universität koblenz
weiter:denken

Big Data
Session 6: Processing Large Graph Data

Frank Hopfgartner
Institute for Web Science and Technologies

Previous week

universität koblenz
weiter:denken

- Introduction to data streaming
- Record-at-a-time streaming
- Declarative, functional streaming
- Declarative, relational streaming

Intended Learning Outcomes

universität koblenz
weiter:denken

At the end of this lecture, you will be able to:

- Explain network theory
- Distinguish between different types of graphs
- Provide graph processing examples
- Outline various distributed systems for graph processing

Outline

universität koblenz
weiter:denken

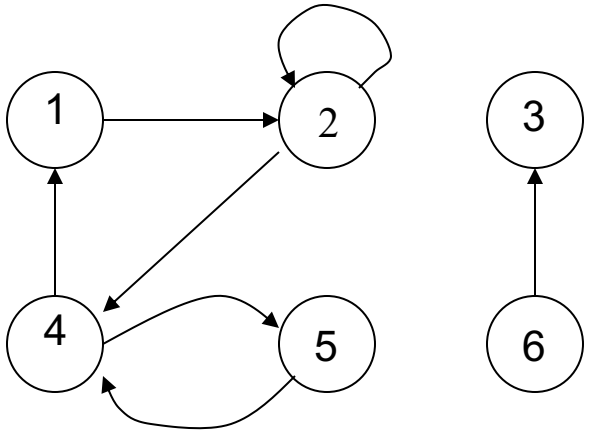
- Network Theory (Briefly)
- Data representation
- Graph Processing Examples
- Distributed Systems for Graph Processing

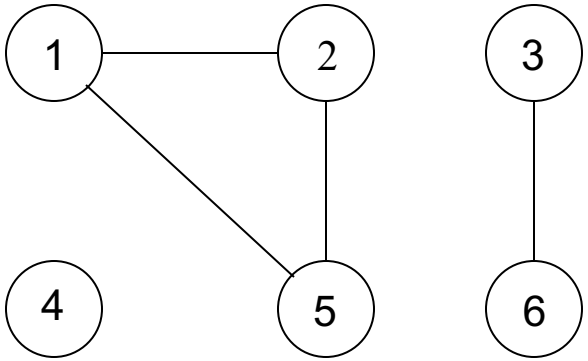
What is a network?

universität koblenz
weiter:denken

Network = graph

Informally a graph is a set of nodes joined by a set of lines or arrows.





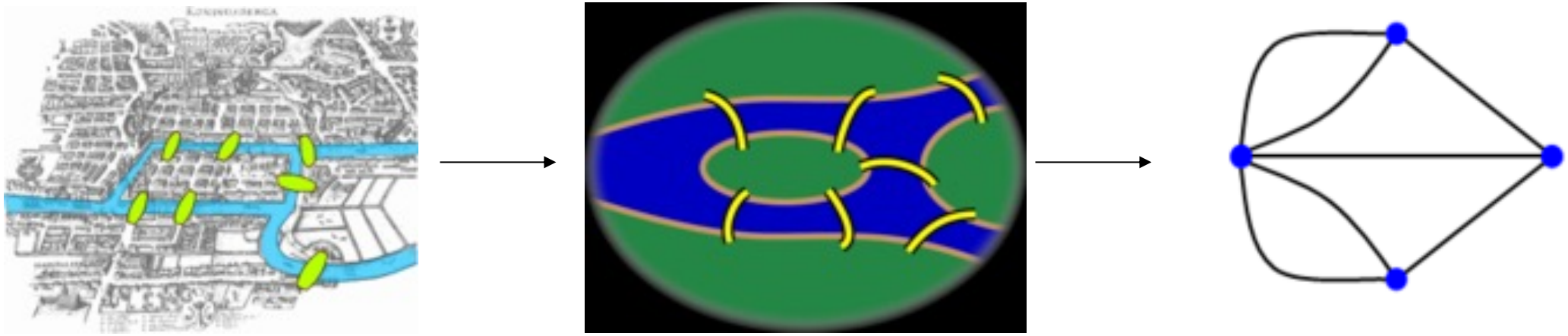
Graph-based representations

universität koblenz
weiter:denken

- Representing a problem as a graph can provide a different point of view
- Representing a problem as a graph can make a problem much simpler
  - More accurately, it can provide the appropriate tools for solving the problem

Graph Theory - History

Leonhard Euler's paper on *“Seven Bridges of Königsberg”* , published in 1736.



What is network theory?

universität koblenz
weiter:denken

- Network theory provides a set of techniques for analysing graphs
- Applying network theory to a system means using a graph-theoretic representation

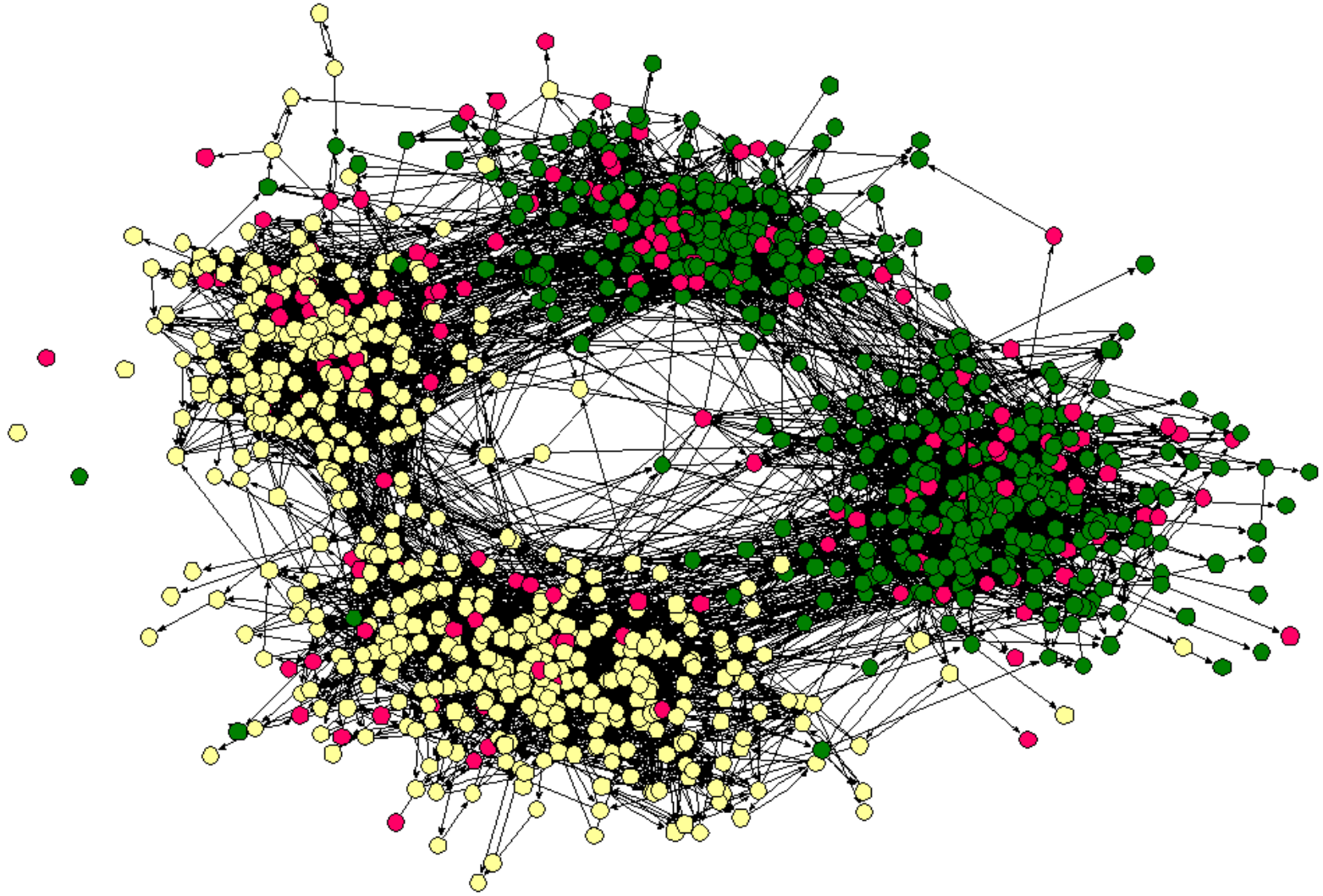
What makes a problem graph-like?

universität koblenz
weiter:denken

- There are two components to a graph
  - Nodes and edges
- In graph-like problems, these components have natural correspondences to problem elements
  - Entities are nodes and interactions between entities are edges
- Most complex systems are graph-like

