\text{Conductance} := \frac{\text{number of edges leaving cluster}}{\text{sum of degrees of vertices in cluster}}
\]

- Conductance({A,B}) = \frac{2}{2 + 2} = \frac{1}{2}
- Conductance({A,B,C}) = \frac{1}{2 + 2 + 3} = \frac{1}{7}

• The smaller the conductance value the better
• Minimizing conductance is NP-hard, we use approximation algorithms
Cluster quality

We measure cluster quality using

\[ \text{Conductance} := \frac{\text{number of edges leaving cluster}}{\text{sum of degrees of vertices in cluster}} \]

- Conductance\((\{A,B\}) = \frac{2}{2 + 2} = \frac{1}{2} \)
- Conductance\((\{A,B,C\}) = \frac{1}{2 + 2 + 3} = \frac{1}{7} \)

• The smaller the conductance value the better
• Minimizing conductance is NP-hard, we use approximation algorithms
Cluster quality

We measure cluster quality using

\[
\text{Conductance} := \frac{\text{number of edges leaving cluster}}{\text{sum of degrees of vertices in cluster}}
\]

- Conductance(\{A,B\}) = \frac{2}{2 + 2} = \frac{1}{2}
- Conductance(\{A,B,C\}) = \frac{1}{2 + 2 + 3} = \frac{1}{7}

• The smaller the conductance value the better
• Minimizing conductance is NP-hard, we use approximation algorithms
Cluster quality

We measure cluster quality using

\[
\text{Conductance} := \frac{\text{number of edges leaving cluster}}{\text{sum of degrees of vertices in cluster}}
\]

\[
\text{Conductance}({A,B}) = \frac{2}{(2 + 2)} = \frac{1}{2}
\]

\[
\text{Conductance}({A,B,C}) = \frac{1}{(2 + 2 + 3)} = \frac{1}{7}
\]

• The smaller the conductance value the better
• Minimizing conductance is NP-hard, we use approximation algorithms
Cluster quality

We measure cluster quality using

\[ \text{Conductance} := \frac{\text{number of edges leaving cluster}}{\text{sum of degrees of vertices in cluster}} \]

- The smaller the conductance value the better
- Minimizing conductance is NP-hard, we use approximation algorithms

Conductance({A,B}) = \(\frac{2}{2 + 2}\) = 1/2
Conductance({A,B,C}) = \(\frac{1}{2 + 2 + 3}\) = 1/7
Local graph clustering methods

- MQI (strongly local): Lang and Rao, 2004
- Approximate Page Rank (strongly local): Andersen, Chung, Lang, 2006
- Spectral MQI (strongly local): Chung, 2007
- Flow-Improve (weakly local): Andersen and Lang, 2008
- MOV (weakly local): Mahoney, Orecchia, Vishnoi, 2012
- Nibble (strongly local): Spielman and Teng, 2013
- Sweep cut rounding algorithm
Shared memory parallel methods

• We parallelize 4 strongly local spectral methods + rounding
  1. Approximate Page Rank ←→ this talk
  2. Nibble
  3. Deterministic HeatKernel Approximate Page-Rank
  4. Randomized HeatKernel Approximate Page-Rank
  5. Sweep cut rounding algorithm ←→ this talk

• All local methods take various parameters
  - Parallel method 1: try different parameters independently in parallel
  - Parallel method 2: parallelize algorithm for individual run
    ‣ Useful for interactive setting where tweaking of parameters is needed
Approximate Page-Rank
**Personalized Page-Rank vector**

Pick a vertex $u$ of interest and define a vector:

$$s[u] = 1, \quad s[v] = 0 \quad \forall v \neq u$$

a teleportation parameter $0 \leq \alpha \leq 1$ and $W = AD^{-1}$ then the PPR vector is given by solving:

$$((1 - \alpha)W + \alpha s e^T)p = p \quad \Leftrightarrow \quad (I - (1 - \alpha)W)p = \alpha s$$
Approximate Personalized Page-Rank
R. Andersen, F. Chung and K. Lang. Local graph partitioning using Page-Rank, FOCS, 2006

Algorithm idea: iteratively spread probability mass from vector $s$ around the graph.

