A Scalable Distributed Graph Partitioner

Daniel Margo and Margo Seltzer
Harvard University
Graph Partitioning

Balance **partitions** → while minimizing some **cut cost**

Models graph computing costs: network traffic, cache efficiency...

*Hepatitis C coauthorship* by Andy Lamb
Motivation

From “Think Like a Vertex” to “Think Like a Graph” by Tian et al. VLDB’14

<table>
<thead>
<tr>
<th>dataset</th>
<th>execution time (sec)</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>hash partitioned (HP)</td>
<td>VM</td>
<td>HM</td>
<td>GM</td>
<td>VM</td>
<td>HM</td>
<td>GM</td>
</tr>
<tr>
<td>uk-2002</td>
<td>441</td>
<td>438</td>
<td>272</td>
<td></td>
<td>532</td>
<td>89</td>
<td>19</td>
</tr>
<tr>
<td>uk-2005</td>
<td>1,366</td>
<td>1,354</td>
<td>723</td>
<td></td>
<td>1,700</td>
<td>230</td>
<td>90</td>
</tr>
<tr>
<td>webbase-2001</td>
<td>4,491</td>
<td>4,405</td>
<td>1,427</td>
<td></td>
<td>3,599</td>
<td>1,565</td>
<td>57</td>
</tr>
<tr>
<td>clueweb50m</td>
<td>1,875</td>
<td>2,103</td>
<td>1,163</td>
<td></td>
<td>1,072</td>
<td>250</td>
<td>103</td>
</tr>
</tbody>
</table>

25x speed up! If the partition model is right... ...except these partitions took 1000-6000s
Partition Models

- Vertex cuts model **message batches** (e.g. GraphX)
- Edge partitions favor “power law” graphs (e.g. social)
Our Partitioner: Sheep

Sheep is...

- *Extremely* fast
- A simple, distributed map-reduce
  - Data scales to any number of machines...
  - ...or out-of-memory via working sets
- Finds low cut-cost partitions
- **Limitation**: Only for edge partitions of “power law” graphs
The Sheep Algorithm

1) **Sort** the vertices of the graph  
(indistributed)

2) **Reduce** the graph to a tree  
(indistributed)

3) **Partition** that tree  
(cheap)

4) Map partitions back to the graph  
(cheap)

We'll go through these steps in reverse
Elimination Trees

If (X,Y) in G, then X is below Y in T or vice-versa.
A vertex partitioning of $T$ gives an edge partitioning of $G$.

$\text{partition}(\{X,Y\})$ in $G = \text{partition}(X)$, $X$ below $Y$ in $T$. 
An \textbf{edge cut} in $T$ gives a \textbf{vertex cut} in $G$ that is upper-bounded by the \textbf{tree-height}.
We construct trees via a **persistent union-find**. For each vertex and its **preceding edges**...
For each preceding edge, if the preceding vertex has no parent, adopt it.
If the preceding vertex has a parent, **union-find**

The **tree-height** depends on the **vertex order**
Distribution

To distribute, make trees from arbitrary subgraphs of G, then merge them

If $G = G_1 \cup G_2$, then $T(G) = T(T(G_1) \cup T(G_2))$

Trees must be in the same vertex order
A low-depth vertex order is equivalent to a successful attack order on a “power law” graph:

**Attack on a Physics coauthor network from**
*Attack Robustness and Centrality of Complex Networks*, Iyer et al. **PloS ONE**
Intuition

Graph and tree of the Physics coauthor network

“Power law” graphs are topologically similar to trees.
Evaluation

- Graphs up to 3.7B edges (uk-2007-05)

- Partition time and cost comparisons with: METIS, FENNEL (many variants), PowerGraph

- Compared to published results by Bourse et al. (Balanced Graph Edge Partition, KDD'14)
Sheep's runtime doesn't vary with the partition count.
Runtime: 1.5 billion $|E|$.

Sheep < 3 min, Fennel 22+ min, Metis ~hours.
Runtime: $3.7 \text{ billion } |E|$

More nodes → more memory bandwidth → speed!
2.5x previous graph → 2x nodes → from 163 to 202s
Sheep's quality is good and in some cases, great
Partition Quality (2)

Sheet's quality is good and in some cases, great

**Limitation:** But only for edge-partitioned vertex cuts
Future Work

- Quality improves with “better” analytic rankings
- Use analysis to find partitions, and vice-versa
Conclusion

• Sheep is an extremely fast and scalable method to partition very large graphs.

• Sheep's partition quality is good... **Limitation:** ...given a certain partition model.

• Promising future work to further improve Sheep's quality and robustness + novel analysis

Thanks for listening!
More Intuition

total order:
1 → 2 → 3 → 4 → 5 → 6 → 7

partial order (tree):
1 → 2 → 3 → 5 → 6 → 7

Sheep is a distributed sort into a partial order. The PO retains some of the graph's topology.