Example: Social Networks

universität koblenz
weiter:denken



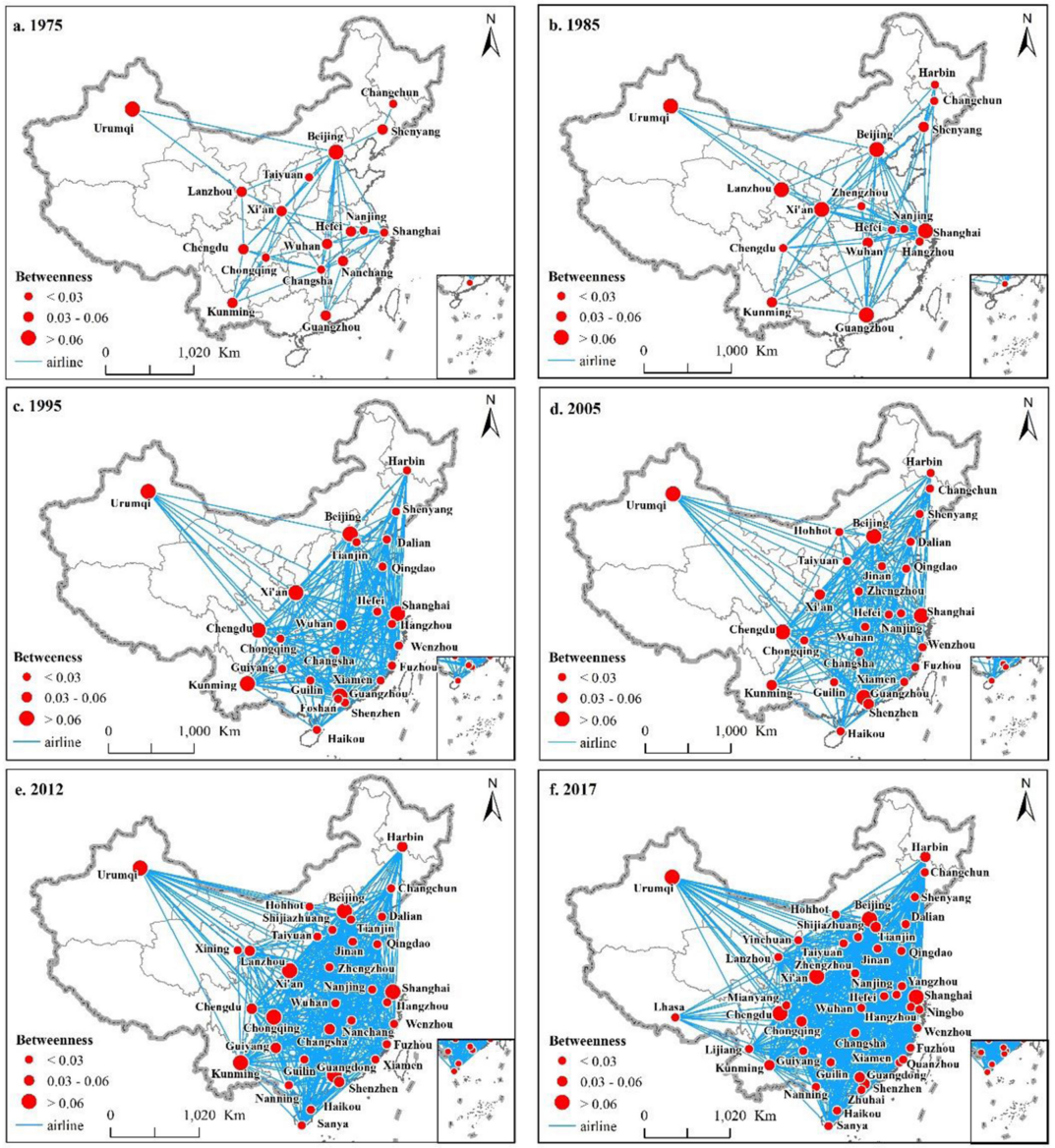
David Camacho, Ángel Panizo-LLedot, Gema Bello-Orgaz, Antonio Gonzalez-Pardo, Erik Cambria. The four dimensions of social network analysis: An overview of research methods, applications, and software tools, *Information Fusion*, Volume 63, 2020, Pages 88-120

This can also look pretty

universität koblenz
weiter:denken

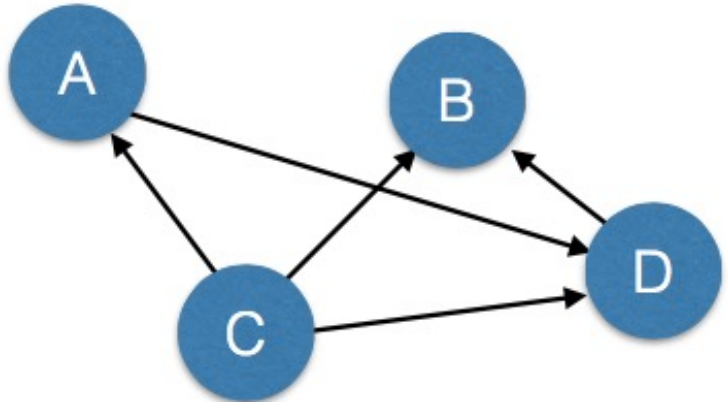
Source: https://www.facebook.com/notes/10158791468612200/

Example: Transportation Networks



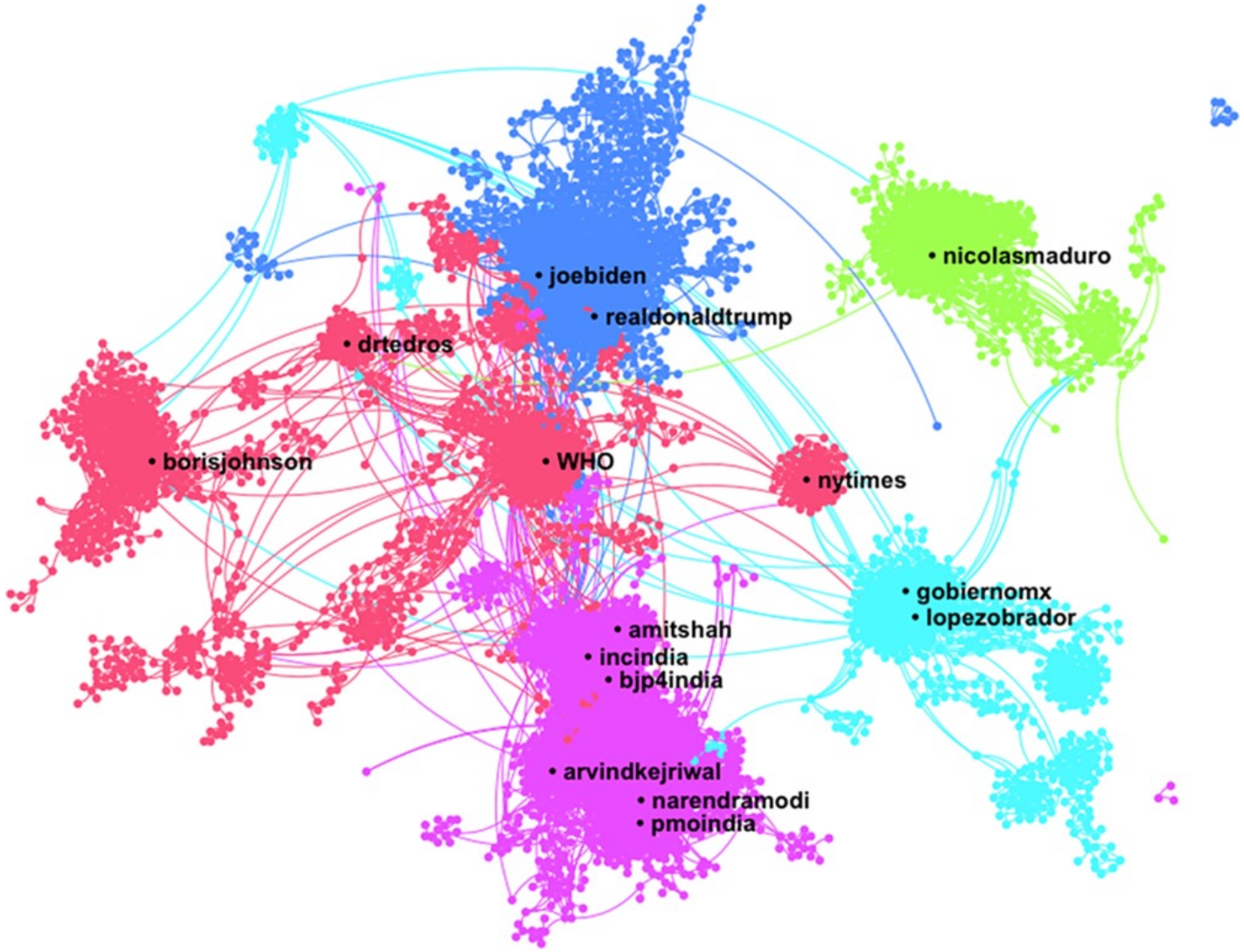
Yu Chen, Jiaoe Wang, Fengjun Jin. Robustness of China's air transport network from 1975 to 2017, *Physica A: Statistical Mechanics and its Applications*, Volume 539, 2020.

