UMD DATA605 - Big Data Systems

12.3: Graph Data Processing

- **Instructor**: Dr. GP Saggese, gsaggese@umd.edu

- ***Typical graph analysis tasks***
- Graph Data Processing Systems

Queries vs Analysis Tasks

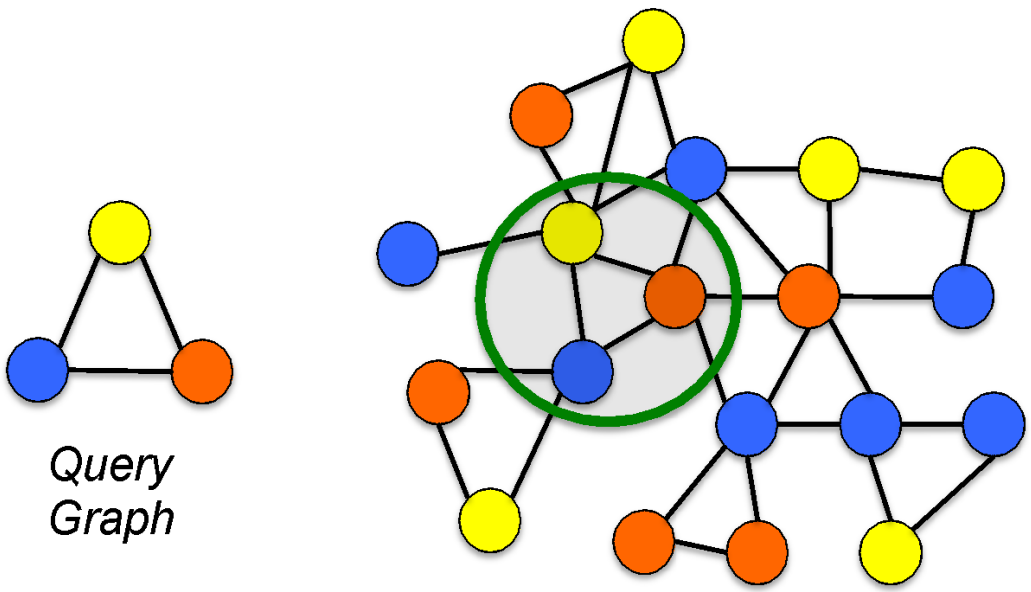
- **Queries**
  - Explore data
  - Result is small graph portion (often a node)
  - Challenges
    - Minimize explored graph portion
    - Use indexes (auxiliary data structures)
- **Analysis tasks**
  - Process entire graph
  - Challenges
    - Handle large data efficiently
    - Parallelize if data doesn't fit in memory/disk

Graph Algorithms

- **Graph algorithms** are general algorithms applicable to various graph types
  - Network flows (e.g., max flow, min cut)
  - Spanning trees (e.g., minimal connectivity)
  - ...
- **Applications**
  - Logistics and supply chain optimization
  - Electric grid design
  - Telecommunications (e.g., bandwidth allocation)

