

Graph Processing

Ashvin Goel

Electrical and Computer Engineering
University of Toronto

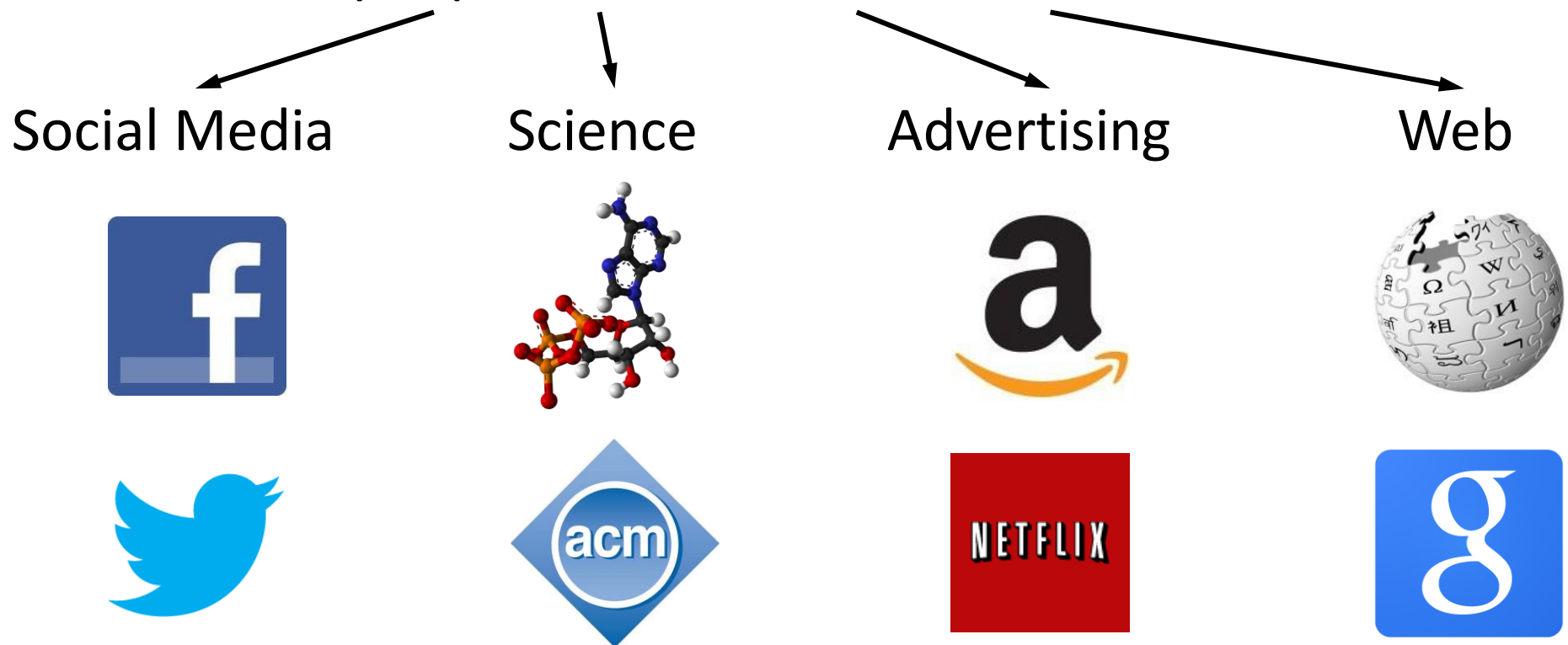
ECE1724

Authors:

Some slides adapted from Michael Freedman, Joseph Gonzalez

Graphs Encode Relationships

- Between people, ideas, interests, facts, ...