Example: Web pages (Hyperlinks)



Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, Janet Wiener, Graph structure in the Web, *Computer Networks*, Volume 33, Issues 1–6, 2000

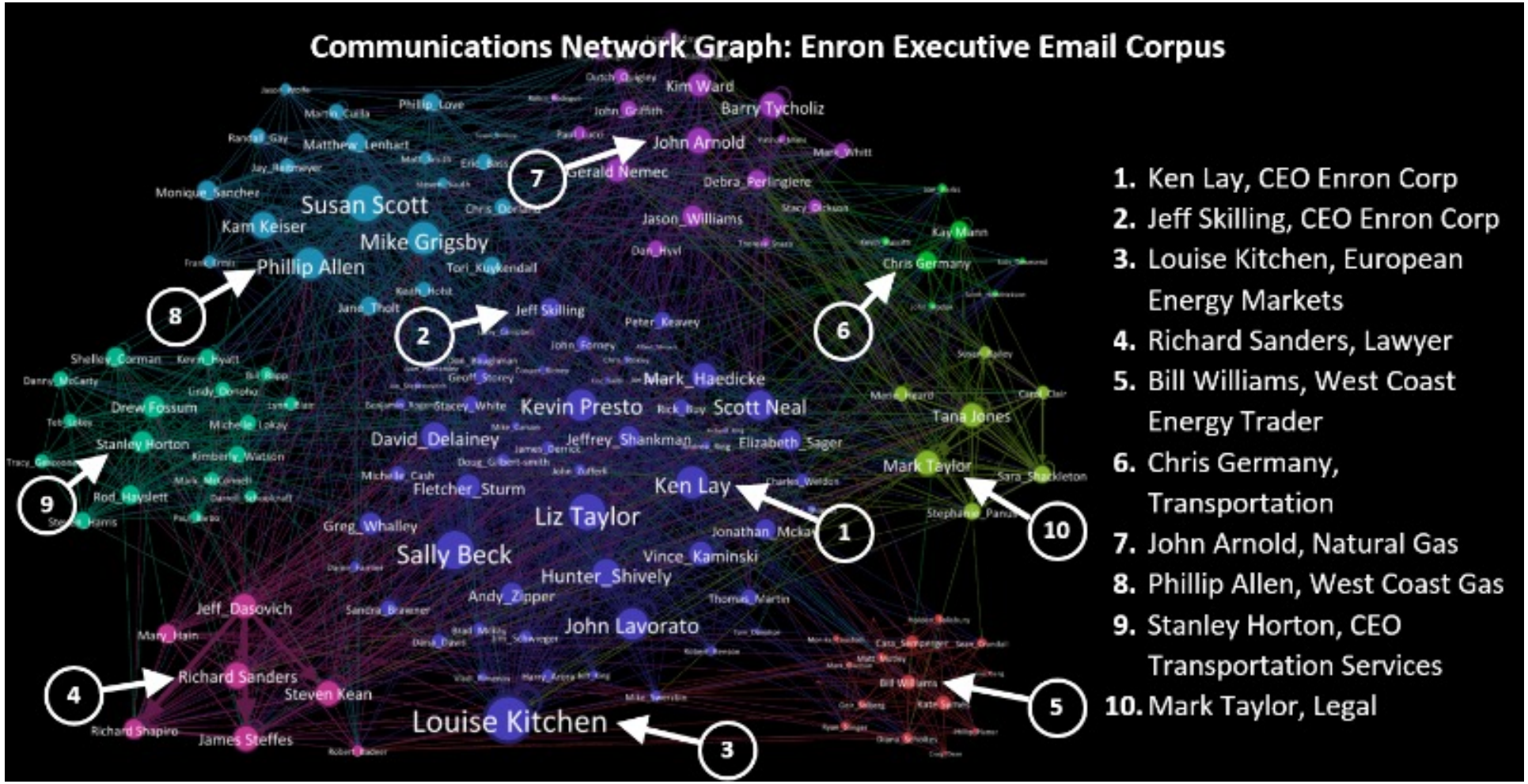
Example: Twitter/X



Pascual-Ferrá, P., Alperstein, N., & Barnett, D. (2022). Social Network Analysis of COVID-19 Public Discourse on Twitter: Implications for Risk Communication. *Disaster Medicine and Public Health Preparedness, 16*(2), 561-569. doi:10.1017/dmp.2020.347

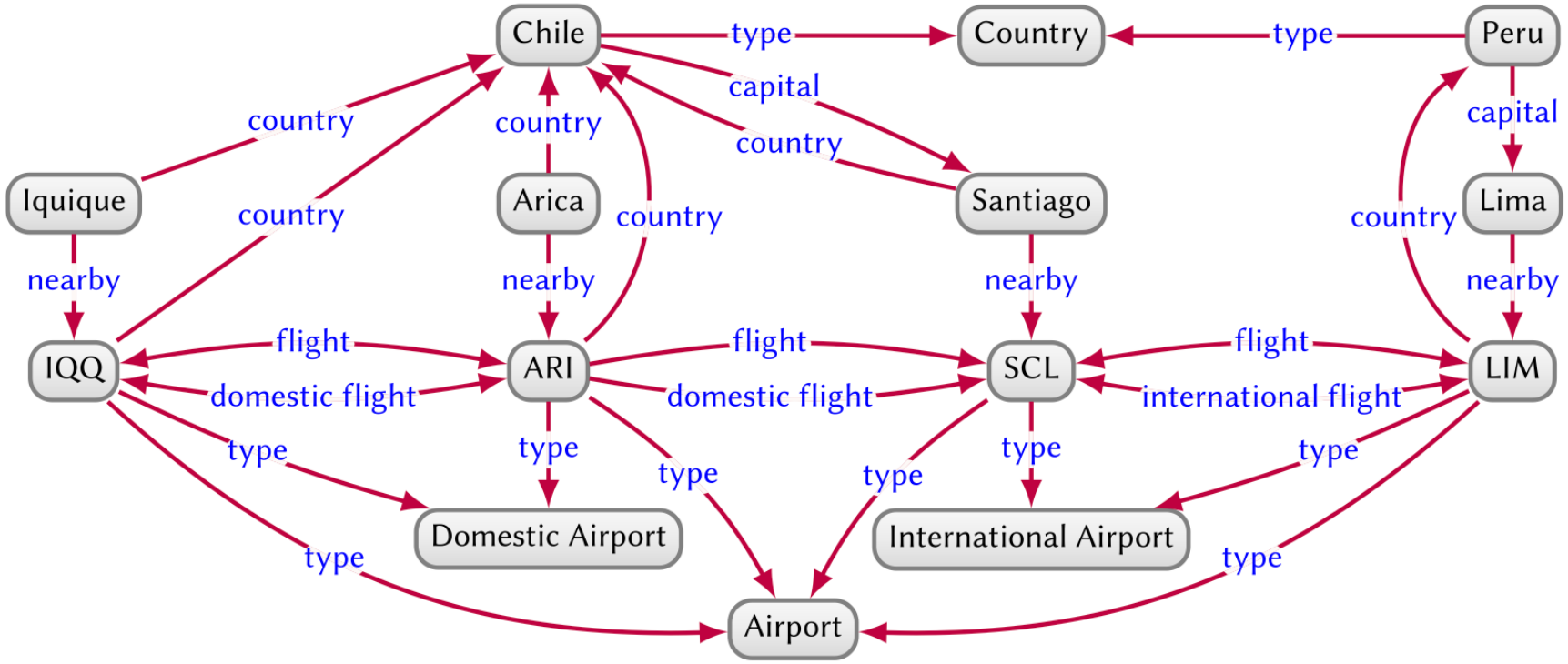
Example: eMails

universität koblenz
weiter:denken



https://www.linkedin.com/pulse/visualizing-email-networks-damon-wade

Example: Knowledge Graphs



Hogan et al. 2021. Knowledge Graphs. *ACM Comput. Surv*. 54, 4, Article 71 (May 2022), 37 pages.

universität koblenz
weiter:denken

What can these data graphs be used for?

Data Graphs – Use Cases

universität koblenz
weiter:denken

- Recommend users in a social network
  - People you may know
  - WTF in Twitter
- Path planning
  - Shortest way from home to office
- Graph clustering
  - Find communities in social graphs
  - Scientific communities based on citation network

Outline

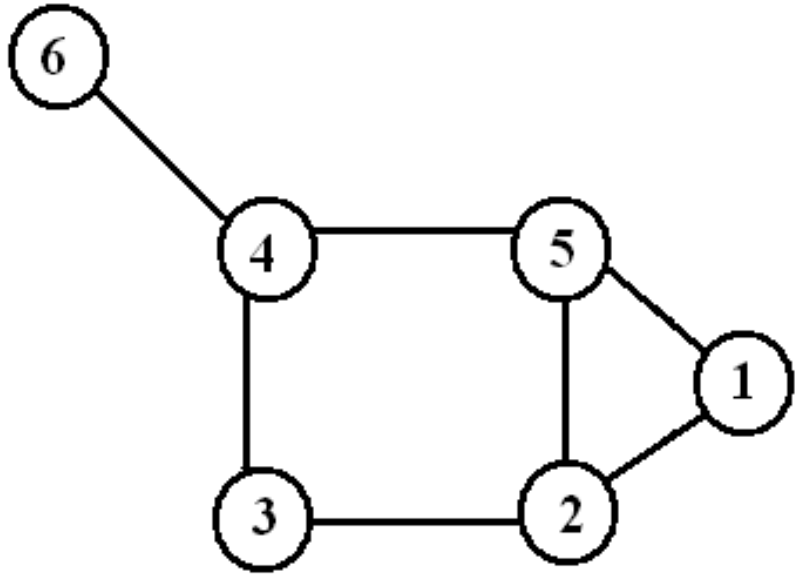
universität koblenz
weiter:denken

- Network Theory (Briefly)
- **Data representation**
- Graph Processing Examples
- Distributed Systems for Graph Processing

Definition: Graph

- G is an ordered triple G:=(V, E, f)
  - V is a set of nodes, points, or **vertices**.
  - E is a set, whose elements are known as **edges** or lines.
  - f is a function
    - maps each element of E
    - to an unordered pair of vertices in V.
- Vertex
  - Basic Element
  - Drawn as a node or a dot.
  - Vertex set of G is usually denoted by V(G), or V
- Edge
  - A set of two elements
  - Drawn as a line connecting two vertices, called end vertices, or endpoints.
  - The edge set of G is usually denoted by E(G), or E.