- $r$ is the residual vector, $p$ is the solution vector
- $\rho > 0$ is tolerance parameter

Run a coordinate descent solver for PPR until: any vertex $u$ satisfies $r[u] \geq -\alpha \rho d[u]$ 

Initialize: $p = 0$, $r = -\alpha s$

While termination criterion is not met do

1. Choose any vertex $u$ where $r[u] < -\alpha \rho d[u]$

2. $p[u] = p[u] - r[u]$


4. $r[u] = 0$

(residual update)

Final step: round the solution $p$ using sweep cut.
Approximate Personalized Page-Rank

R. Andersen, F. Chung and K. Lang. Local graph partitioning using Page-Rank, FOCS, 2006

\[ \frac{||r||_1}{\alpha} = 1, \quad ||p||_1 = 0, \quad \frac{||r||_1}{\alpha} + ||p||_1 = 1 \]

Initialize: \( p = 0, r = -\alpha \)

While termination criterion is not met do

1. Choose any vertex \( u \) where \( r[u] < -\alpha p[u] \)
2. \( p[u] = p[u] - r[u] \)
3. For all neighbours \( v \) of \( u \): \( r[v] = r[v] + (1 - \alpha) r[u] A[u,v] / d[u] \)
4. \( r[u] = 0 \)
Approximate Personalized Page-Rank

R. Andersen, F. Chung and K. Lang. Local graph partitioning using Page-Rank, FOCS, 2006

\[
\frac{\|r\|_1}{\alpha} = 1, \quad \|p\|_1 = 0.1, \quad \frac{\|r\|_1}{\alpha} + \|p\|_1 = 1.1
\]

Initialize: \( p = 0, r = -\alpha \)
While termination criterion is not met do
1. Choose any vertex \( u \) where \( r[u] < -\alpha p[u] \)
2. \( p[u] = p[u] - r[u] \)
3. For all neighbours \( v \) of \( u \): \( r[v] = r[v] + (1-\alpha)r[u]A[u,v]/d[u] \)
4. \( r[u] = 0 \)
Approximate Personalized Page-Rank
R. Andersen, F. Chung and K. Lang. Local graph partitioning using Page-Rank, FOCS, 2006

Initialize: $p = 0$, $r = -\alpha s$, where $s$ is a probability vector

While termination criterion is not met do
1. Choose any vertex $u$ where $r[u] < -\alpha p d[u]$
2. $p[u] = p[u] - r[u]$
4. $r[u] = 0$

$\frac{\|r\|_1}{\alpha} = 0.9$, $\|p\|_1 = 0.1$, $\frac{\|r\|_1}{\alpha} + \|p\|_1 = 1.0$
Approximate Personalized Page-Rank
R. Andersen, F. Chung and K. Lang. Local graph partitioning using Page-Rank, FOCS, 2006

Initialize: $p = 0$, $r = -\alpha$
While termination criterion is not met do
1. Choose any vertex $u$ where $r[u] < -\alpha p d[u]$
2. $p[u] = p[u] - r[u]$
4. $r[u] = 0$

$p = 0.1$, $r = -0.2025$
$p = 0$, $r = -0.6525$
$p = 0.045$, $r = 0$

$\|r\|_1 = 0.855$, $\|p\|_1 = 0.145$, $\frac{\|r\|_1}{\alpha} + \|p\|_1 = 1.0$
Approximate Personalized Page-Rank
R. Andersen, F. Chung and K. Lang. Local graph partitioning using Page-Rank, FOCS, 2006

$$\frac{\|r\|_1}{\alpha} = 0.7897, \quad \|p\|_1 = 0.2103, \quad \frac{\|r\|_1}{\alpha} + \|p\|_1 = 1.0$$

Initialize: $$p = 0, r = -\alpha s$$, where $$s$$ is a probability vector
While termination criterion is not met do
1. Choose any vertex $$u$$ where $$r[u] < -\alpha p d[u]$$
2. $$p[u] = p[u] - r[u]$$
4. $$r[u] = 0$$
Running time APPR