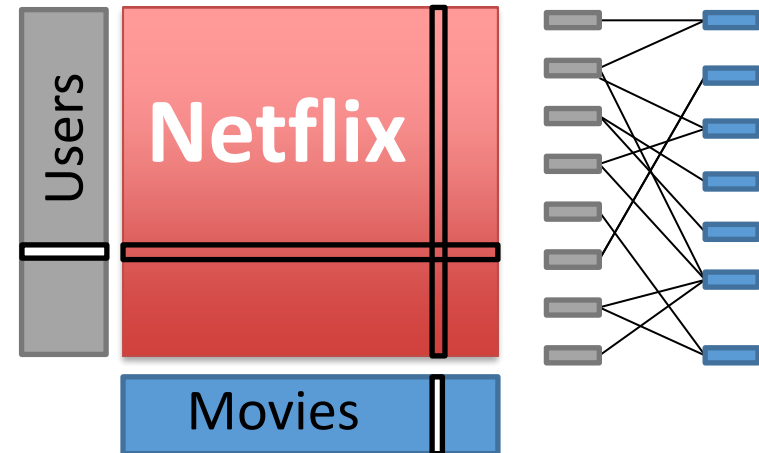
- Graphs are big: billions of vertices, edges, rich metadata

Graphs Are Everywhere

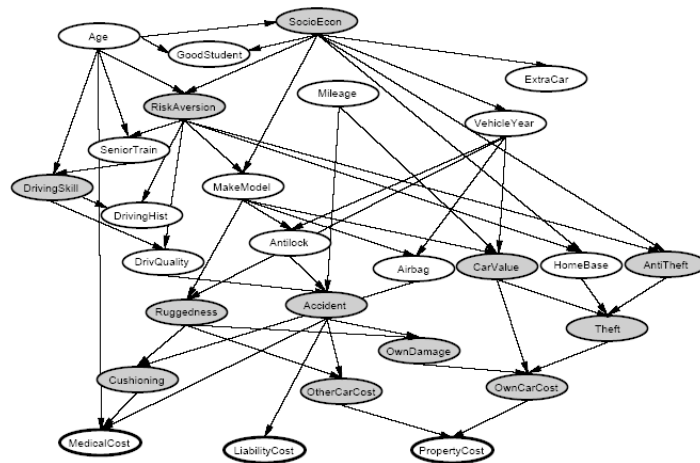
Social Network



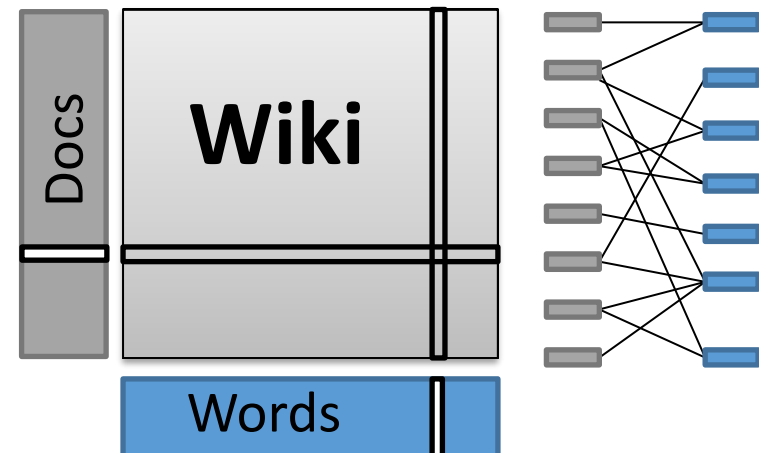
Collaborative Filtering



Probabilistic Analysis



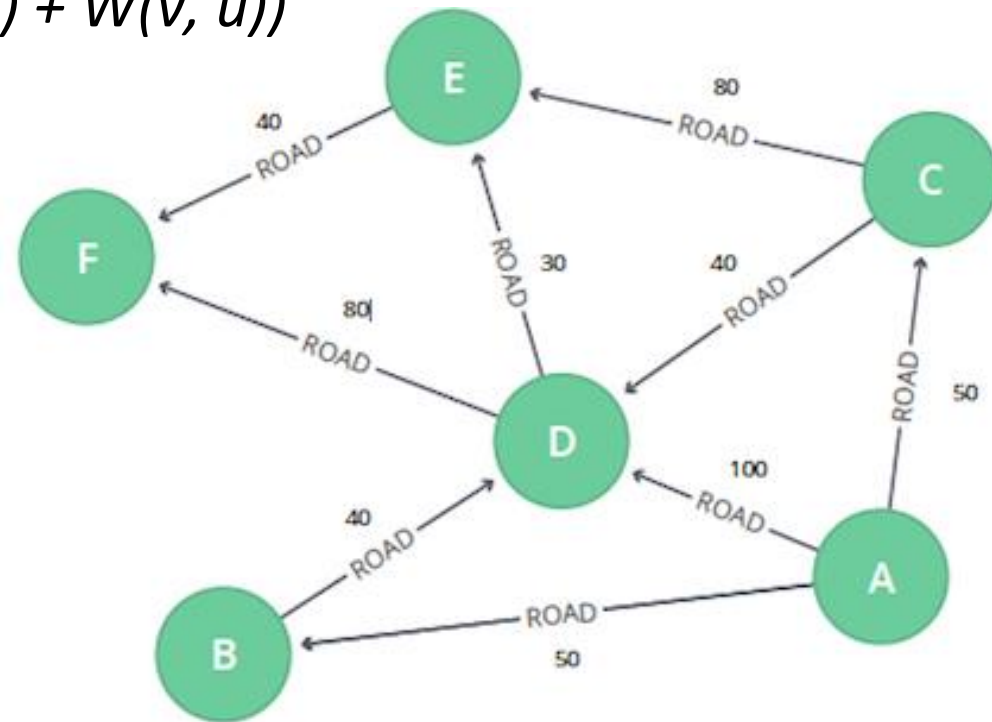
Text Analysis



