Local Graph Sparsification for Scalable Clustering

Venu Satuluri, Srinivasan Parthasarathy and Yiye Ruan
Ohio State University

Presented By: Ahmed Elbagoury
Outline

• Motivation
• Related Work
• Graph Sparsification
  • Global Sparsification
  • Local Sparsification
  • Minwise Hashing
• Experimental Evaluation
• Conclusion
• Discussion
Motivation

• Man real problems can be modeled as graphs
• Analysing these graphs using clustering

Genetic Interaction Networks

Economic Networks

Social Network community discovering
Motivation

• Many graph clustering algorithms have been proposed

• Scalability of these algorithms is a challenge

• Large graphs are common in WWW domain
  • Facebook and Twitter

The goal is:

Develop a preprocessing step
  • To speedup graph clustering algorithms
  • Without loosing quality of the clusters

Not developing a clustering algorithm
Outline

• Motivation

• Related Work
  • Graph Sparsification
    • Global Sparsification
    • Local Sparsification
    • Minwise Hashing

• Experimental Evaluation

• Conclusion

• Discussion
Related Work

• Preserving edge cut
  • Random Sampling
  • Edge Connectivity
  • Resistance of an edge

• Edge filtering
  • Identifies the most interesting edges without keeping cluster structure
  • Inappropriate for unweighted networks

• Graph Sampling
  • Selects subset of edges and nodes
  • How to cluster missing nodes?!
Outline

• Motivation

• Related Work

• Graph Sparsification
  • Global Sparsification
  • Local Sparsification
  • Minwise Hashing

• Experimental Evaluation

• Conclusion

• Discussion
Graph Sparsification

• The objective is scaling up clustering algorithms
• Reduce the size of the graph
• Sparsify the graph: Filter only some edges and retain all the nodes

FIGURES ARE FROM "LOCAL GRAPH SPARSIFICATION FOR SCALABLE CLUSTERING." PROCEEDINGS OF THE 2011 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA. ACM, 2011
Sparsification Method

- Clustering sparsified graph should be much faster than the original graph
- Retain good clustering quality compared to the clustering over the original graph
- Fast sparsification method
Sparsification Method

Retain the intra-cluster edges compared to inter-cluster edges
Intra and inter edges

One way to differentiate between intra and inter edges is

- Edge betweenness centrality
  - Number of shortest paths between any two vertices that pass through the edge \((i, j)\)
  - The higher the betweenness the higher the edge is an inter-cluster edge
  - Expensive to compute the betweenness
Similarity-based Sparsification Heuristic

• An edge \( (i, j) \) is likely to lie within a cluster
  • If the vertices \( i \) and \( j \) have adjacency list with high overlap

• Jaccard measure quantifies the overlap between adjacency lists

\[
Sim(i, j) = \frac{|\text{Adj}(i) \cap \text{Adj}(j)|}{|\text{Adj}(i) \cup \text{Adj}(j)|}
\]
Global Sparsification (G-Spar)

• Each edge \((i, j)\), will be assigned a similarity \(\text{sim}(i, j)\)

• Return the graph with the top \(s\%\) of all the edges in the graph
  • \(s\) is a sparsification parameter

---

**Algorithm 1 Global Sparsification Algorithm**

**Input:** Graph \(G = (V, E)\), Sparsification ratio \(s\)

\[G_{\text{sparse}} \leftarrow 0\]

for each edge \(e=(i,j)\) in \(E\) do
  \[e.\text{sim} = \text{Sim}(i,j)\] according to Eqn 1
end for

Sort all edges in \(E\) by \(e.\text{sim}\)
Add the top \(s\%\) edges to \(G_{\text{sparse}}\)

return \(G_{\text{sparse}}\)
Global Sparsification (G-Spar)

Using one global threshold is not suitable for multiple density clusters

Original graph

Globally sparsified graph

FIGURES ARE FROM “LOCAL GRAPH SPARSIFICATION FOR SCALABLE CLUSTERING.” PROCEEDINGS OF THE 2011 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA. ACM, 2011
Local Sparsification (L-Spar)

• Avoid using a global threshold
• For each node $i$ (with degree $d_i$), compute the similarity of each edge
• Pick the top $f(d_i) = d_i^e$
  • Ensure that for each node, at least one of its incident edges is picked
• Adapt to different densities
Local Sparsification (L-Spar)

Original graph

Locally sparsified graph
Time Complexity

• For node $i$ (with degree $d_i$) and node $j$ (with degree $d_j$)
  • Number of operations to intersect their adjacency lists is proportional to $d_i + d_j$