Example



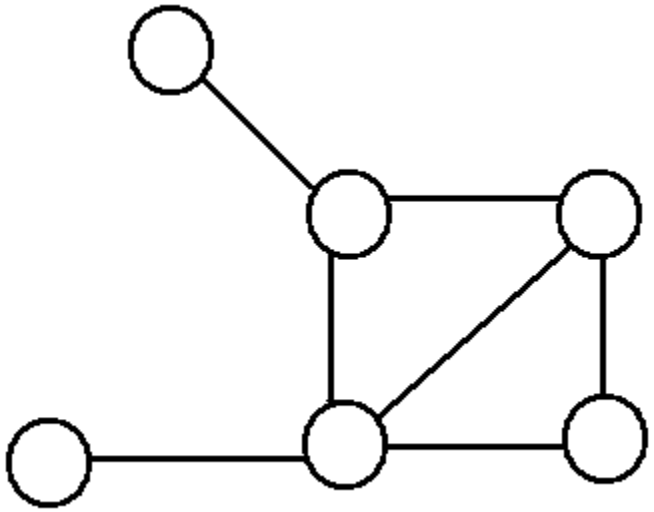
V:={1,2,3,4,5,6}

E:={{1,2},{1,5},{2,3},{2,5},{3,4},{4,5},{4,6}}

Simple Graphs

universität koblenz
weiter:denken

Graphs without multiple edges or self-loops.

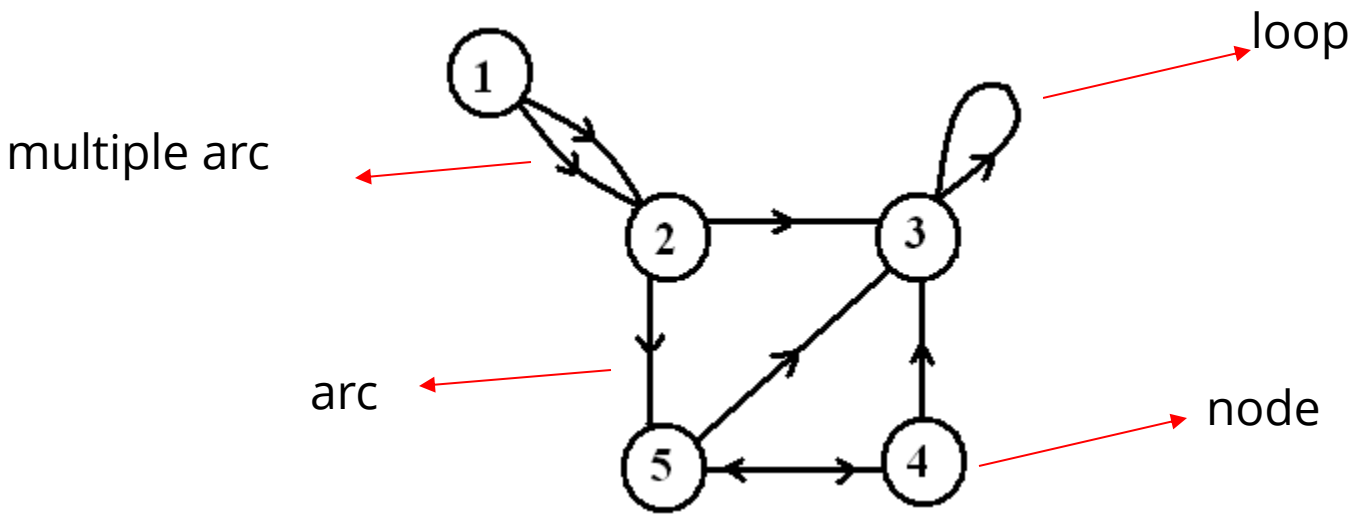


Simple Graphs

universität koblenz
weiter:denken

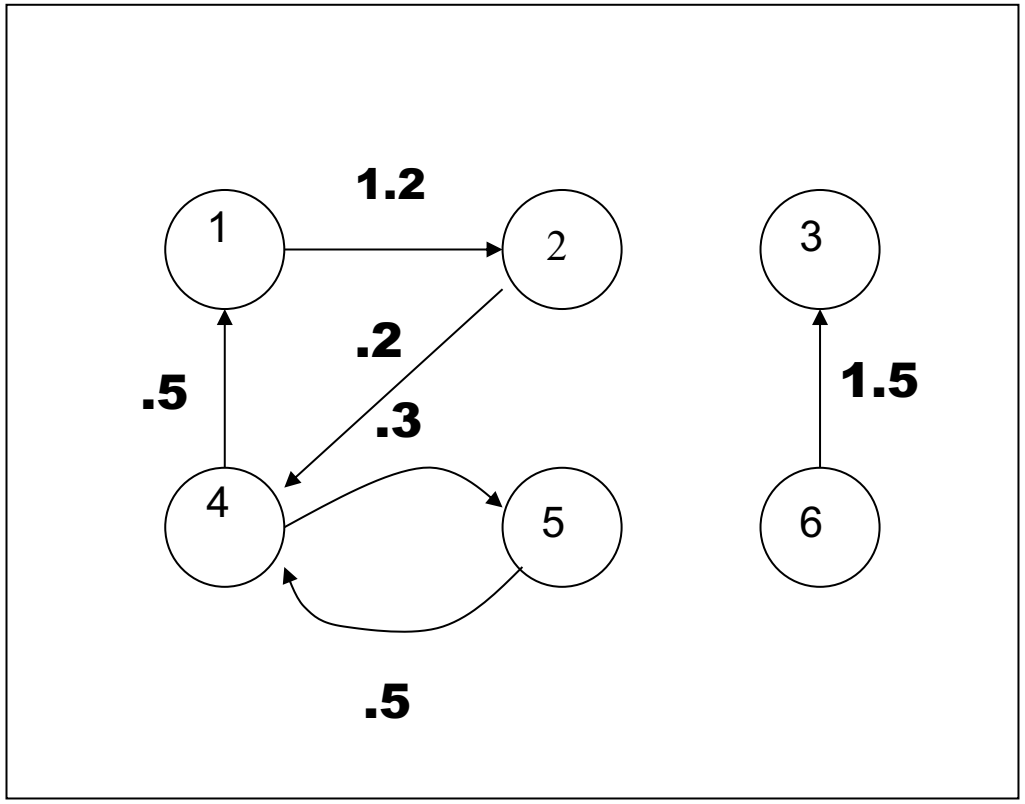
Edges have directions

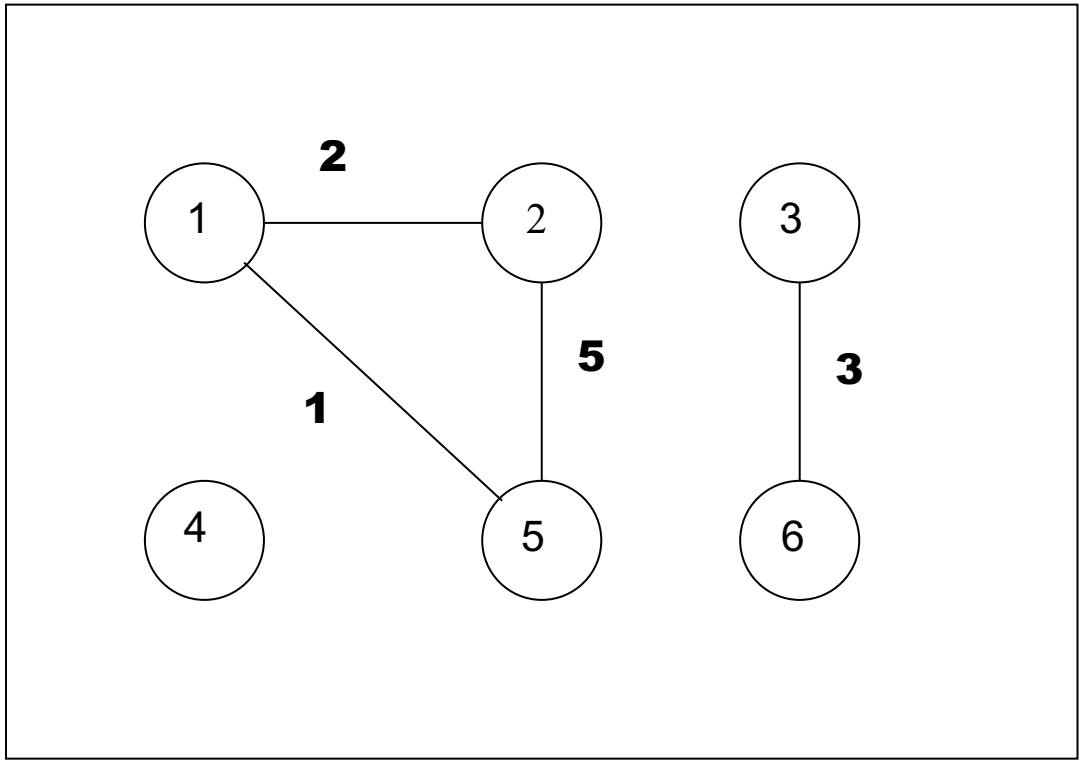
An edge is an ordered pair of nodes



Weighted graphs

Graphs for which each edge has an associated ***weight***, usually given by a ***weight function*** *w: E* → **R**.