Graph Algorithm Examples

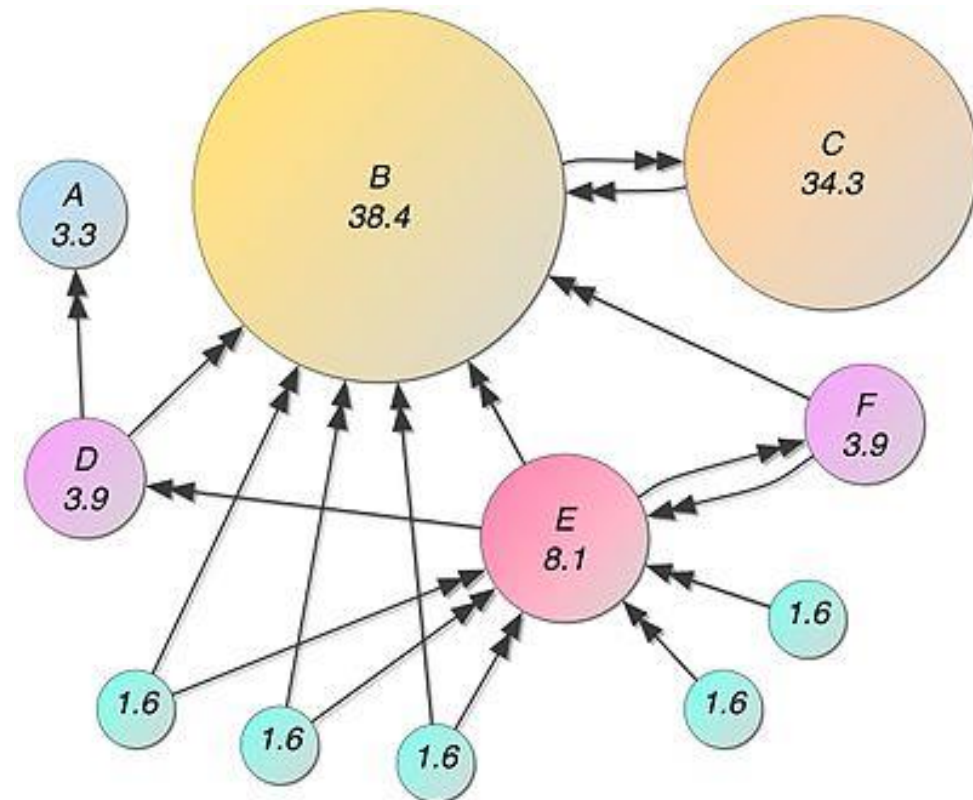
Single-Source Shortest Paths

- Find the shortest path from a node, say A, to all other nodes
- Iterative algorithm:
 - $Dist(A) = 0$
 - $Dist(u) = \min_{v \in Neighbor(u)} (Dist(v) + W(v, u))$
 - Iterate until convergence
- Parallelism:
 - Compute all $Dist(u)$ in parallel