- At each iteration APPR touches a single node and its neighbours
  - Let supp(p) be the support of vector p at termination which satisfies \( \text{vol(supp(p))} \leq \frac{1}{\alpha \rho} \)
  - Overall until termination the work is: \( O(1/(\alpha \rho)) \) [Andersen, Chung, Lang, FOCS, 2006]

- We store vectors p and r using sparse sets
  - We can only afford to do work proportional to nodes and edges currently touched
  - We used \textit{unordered\_map} data structure in STL (Standard Template Library)
  - Guarantees \( O(1/(\alpha \rho)) \) work
L1-regularized Page-Rank

APPR is an approximation algorithm but what is it minimizing?

\[
\minimize \frac{1 - \alpha}{2} \|Bp\|^2_2 + \alpha \|H(1 - p)\|^2_2 + \alpha \|Zp\|^2_2 + \rho \alpha \|Dp\|_1
\]

where

- B: is the incidence matrix
- Z, H: are diagonal scaling matrices

Incidence matrix B

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-B</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A-C</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B-C</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-D</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D-E</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-H</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Shared memory
Running time: work depth model

Work depth model: J. Jaja. Introduction to parallel algorithms. Addison-Wesley Professional, 1992

Note that our results are not model dependent.

Model
• Work: number of operations required
• Depth: longest chain of sequential dependencies

Let P be the number of cores available.

By Brent’s theorem [1] an algorithm with work \( W \) and depth \( D \) has overall running time: \( \frac{W}{P} + D \).

In practice \( \frac{W}{P} \) dominates. Thus parallel efficient algorithms require the same work as its sequential version.

Parallel Approximate Personalized Page-Rank

While termination criterion is not met do
1. Choose **ALL (instead of any)** vertex u where \( r[u] < -\alpha p \deg[u] \)
2. \( p[u] = p[u] - r[u] \)
3. For all neighbours v of u: \( r[v] = r[v] + (1-\alpha)/(2\deg[u])r[u] \)
4. \( r[u] = (1-\alpha)r[u]/2 \)

- **Asymptotic work remains the same:** \( O(1/(\alpha p)) \).
- **Parallel randomized implementation:** work \( O(1/(\alpha p)) \) and depth \( O(\log(1/(\alpha p))) \).
  - Keep track of two **sparse** copies of p and r
  - Concurrent hash table for sparse sets ← important for \( O(1/(\alpha p)) \) work
  - Use atomic increment to deal with conflicts
  - Use of Ligra (Shun and Blelloch 2013) to process only “active” vertices and their edges
- **Same theoretical graph clustering guarantees**, Fountoulakis et al. 2016.
## Data

<table>
<thead>
<tr>
<th>Input graph</th>
<th>Num. vertices</th>
<th>Num. edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>soc-JL</td>
<td>4,847,571</td>
<td>42,851,237</td>
</tr>
<tr>
<td>cit-Patents</td>
<td>6,009,555</td>
<td>16,518,947</td>
</tr>
<tr>
<td>com-LJ</td>
<td>4,036,538</td>
<td>34,681,189</td>
</tr>
<tr>
<td>com-Orkut</td>
<td>3,072,627</td>
<td>117,185,083</td>
</tr>
<tr>
<td>Twitter</td>
<td>41,652,231</td>
<td>1,202,513,046</td>
</tr>
<tr>
<td>Friendster</td>
<td>124,836,180</td>
<td>1,806,607,135</td>
</tr>
<tr>
<td>Yahoo</td>
<td>1,413,511,391</td>
<td>6,434,561,035</td>
</tr>
</tbody>
</table>
Performance

- Slightly more work for the parallel version
- Number of iterations is significantly less
Performance

- 3-16x speed up
- Speedup is limited by small active set in some iterations and memory effects
Network community profile plots

- For Friendster, 124M nodes, 1.8B edges
- For Yahoo, 1.4B nodes, 6.4B edges

$\mathcal{O}(10^5)$ approximate PPR problems were solved in parallel for each plot,

Agrees with conclusions of [Leskovec et al. 2008], i.e., good clusters tend to be small.
Rounding: sweep cut