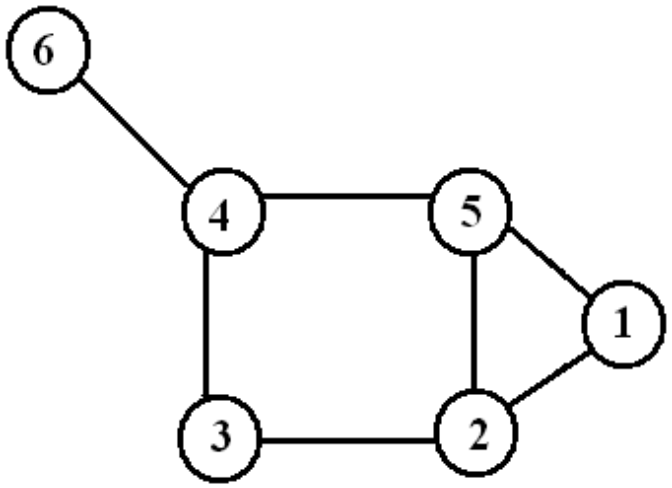


Representation (Matrix)

universität koblenz
weiter:denken

- Incidence Matrix
  - V x E
  - [vertex, edges] contains the edge's data
- Adjacency Matrix
  - V x V
  - Boolean values (adjacent or not)
  - Or Edge Weights

Matrices



	1,2	1,5	2,3	2,5	3,4	4,5	4,6
1	1	1	0	0	0	0	0
2	1	0	1	1	0	0	0
3	0	0	1	0	1	0	0
4	0	0	0	0	1	1	1
5	0	1	0	1	0	1	0
6	0	0	0	0	0	0	1

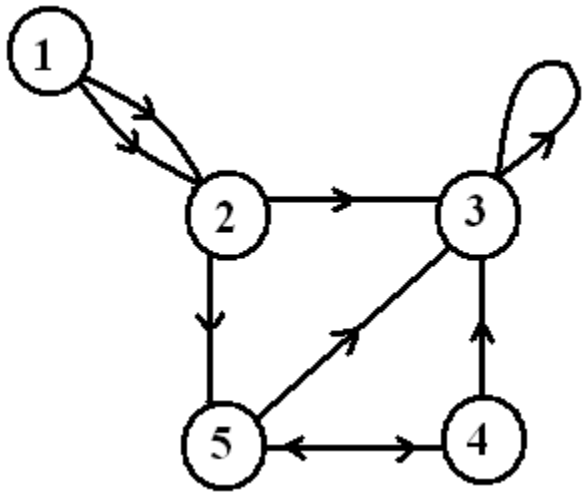
	1	2	3	4	5	6
1	0	1	0	0	1	0
2	1	0	1	0	1	0
3	0	1	0	1	0	0
4	0	0	1	0	1	1
5	1	1	0	1	0	0
6	0	0	0	1	0	0

Representation (List)

universität koblenz
weiter:denken

- Edge List
  - pairs (ordered if directed) of vertices
  - Optionally weight and other data
- Adjacency List (node list)
  - an array of |V | lists, one for each vertex in V.
  - For each u $\in$ V , ADJ [u] points to all its adjacent vertices.

Edge and Node Lists



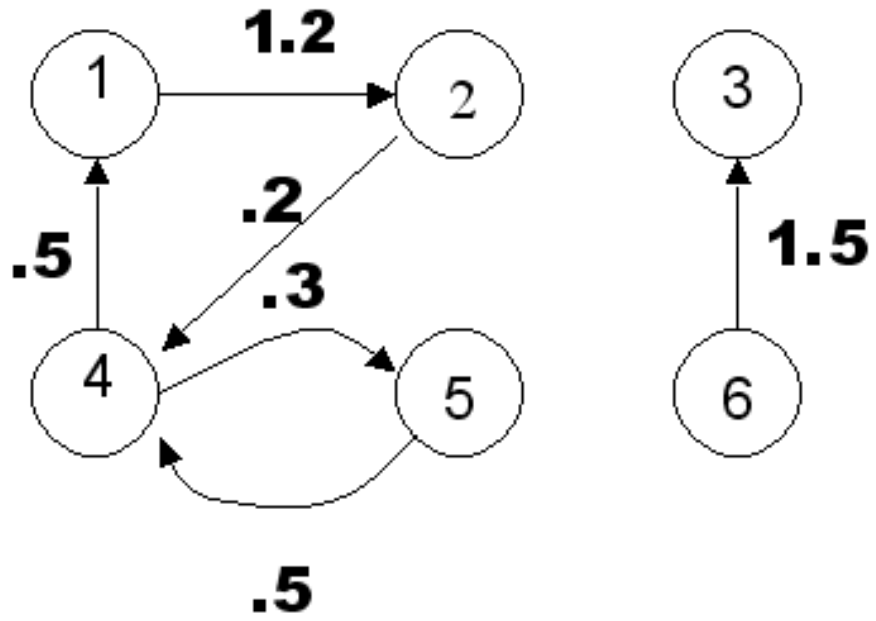
Edge List

1 2
1 2
2 3
2 5
3 3
4 3
4 5
5 3
5 4

Node List

1 2 2
2 3 5
3 3
4 3 5
5 3 4

Edge Lists for Weighted Graphs



Edge List

1 2 1.2
2 4 0.2
4 5 0.3
4 1 0.5
5 4 0.5
6 3 1.5

Outline

universität koblenz
weiter:denken

- Network Theory (Briefly)
- Data representation
- **Graph Processing Examples**
- Distributed Systems for Graph Processing

Real world graphs

universität koblenz
weiter:denken

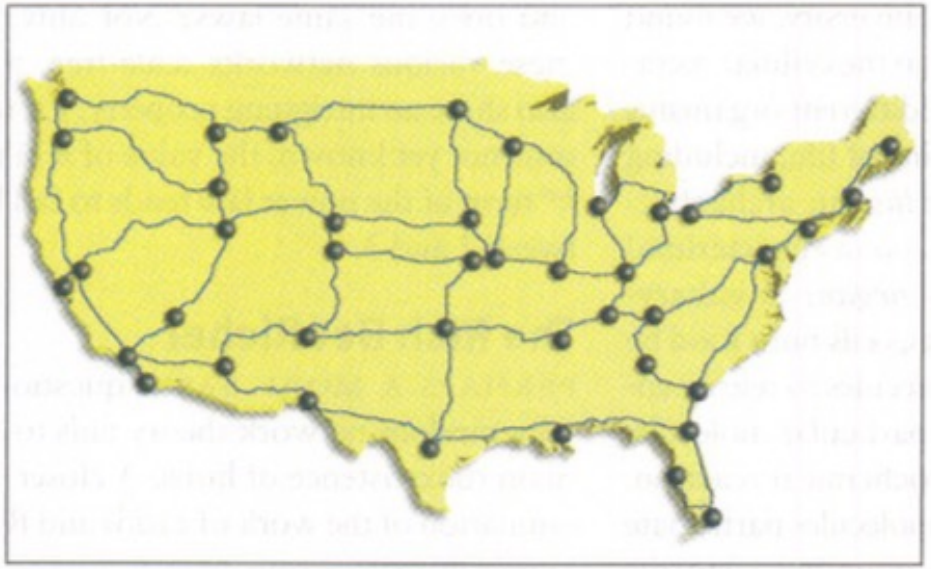
- Millions or billions of nodes & millions or billions of edges
  - But real world graphs are typically **sparse**:
  - The number of actual edges is far smaller than the number of possible edges
- How many possible edges in a friendship graph with **n** nodes?
  - n=100?
  - n=2’000’000’000?

Classes of Large Graphs

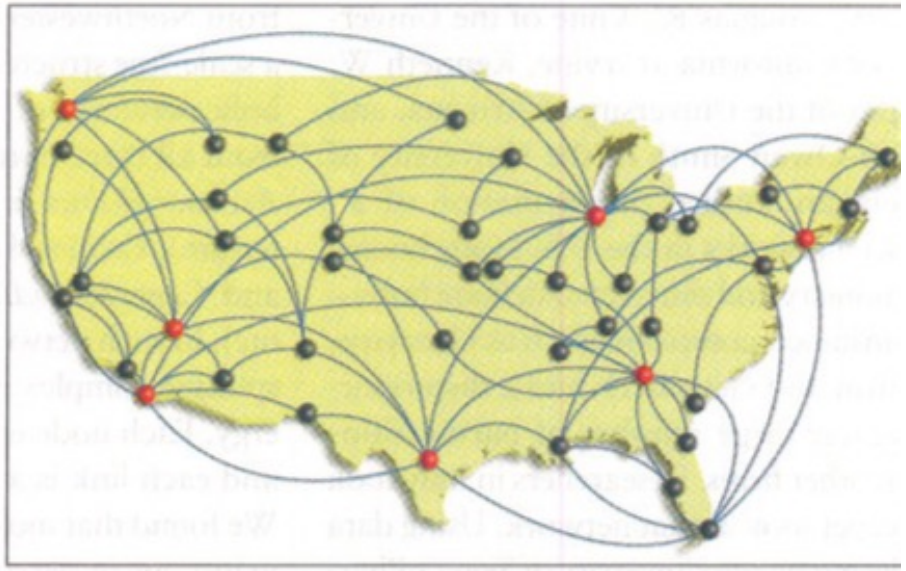
universität koblenz
weiter:denken

- Random graphs
  - Node degree is constrained
  - Less common
- Scale-free graphs
  - Distribution of node degree follows power law
  - Most large graphs are scale-free
  - Small world phenomena & hubs
  - Harder to partition