PageRank Algorithm

- PageRank of u depends on PR of all pages v linking to u , divided by the number of links from each of these pages
- Recurrence Algorithm:
 - $PR[u] = \sum_{v \in \text{Links}(u)} (PR[v] / |\text{neighbors}(v)|)$
 - Iterate until convergence
- Parallelism:
 - Compute all $PR[u]$ in parallel



Label Propagation Algorithm

- Social Arithmetic:

50%: What I list on my profile

40%: Sue Ann Likes

+ 10%: Carlos Like

I Like: 60% Cameras, 40% Biking

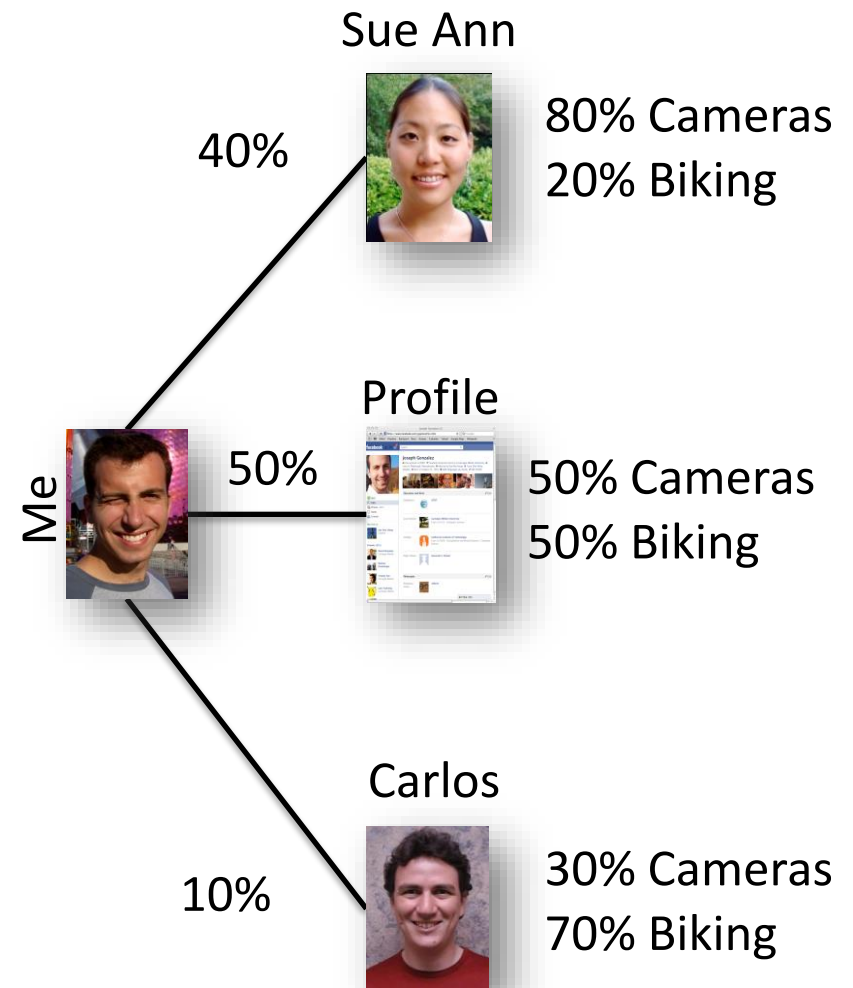
- Recurrence Algorithm:

- $Likes[i] = \sum_{j \in Friends(i)} W_{ij} \times Likes[j]$

- Iterate until convergence

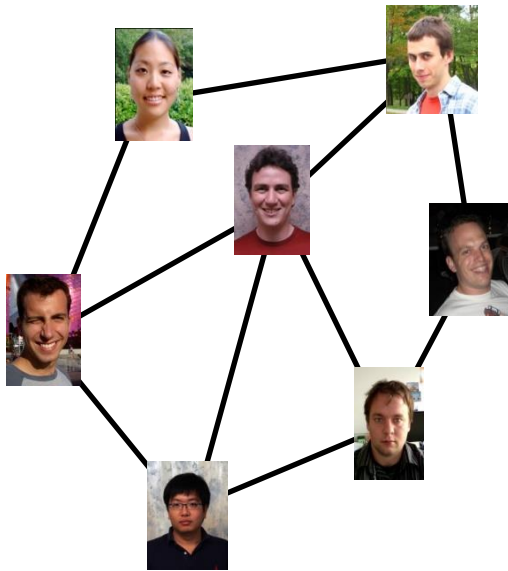
- Parallelism:

