Evaluating Graph Analysis Algorithms on Evolving Graphs Using GraphChi

Will Sewell
What Are Evolving Graphs?

- Also known as “iterative” or “dynamic”
- Where processing must be performed on graphs whose edges are constantly updating
- Algorithms perform incremental updates rather than re-computing values for the entire graph in batch
Motivation

• Why compute graph properties (PageRank, etc.) incrementally rather than statically?

• Performance
  – Most of the graph does not change, so properties will be the same
    • Thus wasteful

• Timely updates
  – Graph updates visible rapidly
Approaches

- Still a relatively new area, with not much work
- Kineograph
- Naiad
- GraphChi
Why GraphChi?

- Interesting new algorithm
- Impressive Performance
- However paper seemed to present the evolving graphs as an afterthought
  - Therefore an interesting area for further work
The Dataset

- Amazon products
- Edges are “similar” products linked to or from product detail pages
- 542,684 nodes; 1,231,398 edges
- The evolving property can be simulated by a script that incrementally builds up a new graph from this existing one
Test Algorithms

• GraphChi has many static graph processing algorithms that Amazon would likely want to compute on products
  – PageRank
  – Community Detection
  – Connected Components

• Plan to implement my own
  – Betweenness Centrality
Test Machine

- My Laptop!
- Exactly what GraphChi is targeted at
Planned Tests

- One test to measure the maximum number of streaming edges per second (e/s) the algorithm can handle
  - GraphChi paper does this, but only with a single algorithm
  - Can be plotted as a line with nodes e/s against iteration time
- Can control for rate of update as well as number of edges in each update
Planned Tests

- Example from GraphChi Paper (PageRank)
Planned Tests

• For the optimal edges e/s stream, I will measure the time taken to ingest the entire graph, as opposed to running it statically at varying intervals.
  
  – For this I can plot the point at which the evolving graph method overtakes the static method

• Will combine relative performances of all algorithms into a single graph for easier comparison
Expectations

• Some algorithms will perform well on a streaming graph, others will be extremely slow if all combinations edges/nodes are used in calculating properties

• These slower algorithms are unlikely to ever beat static graph analysis
Possible Extensions

• Compare results with another system that supports evolving graphs (Naiad)
  – May be able to test on a cluster to play to Naiad's strengths

• Try other centrality measures:
  – Louvain method
  – k-clique percolation method

• Huge number of other algorithms I could test
Any questions/suggestions?