• A node of degree $d_i$ requires $d_i$ intersections
  • The total number of operations is proportional to $\sum_i d_i^2$

• The total number of operations is proportional to $n \cdot d_{avg}^2$
  • $d_{avg}$ is the average degree of the graph

Prohibitively large for most graphs
Minwise Hashing

• Min-wise hashing is as a technique for estimating the jaccard coefficient between sets $A$ and $B$
• Let $h$ be a hash function that maps the members of a set $S$ of size $n$ to the range $\{0, \ldots, n - 1\}$

<table>
<thead>
<tr>
<th>10</th>
<th>12</th>
<th>4</th>
<th>7</th>
<th>5</th>
<th>3</th>
</tr>
</thead>
</table>

$h$

| 3 | 5 | 0 | 1 | 2 | 4 |

• Define $h_{min}(S)$ to be $\min_{x \in S} h(x)$
  • The member $x$ of $S$ with the minimum value of $h(x)$
  • $h_{min}(S) = 4$
Minwise Hashing

- $h_{\text{min}}(A) = h_{\text{min}}(B)$ when $h_{\text{min}}(A)$ is in B
- Which means $h_{\text{min}}(A \cup B)$ lies in the intersection $A \cap B$
- This happens with probability $\frac{|\text{Adj}(i) \cap \text{Adj}(j)|}{|\text{Adj}(i) \cup \text{Adj}(j)|}$ Which is equal to $\text{sim}(i, j)$
Minwise Hashing

• A simple estimator for $\text{sim}(A, B)$ is $I[h_{\text{min}}(A) = h_{\text{min}}(B)]$
  • $I[x] = 1$ if $x$ is true and 0 otherwise.

• Hashing can be considered as a random permutation $\pi$ on $n$ numbers rather than a hash function

\[
h_{\text{min}}(A) = 7
\]

\[
\text{first}(\pi(A)) = 7
\]

which is the first element in $\pi(A)$
Minwise Hashing

This can be considered as a random permutation $\pi$ on $n$ numbers rather than a hash function.

$\pi(A)$

\[
\begin{array}{cccccc}
7 & 4 & 3 & 5 & 12 & 10 \\
\end{array}
\]

first($\pi(A)$) = 7

$\pi(B)$

\[
\begin{array}{cccccc}
8 & 3 & 4 & 12 & 11 & 8 \\
\end{array}
\]

first($\pi(B)$) = 8
Minwise Hashing

• A simple estimator for $\text{sim}(A, B)$ is $I[\text{first}(\pi(A)) = \text{first}(\pi(B))]$

• This estimate has a high variance
  • It is always zero or one
Minwise Hashing

To reduce variance, use $k$ independent permutations $\pi_i, i = 1:k$

<table>
<thead>
<tr>
<th>A</th>
<th>10</th>
<th>12</th>
<th>4</th>
<th>7</th>
<th>5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_1(A)$</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>$\pi_2(A)$</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>10</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>$\pi_3(A)$</td>
<td>4</td>
<td>10</td>
<td>7</td>
<td>3</td>
<td>12</td>
<td>5</td>
</tr>
</tbody>
</table>

$\text{first}(\pi_1(A)) = 7$

$\text{first}(\pi_2(A)) = 5$

$\text{first}(\pi_3(A)) = 4$
Minwise Hashing

- To reduce variance, use $k$ independent permutations $\pi_i, i = 1: k$

$$\text{sim}(A, B) = \frac{1}{k} \sum_{i=1}^{k} I[\text{first}(\pi_i(A)) = \text{first}(\pi_i(B))]$$
Generating Permutations

• Linear permutations

\[ \pi(i) = (a \times i + b) \mod P \]

• \( P \) is large prime number

• \( a \) is an integer drawn uniformly at random from \([1, P - 1]\)

• \( b \) is an integer drawn uniformly at random from \([0, P - 1]\)

• Multiple linear permutations can be generated by generating random pairs of \((a, b, P)\)
Minwise Hashing

• Generate $k$ linear permutations, by generating $k$ triplets $(a, b, P)$

• Compute a length $k$ signature for each node by minhashing its adjacency list $k$ times

• Fill up a hashtable of size $n \times k$

• $\forall (i, j) \in G$
  • Compare the signatures of $i$ and $j$
  • Count the number of matching minhashes

• Sort the edges incident to a node $i$ by the number of matches of each edge

• Include the top $d_i^e$ inclusion in the sparsified graph
Outline

• Motivation
• Related Work
• Graph Sparsification
  • Global Sparsification
  • Local Sparsification
  • Minwise Hashing
• Experimental Evaluation
• Conclusion
• Discussion
Experimental Results