- Compute all $Likes[i]$ in parallel

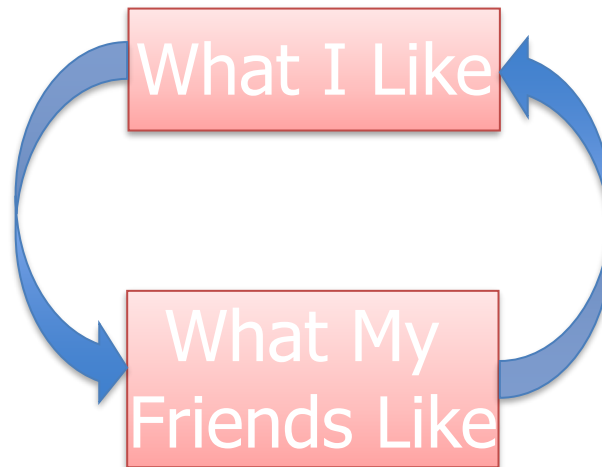


Properties of Graph Algorithms

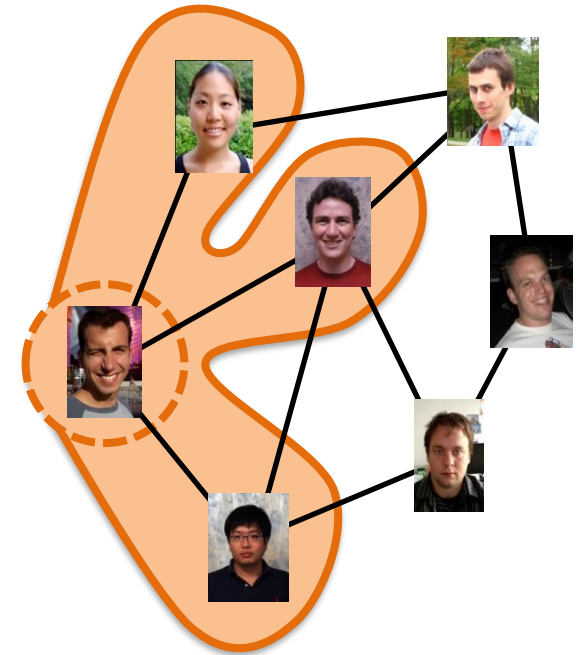
Graph
Dependencies



Iterative
Computation



Factored
Computation



Data Parallel versus Graph Parallel

- We have looked at map-reduce style data parallel frameworks (e.g., Hadoop, Spark)
- Why not use them for graph processing?
- They provide inefficient support for:
 - **Graph dependencies**
 - Users need to write complex transformations, e.g., reduce, to express dependencies between different vertices
 - **Iterative computation**
 - The transformations are stateless, so many data copies are required
 - Each iteration is a barrier, so no support for asynchronous operation
 - **Factored computation**
 - Even when subset of graph changes, require graph-wide computation

Graph Processing Challenges

- How to partition graphs across machines?
 - Need to provide good load balance and locality
- How to support many classes of graph algorithms with a common graph programming model?
 - E.g., algorithms may require exact or approximate outputs
 - E.g., should we use message passing or shared memory?

More Challenges

- How to scale efficiently?
 - Computation per vertex is small
 - Memory accesses have poor locality
 - Parallelism changes over the course of execution
- How to support factored computation efficiently?
 - E.g., avoid any computation if nothing has changed
- What is the consistency model?
 - Sync or async communication, exactly-once, at-least once ...
- How to support fault tolerance?

Today's Papers

- Pregel
 - Partitions graphs using edge cuts (discussed later)
 - Uses message passing based programming model
 - Supports sync communication, with exactly-once semantics
 - Checkpointing for fault tolerance
- Powergraph
 - Partitions graphs using vertex cuts (discussed later)
 - Uses shared-memory based programming model
 - Supports sync communication, with exactly-once semantics
 - Supports async communication, with serializability semantics
 - Checkpointing for fault tolerance