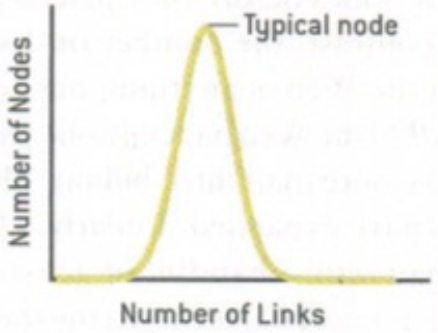
Random Network



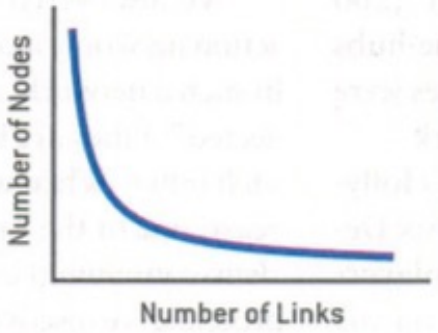
Scale-Free Network

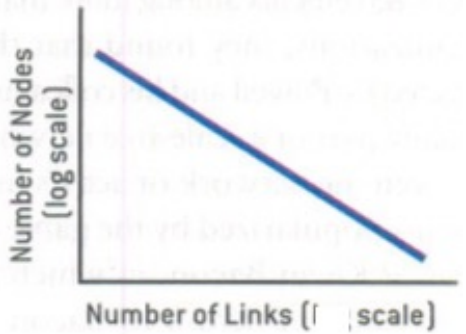


Bell Curve Distribution of Node Linkages



Power Law Distribution of Node Linkages





Scale-free networks

universität koblenz
weiter:denken

- Scale free networks are characterized by two properties: **continuous growth** and **preferential attachment**

- Earlier models of networks did not include the addition of nodes over time. The graphs remained static. However, in real life there are always new people that are joining a social network, new websites that are created and so on that cause the network to be in a state of continuous growth.

Preferential attachment

universität koblenz
weiter:denken

- The other characteristic of scale-free networks, preferential attachment, means that new nodes tend to connect to nodes that are already well connected.

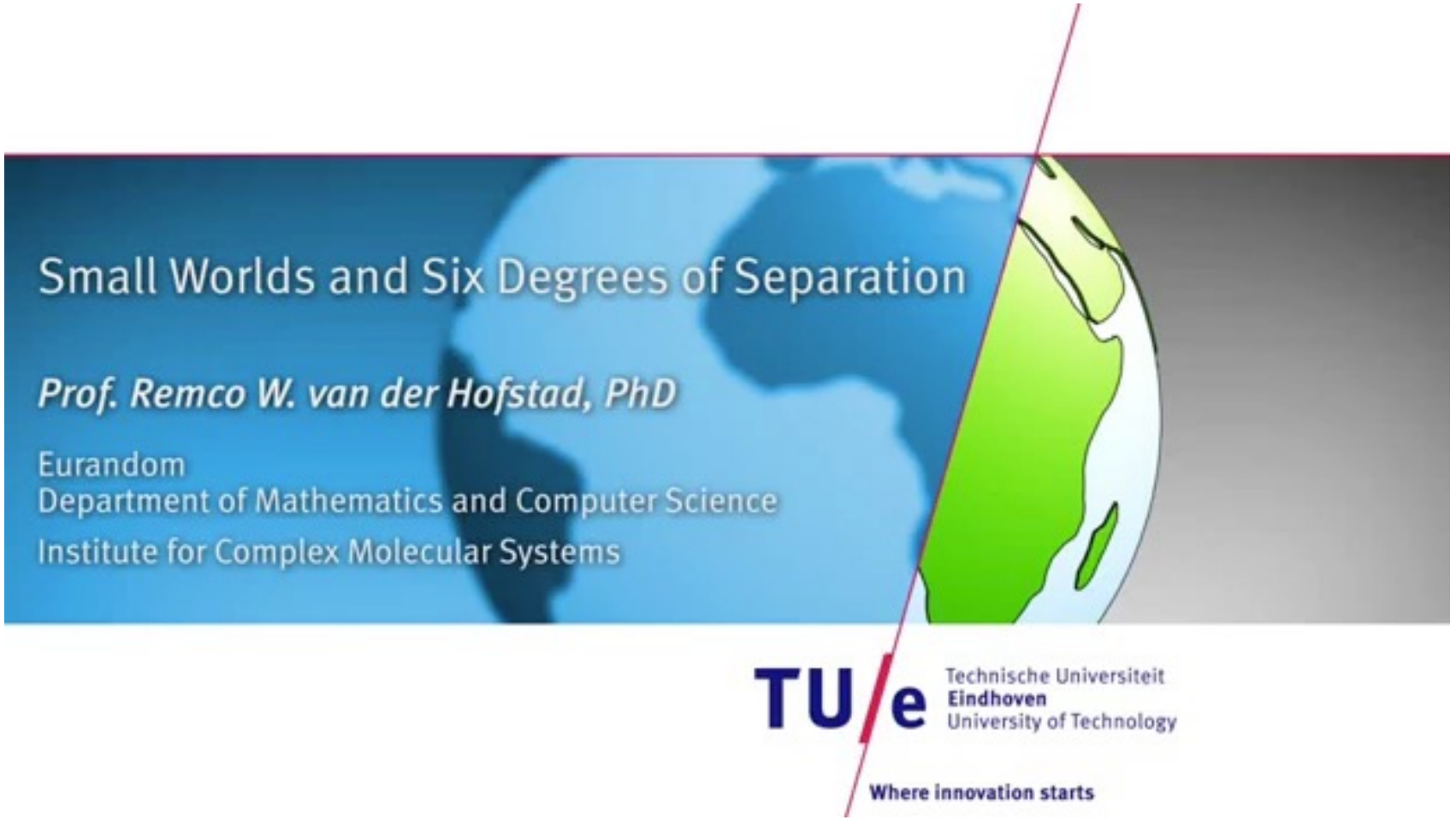
- For example, new websites are likely to have links to very connected websites like Google, Wikipedia, etc.

Degree of separation in social networks

universität koblenz
weiter:denken

- Graph diameter
  - Distance (i.e., steps/edges) between two nodes in the graph
- Milgram’s Six degrees of separation

Small worlds and six degree of separation



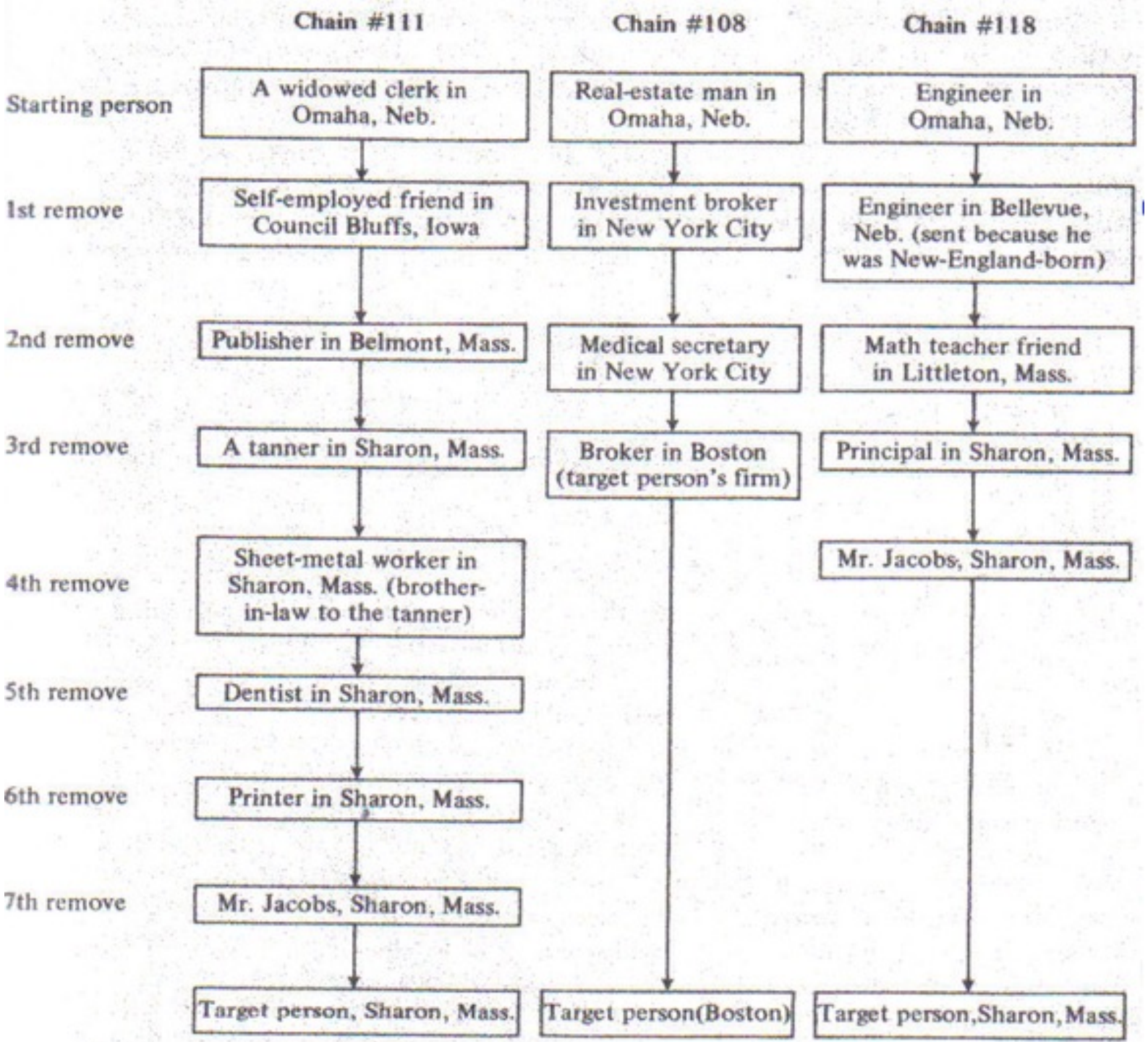
https://www.youtube.com/watch?v=LeUhVC8xVQI