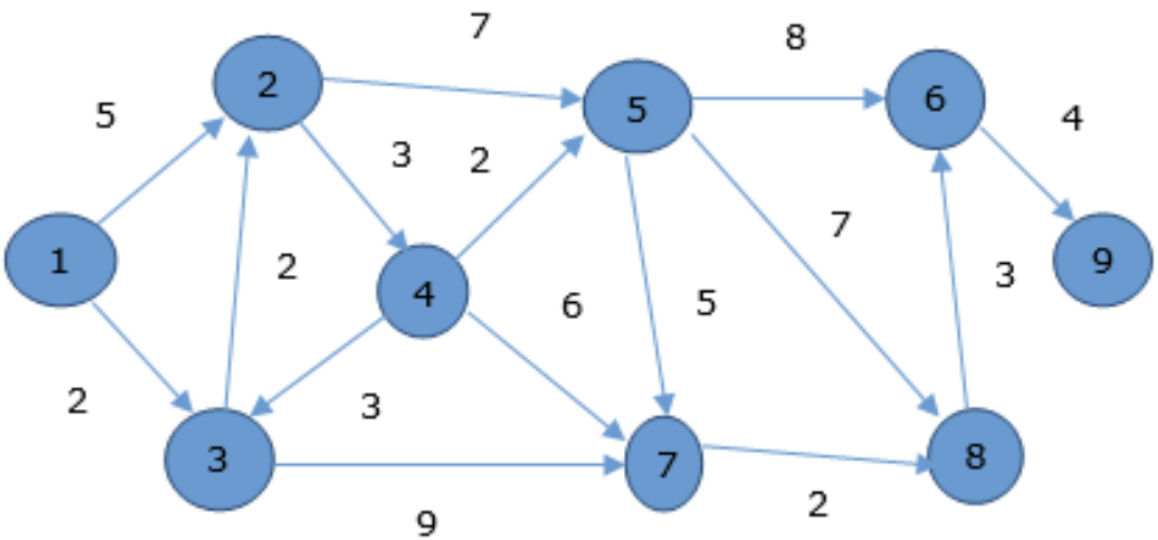
Subgraph Matching

- **Subgraph Matching** = find matching instances of a small pattern in a larger graph
  - Patterns are typically small and fixed
  - Matching can be *approximate*
- **Applications**
  - Fraud detection (e.g., detecting fraudulent transaction patterns)
  - Bioinformatics (e.g., finding protein interaction motifs)
  - Social network analysis (e.g., community pattern detection)



Shortest Path Queries

- **Shortest Path Queries** = find the shortest or optimal path between two nodes
  - May consider edge weights (e.g., distance, cost)
- **Applications**
  - GPS navigation (e.g., Google Maps)
  - Network routing (e.g., Internet traffic routing)
  - Robotics (e.g., path planning)

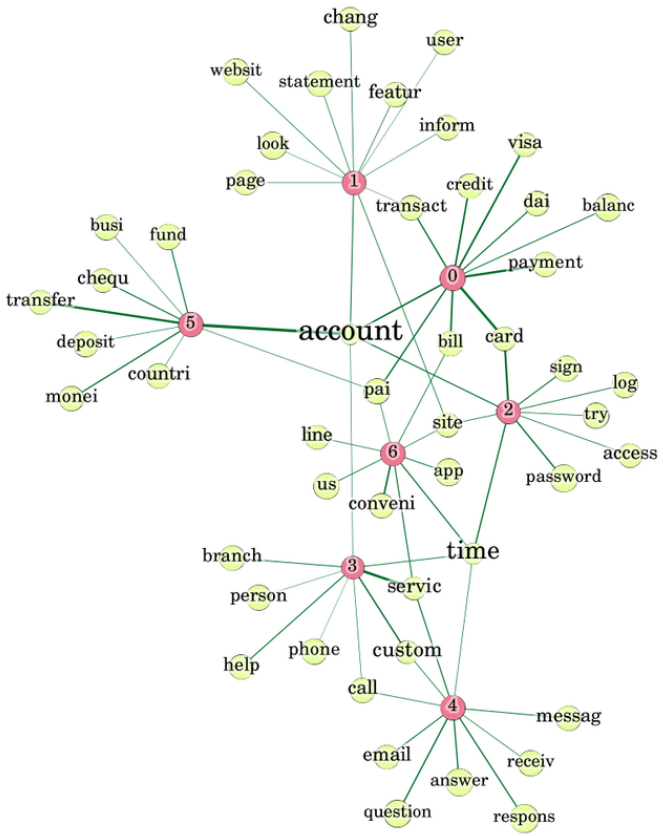


Reachability

- **Reachability** = determine if a path exists between two nodes
  - May include constraints (e.g., edge types, direction)
- **Applications**
  - Access control (e.g., "can user X reach resource Y?")
  - Dependency analysis (e.g., package installation)
  - Workflow engines (e.g., task progression)

Keyword Search

- **Keyword Search** = find smallest subgraph containing all specified keywords
- **Applications**
  - Knowledge graphs (e.g., question answering)
  - Enterprise search (e.g., documents linked by shared entities)
  - Academic citation networks