- Round returned vector $p$ of approximate PPR
  - **1st step ($O(1/(\alpha \rho) \log(1/(\alpha \rho)))$ work):** Sort vertices by non-increasing value of non-zero $p[u]/d[u]$
  - **2nd step ($O(1/(\alpha \rho))$ work):** Look at all prefixes of sorted order and return the cluster with minimum conductance,

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Conductance</th>
</tr>
</thead>
<tbody>
<tr>
<td>${A}$</td>
<td>1</td>
</tr>
<tr>
<td>${A,B}$</td>
<td>1/2</td>
</tr>
<tr>
<td>${A,B,C}$</td>
<td>1/7</td>
</tr>
<tr>
<td>${A,B,C,D}$</td>
<td>3/11</td>
</tr>
</tbody>
</table>
Parallel sweep cut

• **1st step:** Sort vertices by non-increasing value of non-zero \( p[u]/d[u] \).
  - Use parallel sorting algorithm, \( O(\frac{1}{\alpha \rho} \log(\frac{1}{\alpha \rho})) \) work and \( O(\log(\frac{1}{\alpha \rho})) \) depth.

• **2nd step:** Look at all prefixes of sorted order and return the cluster with minimum conductance.
  - Naive implementation: for each sorted prefix compute conductance, \( O(\frac{1}{(\alpha \rho)^2}) \).
  - *We design a parallel algorithm based on integer sorting and prefix sums that takes \( O(\frac{1}{(\alpha \rho)}) \) time.*
  - The algorithm computes the conductance of ALL sets with a single pass over the nodes and the edges.
Parallel sweep cut: 2nd step

Incidence matrix B

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-B</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A-C</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B-C</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-D</td>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D-E</td>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D-F</td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D-G</td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-H</td>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sorted vertices: \{A,B,C,D\}

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Sum cols B</th>
<th>Volume</th>
<th>Conductance</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A}</td>
<td>2</td>
<td>2</td>
<td>2/2=1</td>
</tr>
<tr>
<td>{A,B}</td>
<td>2</td>
<td>4</td>
<td>2/4=1/2</td>
</tr>
<tr>
<td>{A,B,C}</td>
<td>1</td>
<td>7</td>
<td>1/7</td>
</tr>
<tr>
<td>{A,B,C,D}</td>
<td>3</td>
<td>11</td>
<td>3/11</td>
</tr>
</tbody>
</table>

- Sort vertices
  - work: $O(1/(\alpha \rho) \log(1/(\alpha \rho)))$, depth: $O(\log(1/(\alpha \rho)))$
- Represent matrix B with a sparse set using vertex identifiers and the order of vertices
  - work: $O(1/(\alpha \rho))$, depth: $O(\log(1/(\alpha \rho)))$
- Use prefix sums to sum elements of the columns
  - work: $O(1/(\alpha \rho))$, depth: $O(\log(1/(\alpha \rho)))$
Parallel sweep cut: performance

The graph shows the performance of parallel sweep (solid line) and sequential sweep (dashed line) as a function of the number of cores. The x-axis represents the number of cores, ranging from 1 to 40, and the y-axis represents the running time in seconds, with a logarithmic scale.

The graph indicates that the running time decreases as the number of cores increases, demonstrating the benefits of parallel processing. The parallel sweep method is consistently faster than the sequential sweep method across all core counts.
Summary

• We parallelise 4 spectral algorithms for local graph clustering.

• The proposed algorithms are work efficient, i.e., same worst-case work.

• We parallelise the rounding procedure to obtain the clusters.

• Useful in interactive setting where one has to experiment with parameters.

• 3-15x faster than sequential version

• Parallelisation allowed us to solve problems of billions of nodes and edges.
Further work: distributed block coordinate descent

• Generalization to $l_2$-regularized least-squares and kernel learning:

  A. Devarakonda, KF, J. Demmel, M. Mahoney: Avoiding communication in primal and dual block coordinate descent methods (work in progress ≤ month)

• Given a positive integer $k$
  - We reduce latency for BCD by a factor of $k$
  - at the expense of a factor of $k$ more work and number of words.
Thank you!