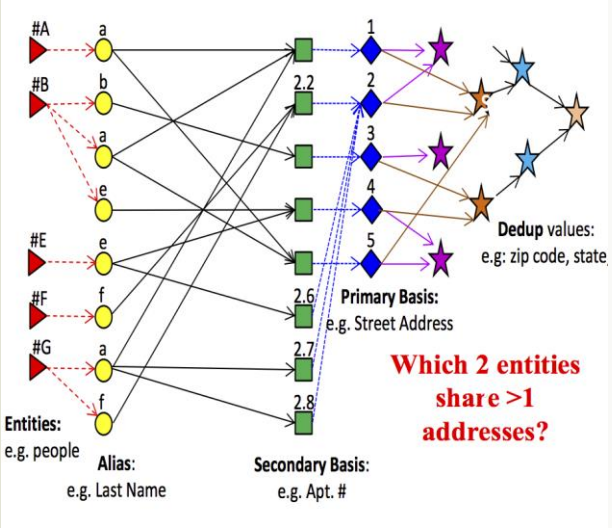
Jaccard Coefficients

Neil Butcher

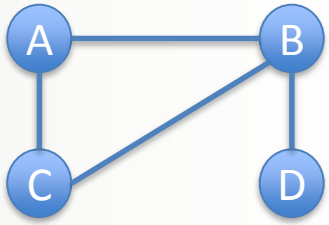
Jaccard

- Jaccard emulates real world problems
- Sparse access patterns
- Can be used for community detection



What is a Jaccard Coefficient?

- Similarity between neighborhoods of two nodes (V, U):
  - $\Gamma(u,v) = |N(V) \cup N(U)|$
  - $\aleph(u, v) = |N(V) \cap N(U)|$
  - $Jaccard(V,U) = \frac{\aleph(u,v)}{\Gamma(u, v)}$
  - $\aleph(A, C) = 1$
  - $\Gamma(A,C) = 3$
  - Jaccard(A, C) = 1/3



How to compute Jaccard

- Comes down to being able to compute intersection of neighborhoods (ɣ(u, v))
  - ɣ(u, v) = $|N(V) \cap N(U)|$
  - Γ(u,v) = |N(V)| + |N(U)| - ɣ(u, v)

Intersection Algorithm

- Intersect(U, V)
- For each vertex Y in Neighborhood(V):
  - If Y is in N(U)
    - IntersectCounter++
- Given neighborhoods that are sorted complexity is: O(M) – M is max of |N(U)| or |N(V)|
- Could sort first: O(Mlog(M))
- Otherwise O(M^2)

Jaccard – Compute all pairs

- Simple Soln: Compute Jaccards for all pairs $O(N^2*M)$
- 0 value Jaccards could be ignored if detected

Pseudocode

- For each vertex V
  - For each vertex U in N(V)
    - For each W in N(U)
      - If intersection(V, W) hasn't been computed, compute it
- Any pairs without a value have no shared neighborhood

Pseudocode Complexity

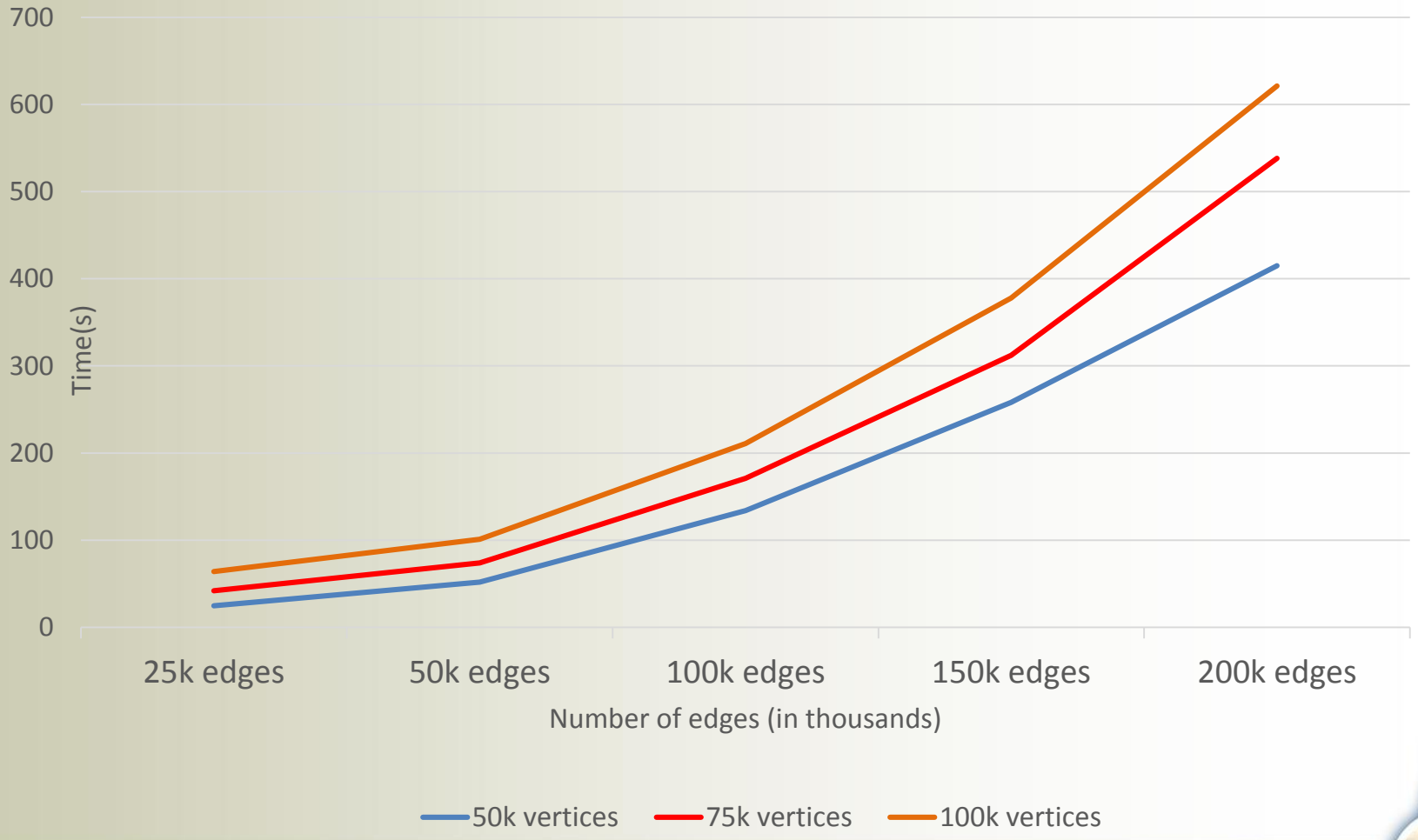
- For each vertex V **O(N)**
  - For each vertex U in N(V) **O(M)**
    - For each W in N(U) **O(M)**
      - If intersection(V, W) hasn't been computed, compute it **O(M)**

- **M is avg neighborhood size**
- **Overall complexity $O(N*M^3)$**

Input Graphs

- Use RMAT graphs
  - Generated using PaRMAT
- https://github.com/farkhor/PaRMAT
- Advantages:
  - Simple, easy to produce
  - Control input size/scaling
  - Evaluate Jaccard as HPC benchmark
- Disadvantages
  - Misses characteristics of real datasets

Results:



Future Work

- Adapt existing Triangle Counting algorithms to compute Jaccard (using GraphBLAS)
- Use Real World Graphs
- Adapt high levels of parallelism/cache oblivious techniques to utilize MLM
- Map Reduce Techniques?