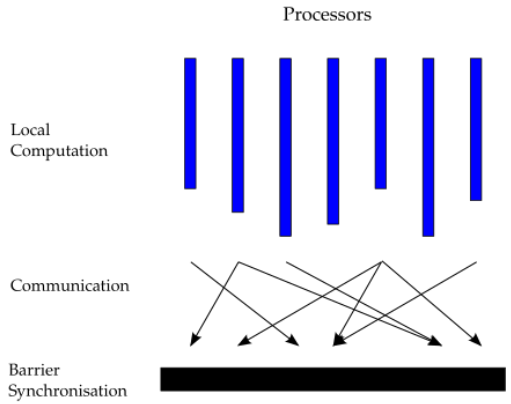
Historical Queries

- **Historical Queries** = find nodes with similar evolution or temporal behavior
  - Typically on dynamic or time-stamped graphs
- **Applications**
  - Stock market analysis (e.g., similar price movement)
  - Social media trends (e.g., user behavior over time)
  - Epidemiology (e.g., disease spread patterns)

- Typical graph analysis tasks
- ***Graph Data Processing Systems***

Bulk Synchronous Parallel Model

- BSP is a model for parallel computation using synchronized steps
- **Computation is organized into supersteps**
  - *Local computation*
  - *Message passing*
    - Parallel processors to exchange data
    - Messages are sent during a superstep but delivered in the next
  - *Synchronization barriers*
    - Ensure all processors complete a superstep before moving on



- **Pros**
  - Deterministic and predictable behavior
  - Simple abstraction for reasoning about parallelism
  - Suitable for graph computations with regular communication patterns
- **Cons**
  - Global synchronization can cause idle time (stragglers)
  - Inefficient for highly irregular workloads or dynamic computation patterns

Pregel

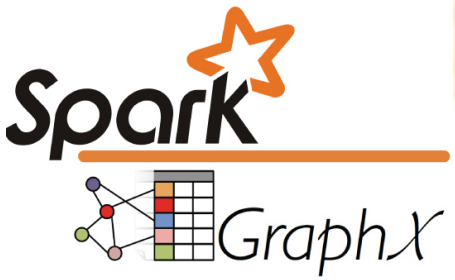
- **Pregel** is a graph processing framework inspired by the BSP model
  - Pregel = Parallel + Graph + Google
  - Designed for large-scale distributed graph computations
  - Uses a vertex-centric programming model
    - Each vertex performs computation independently in supersteps
    - Motto: *"Think like a vertex"*
  - Vertices
    - Send messages to neighbors
    - Update state based on received messages
    - Can vote to halt; computation ends when all are inactive
- **PageRank example**
  - Each vertex divides its rank among neighbors
  - In each superstep, vertices update rank from incoming messages
- **Connected components example**
  - Each vertex propagates smallest ID seen to neighbors
  - Repeats until no changes occur

Apache Giraph

- **Apache Giraph** is an open-source implementation of Pregel
  - Written in Java for high scalability
- Suitable for batch-processing large graphs
- **Integrates with Hadoop ecosystem**
  - Runs on Hadoop MapReduce
  - Uses HDFS for storage and YARN for resource management

- **Example use cases**:
  - Social network analysis (e.g., friend recommendation)
  - Web graph analysis (e.g., PageRank)
  - Biological network exploration (e.g., gene interaction)

Apache Spark GraphX



- **GraphX** is Spark's API for graph-parallel computation
- Seamless integration with Spark ecosystem and pipelines
- Graphs represented using RDDs
  - Vertices and edges as distributed collections
- Includes a Pregel API
  - Enables iterative message-passing computations
  - Preserves immutability and functional programming style
- **Strengths vs traditional graph DBs**
  - Scalable to very large graphs
  - Efficient for batch analytics and machine learning workflows
- **Limitations**
  - Less suitable for real-time queries and transactional workloads
  - Requires understanding of Spark internals for tuning performance