Milgram's (Small-World) experiment

universität koblenz
weiter:denken

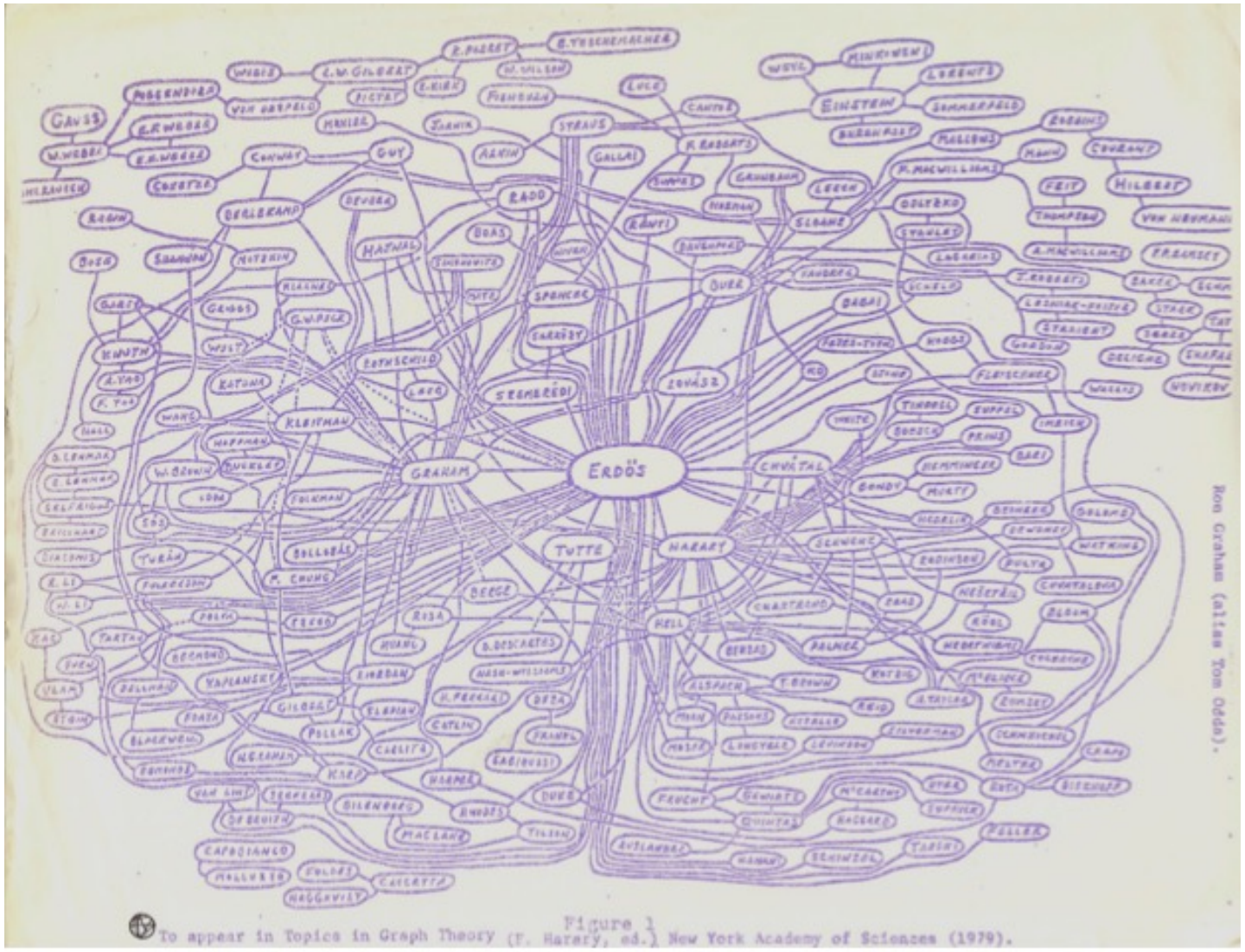
- Average length of the chains that were completed lied between 5 and 6 steps;
  - Coined as "Six degrees of separation" principle.
  - This was far less than assumed under the 'grid-like' assumption!
  - Similar results have been found in many other social networks

Lars Backstrom, Paolo Boldi, Marco Rosa, Johan Ugander, and Sebastiano Vigna. 2012. Four degrees of separation. In Proceedings of the 4th Annual ACM Web Science Conference (WebSci '12). Association for Computing Machinery, New York, NY, USA, 33–42.



Erdős number

universität koblenz
weiter:denken



Erdos Number Project: http://www.oakland.edu/enp/index.html

Example: My Erdős number

Outline

universität koblenz
weiter:denken

- Network Theory (Briefly)
- Data representation
- Graph Processing Examples
- **Distributed Systems for Graph Processing**
  - **Pregel**
  - **Giraph**
  - **Spark GraphX**

Pregel (2010)

universität koblenz
weiter:denken

- Pregel is a large-scale **graph-parallel** distributed analytics engine

- Some Characteristics:
  - In-Memory (opposite to MapReduce)
  - High scalability
  - Automatic fault-tolerance
  - Flexibility in expressing graph algorithms
  - Message-Passing programming model
  - Tree-style, master-slave architecture
  - Synchronous

- Pregel is inspired by Valiant's **Bulk Synchronous Parallel** (BSP) model

Vertex centric model

universität koblenz
weiter:denken

- Simple distributed programming model
- Algorithms are expressed by “thinking like a vertex”
- A vertex contains information about itself and the outgoing edges
- Computation is expressed at vertex level
- Vertex execution take place in parallel and are interleaved with synchronized message exchanges

Pregel‘s Computation Model

universität koblenz
weiter:denken

- Consists of a sequence of iterations (supersteps), where the same user-defined function is executed for each vertex

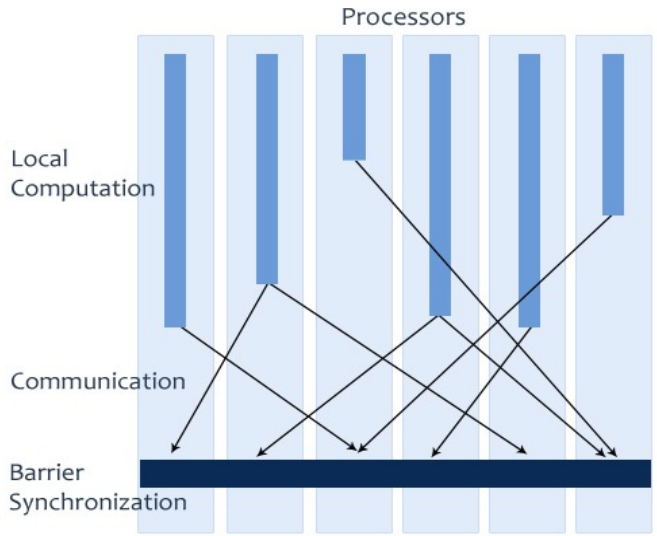
- This function specifies behaviour at a single vertex V and superset S. It can read messages sent to the vertex in super Steps-1, send messages to other vertices that will be read in superset S+1, and modify the state of V and ist outgoing edges.

Entities and Super-steps

universität koblenz
weiter:denken

- The computation is described in terms of vertices, edges and a sequence of **super-steps**

- You give Pregel a *directed* graph consisting of vertices and edges
  - Each vertex is associated with a modifiable user-defined value
  - Each edge is associated with a source vertex, value and a destination vertex

