ParViz: Visualizing Graph Partitioners

Hadi Sinaee

CPSC 547, December 2021
University of British Columbia
Motivation

Persons

Friends

Joe

Kevin

Andy

Alex
Graph Partitioners

- To create a new partitioner we need to understand the previous ones work
- Different metrics for graph partitioners:
  - Load Balance (LB)
  - Edge-Cuts (EC)
  - Duplication Factor (DF)
- These metrics are measured at the end of the partitioning process
  - It'd help to see how they were evolving
ParViz: Visualizing Graph Partitioners

“Build a viz tool to help algorithm designer understand the partitioning process of a partitioner.”
Task Abstraction

1. How do partitioning metrics evolve during the partitioning process?
2. How does the size of partitions change over time?
3. What does the final partitioned graph look like?
1. The input graph is the Yahoo! Messenger dataset which is a public dataset
   a. An undirected graph
   b. ~1.9M nodes and 4M edges
   c. Nodes are users and the edges show at least one the nodes is a contact of another one

2. Number of partitions is three

3. The graph was partitioned by a vertex partitioner named Fennel
   a. We assign each node and all of its edges into one partition

4. Dataset type is Table with Items and Attributes
   a. Input format is CSV
## What-How-Why

<table>
<thead>
<tr>
<th>What: Data</th>
<th>Table: DF, LB, EC quantitative attributes</th>
</tr>
</thead>
</table>
| What: Derived | ● Partitioner Steps: Ordered key attribute  
● Normalized Values of DF, LB, EC |
| How: Encode | Express DF, LB, EC horizontally; using a heat map with different hues and changing saturation |
| How: Reduce | Sampled at every 1000 point |
| Why: Task | Overview of changes in metrics |
| Scale | Items: ~200K |

![Metrics Heatmap](image-url)
<table>
<thead>
<tr>
<th>What: Data</th>
<th>Table; DF quantitative attributes</th>
</tr>
</thead>
</table>
| What: Derived      | ● Partitioner Step: Ordered key attributed  
|                    | ● Normalized DF                   |
| How: Encode        | Express DF horizontally; the x-axis is the steps and y-axis shows the DF value |
| How: Reduce        | Sampled every 1000 steps          |
| Why: Task          | Overview of changes in DF         |
| Scale              | Items: ~200K                      |
### What-How-Why

| What: Data | Table; **NodeId** key attribute **PartNum** categorical key attribute |
| What: Derived | ● Matrix of the size: 200K * 3 ● Each cell shows the number of nodes at a specific step in a specific partition |
| How: Encode | Facet; Superimpose the size of partitions for at each step |
| How: Reduce | Sampled every 1000 steps |
| Why: Task | Overview of changes in partitions size |
| Scale | Items: ~200K * 3 |
What: Data

Table:
- **Part1**, **Part2** categorical key attribute
- **ec** quantitative attribute
- **nedge_part** quantitative attribute
- **nedge** quantitative attribute

What: Derived

- Matrix of the size: 3 * 3
- Each cell shows the normalized ec between two corresponding partitions

How: Encode

Express the normalized ec value using a heat map

Why: Task

Overview of ec in partitions size

Scale

Items: 3 * 3
Limitations/Critiques

- **Limitations**
  - Limited to vertex partitioners
  - Doesn’t scale for large graphs with more than 1M nodes after sampling
  - Doesn’t scale for large number of partitions, e.g. 256.
  - Lack of interactivity
  - Cannot compare two different algorithms against each other

- **Critiques**
  - Using RGB for different categories
  - D3 was not the right choice for the mentioned tasks
  - Some tasks are done manually, such changing the file
  - Lack of proper legends
  - Too many idioms