• Seven real-world datasets
  • Three biological datasets: Yeast-BioGrid (BioGrid), Yeast-DIP (DIP), Human-PPI (Human)
  • Wikipedia: an undirected version of the graph of hyperlinks between Wikipedia articles
  • Three social networks datasets: Orkut, Twitter, Flickr tags

• Baselines
  • L-Sparse
  • G-Sparse
  • Random Edge Sampling
  • ForestFire

• All the experiments were performed on a PC with 16 GB RAM
Experimental Results

• Average F-score (higher is better) when ground truth is available

• Average Conductance (lower is better) when ground truth is not available
  \[
  \phi(S) = \frac{|(i,j) \text{ s.t. } i \in S, j \in \overline{S}|}{\min(|(i,j) \text{ s.t. } i \in S, j \in V|, |(i,j) \text{ s.t. } i \in \overline{S}, j \in V|)}
  \]

• Four state-of-the-art algorithms for clustering
  • Metis
  • Metis+MQI
  • MLR-MCL
  • Graclus
### Experimental Results

#### Clustering Algorithm: Metis

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Original</th>
<th>Sp. Ratio</th>
<th>RandomEdge</th>
<th>G-Spar</th>
<th>ForestFire</th>
<th>L-Spar</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F-score</td>
<td>Time (s)</td>
<td>F-score</td>
<td>Spdup</td>
<td>F-score</td>
<td>Spdup</td>
</tr>
<tr>
<td>BioGrid</td>
<td>17.78</td>
<td>3.02</td>
<td>15.98</td>
<td>11x</td>
<td>15.15</td>
<td>30x</td>
</tr>
<tr>
<td>DIP</td>
<td>20.04</td>
<td>0.11</td>
<td>17.58</td>
<td>2x</td>
<td>19.38</td>
<td>2x</td>
</tr>
<tr>
<td>Human</td>
<td>8.96</td>
<td>0.59</td>
<td>7.75</td>
<td>4x</td>
<td>8.64</td>
<td>4x</td>
</tr>
<tr>
<td>Wiki</td>
<td>12.34</td>
<td>7485</td>
<td>9.11</td>
<td>8x</td>
<td>9.38</td>
<td>104x</td>
</tr>
<tr>
<td>Orkut</td>
<td>0.85</td>
<td>(0.1)</td>
<td>0.82</td>
<td>(0.4)</td>
<td>0.76</td>
<td>(0.4)</td>
</tr>
<tr>
<td>Flickr</td>
<td>0.87</td>
<td>(2.5)</td>
<td>0.91</td>
<td>(0.1)</td>
<td>0.77</td>
<td>(0.1)</td>
</tr>
<tr>
<td>Twitter</td>
<td>0.95</td>
<td>(0.1)</td>
<td>1.0</td>
<td>(0.4)</td>
<td>0.97</td>
<td>(0.0)</td>
</tr>
</tbody>
</table>

#### Clustering Algorithm: MLR-MCL

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Original</th>
<th>Sp. Ratio</th>
<th>RandomEdge</th>
<th>G-Spar</th>
<th>ForestFire</th>
<th>L-Spar</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F-score</td>
<td>Time (s)</td>
<td>F-score</td>
<td>Spdup</td>
<td>F-score</td>
<td>Spdup</td>
</tr>
<tr>
<td>BioGrid</td>
<td>23.95</td>
<td>8.44</td>
<td>20.28</td>
<td>6x</td>
<td>18.29</td>
<td>38x</td>
</tr>
<tr>
<td>DIP</td>
<td>24.85</td>
<td>0.28</td>
<td>20.57</td>
<td>3x</td>
<td>22.45</td>
<td>3x</td>
</tr>
<tr>
<td>Human</td>
<td>10.55</td>
<td>1.68</td>
<td>8.81</td>
<td>4x</td>
<td>9.21</td>
<td>6x</td>
</tr>
<tr>
<td>Wiki</td>
<td>20.22</td>
<td>7898</td>
<td>8.74</td>
<td>19x</td>
<td>9.3</td>
<td>92x</td>
</tr>
<tr>
<td>Orkut</td>
<td>0.78</td>
<td>(6.4)</td>
<td>0.85</td>
<td>(1.2)</td>
<td>0.91</td>
<td>(10.1)</td>
</tr>
<tr>
<td>Flickr</td>
<td>0.71</td>
<td>(0.6)</td>
<td>0.83</td>
<td>(2.2)</td>
<td>0.72</td>
<td>(3.0)</td>
</tr>
<tr>
<td>Twitter</td>
<td>0.90</td>
<td>(5.6)</td>
<td>0.99</td>
<td>(0.6)</td>
<td>0.89</td>
<td>(1.0)</td>
</tr>
</tbody>
</table>
## Experimental Results

### Clustering Algorithm: Metis+MQI

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Original</th>
<th>Sp. Ratio</th>
<th>RandomEdge</th>
<th>G-Sparified</th>
<th>ForestFIRE</th>
<th>L-Sparified</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F-score</td>
<td>Time (s)</td>
<td>F-score</td>
<td>Spdup</td>
<td>F-score</td>
<td>Spdup</td>
</tr>
<tr>
<td>BioGrid</td>
<td>23.16</td>
<td>4.0</td>
<td>0.17</td>
<td>19.76</td>
<td>11x</td>
<td>17.74</td>
</tr>
<tr>
<td>DIP</td>
<td>23.09</td>
<td>0.32</td>
<td>0.53</td>
<td>19.55</td>
<td>1x</td>
<td>21.18</td>
</tr>
<tr>
<td>Human</td>
<td>10.17</td>
<td>1.16</td>
<td>0.39</td>
<td>8.42</td>
<td>1x</td>
<td>9.1</td>
</tr>
<tr>
<td>Wiki</td>
<td>19.21</td>
<td>15511</td>
<td>0.15</td>
<td>14.97</td>
<td>5x</td>
<td>9.98</td>
</tr>
<tr>
<td>Orkut</td>
<td>0.756 (1.2)</td>
<td>19799</td>
<td>0.17</td>
<td>0.86 (0.5)</td>
<td>2x</td>
<td>0.77 (0.2)</td>
</tr>
<tr>
<td>Flickr</td>
<td>0.55 (14.1)</td>
<td>72.35</td>
<td>0.2</td>
<td>0.68 (0.4)</td>
<td>3x</td>
<td>0.67 (0.3)</td>
</tr>
<tr>
<td>Twitter</td>
<td>0.86 (0.6)</td>
<td>1708</td>
<td>0.04</td>
<td>0.99 (0.6)</td>
<td>35x</td>
<td>0.97 (0.0)</td>
</tr>
</tbody>
</table>

### Clustering Algorithm: Graclus

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Original</th>
<th>Sp. Ratio</th>
<th>RandomEdge</th>
<th>G-Sparified</th>
<th>ForestFIRE</th>
<th>L-Sparified</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F-score</td>
<td>Time (s)</td>
<td>F-score</td>
<td>Spdup</td>
<td>F-score</td>
<td>Spdup</td>
</tr>
<tr>
<td>BioGrid</td>
<td>19.15</td>
<td>0.32</td>
<td>0.17</td>
<td>17.59</td>
<td>4x</td>
<td>16.56</td>
</tr>
<tr>
<td>DIP</td>
<td>21.77</td>
<td>0.19</td>
<td>0.53</td>
<td>18.27</td>
<td>2x</td>
<td>21.27</td>
</tr>
<tr>
<td>Human</td>
<td>0.53</td>
<td>0.81</td>
<td>0.39</td>
<td>8.03</td>
<td>2x</td>
<td>8.75</td>
</tr>
<tr>
<td>Flickr</td>
<td>0.66 (1.3)</td>
<td>1.35</td>
<td>0.2</td>
<td>0.72 (0.1)</td>
<td>2x</td>
<td>0.66 (0.1)</td>
</tr>
<tr>
<td>Twitter</td>
<td>0.90 (2.4)</td>
<td>1518</td>
<td>0.04</td>
<td>1.0 (0.7)</td>
<td>138x</td>
<td>0.97 (0.0)</td>
</tr>
</tbody>
</table>
Speedup and F-score as $e$ changes

(a) F-score as $e$ changes; Metis
(b) Speedup as $e$ changes; Metis
Speedup and F-score as $k$ changes

(e) F-score as $k$ changes; Metis

(f) Speedup as $k$ changes; Metis
Conclusion

• Scaling clustering algorithm is an important task

• Introduced an efficient and effective localized method to sparsify a graph
  • Retain only a fraction of the original number of edges
  • Handle multiple density clusters

• The proposed sparsification speeds up graph clustering algorithms without sacrificing quality
Thanks
Questions?
Discussion

Implementing the sparsification method on multiple machines
  ◦ How to partition the data?
  ◦ Some framework doesn’t support Graph mutations: GraphLab and PowerGraph

How to handle streams (incremental clustering)?
  ◦ Will we need to re-cluster
  ◦ Will the density change, do we need to change $e$

Can we use other measures that takes in account edge weight and edge direction?

How to handle outliers more efficiently?