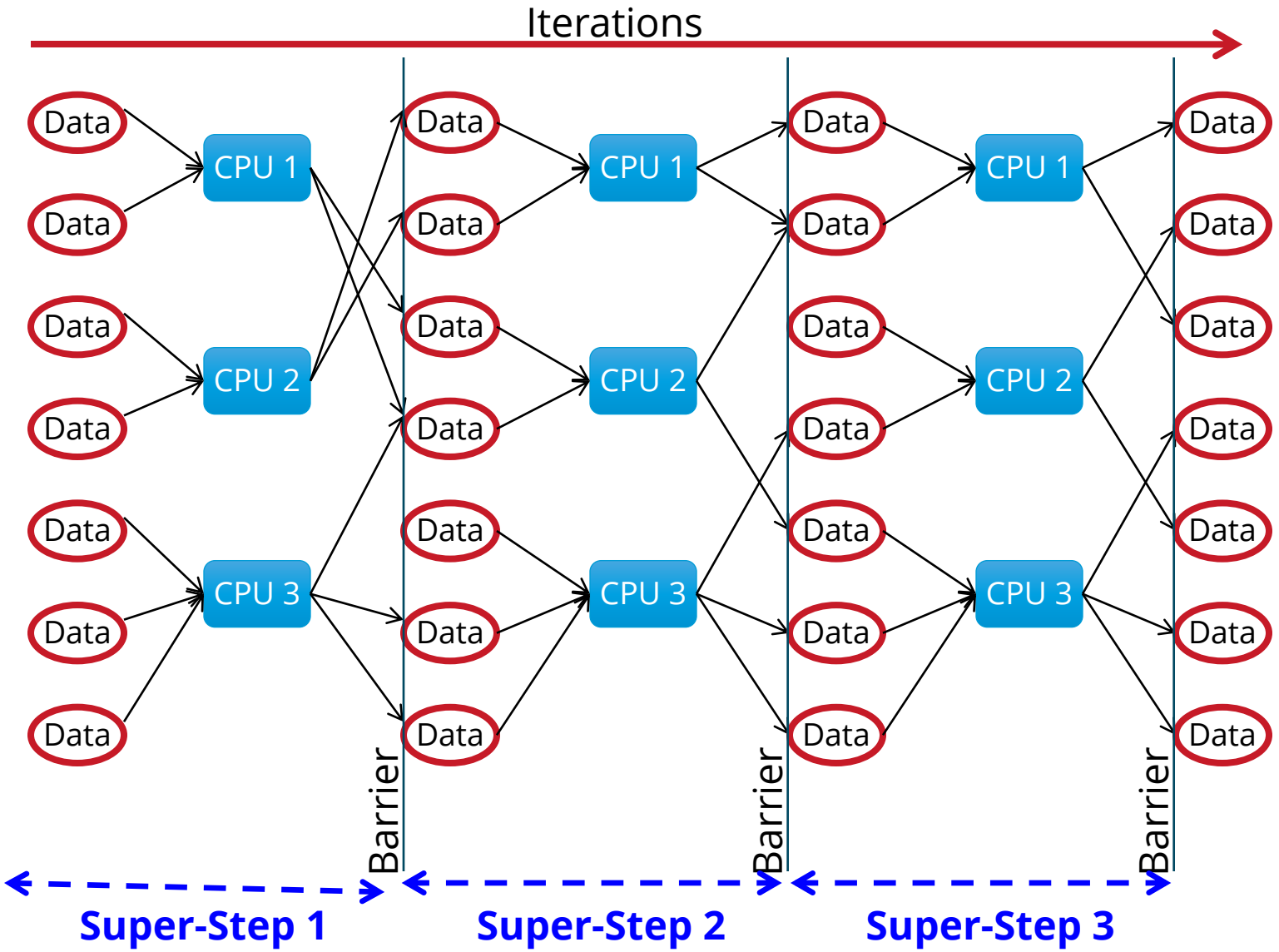


Vertical Structure of a Superstep

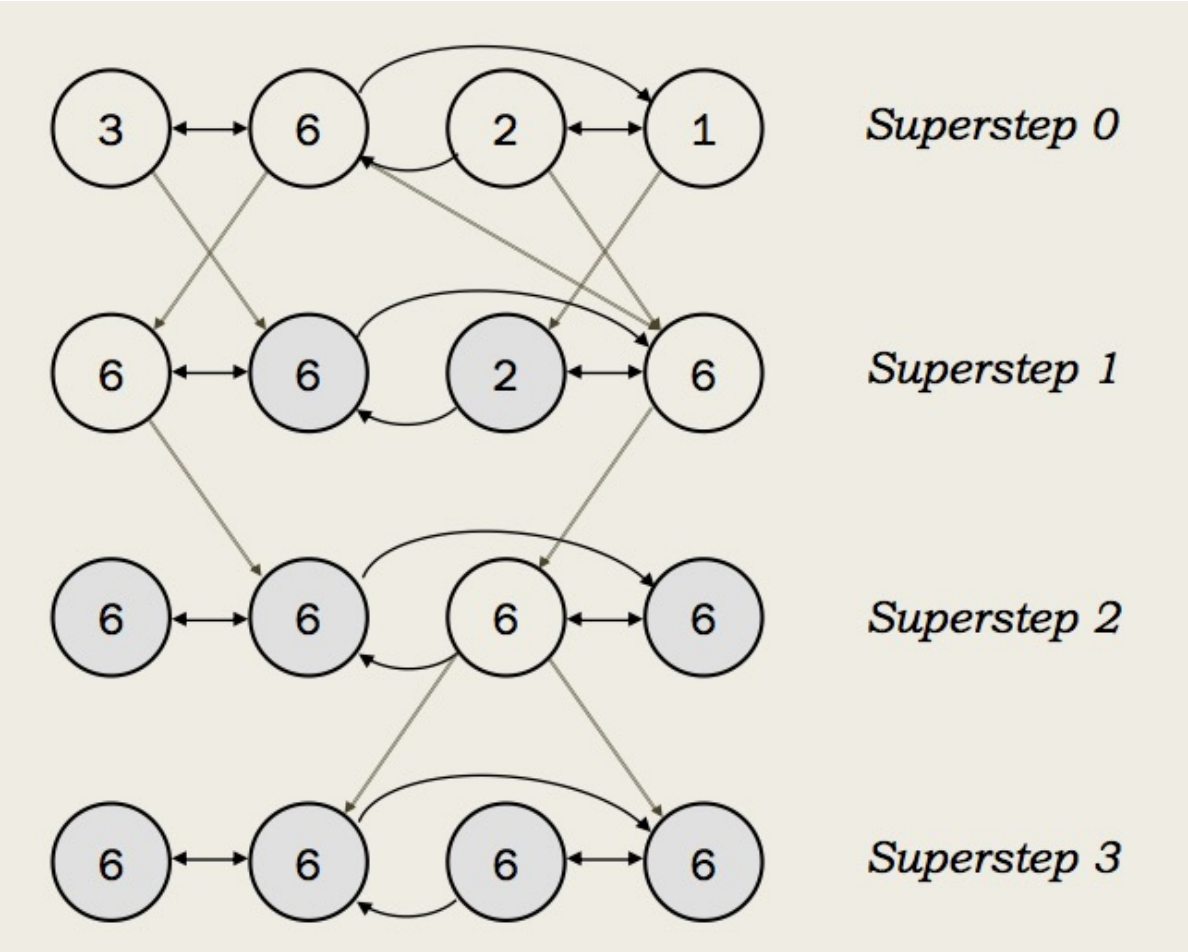
- During a super-step:
  - A user-defined function ***F*** is executed at each vertex ***V***
  - ***F*** can read messages sent to ***V*** in superset ***S – 1*** and send messages to other vertices that will be received at superset ***S + 1***
  - ***F*** can modify the state of ***V*** and its outgoing edges
  - ***F*** can alter the *topology* of the graph

The BSP Model

universität koblenz
weiter:denken



Example: Find maximum value

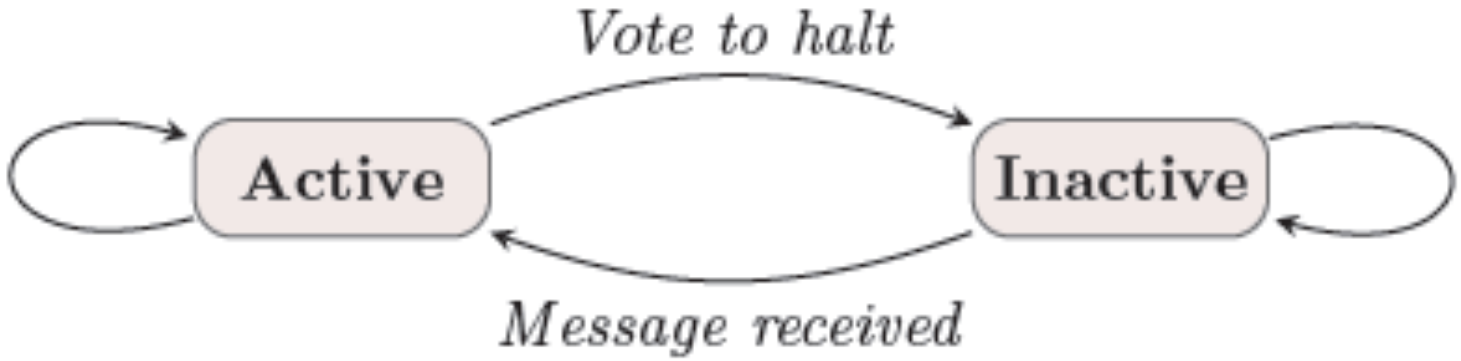


Vertex State Machine

universität koblenz
weiter:denken

- Initially each vertex is in an active state. Each vertex can ‘vote to halt’, where it runs no further computation in any further super step unless it receives a message from another vertex.

- It is then reactivated again and needs to explicitly vote to halt to deactivate itself again. Algorithm terminates when all vertices have halted.



Pregel in detail

universität koblenz
weiter:denken

- Master node
  - Maintains worker
  - Recovers faults of workers
  - Provides Web-UI monitoring tool of job progress
- Worker
  - Processes its task
  - Communicates with the other workers

Pregel in detail

universität koblenz
weiter:denken

- Input and output
  - Can be generated from any arbitrary format and stored in a form most suitable for a given application
- Fault tolerance
  - Archived through checkpointing
  - Master instructs workers to save their state to persistent storage at the beginning of each superstep
  - If masters detect workers are down, it reassigns their partitions to available workers

Pregel

universität koblenz
weiter:denken

- Scalability
  - Local actions
  - Each vertex is processed **independently**
  - Action composition
- Node-centric approach
  - Each node is associated with a value
  - Edges are associate with source nodes
  - Output: values for each node (e.g., PageRank scores to be distributed to other nodes)

Pregel – Graph partitioning

universität koblenz
weiter:denken

- Graph is divided into partitions, each consisting of a set of vertices and all those vertices outgoing edges in each iteration
- Assignment of a vertex to a partition depends on the vertex ID

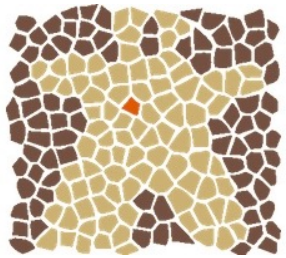
Pregel – Fault Tolerance

universität koblenz
weiter:denken

- At the beginning of some super-steps the master instructs the workers to save the state of their partitions to persistent storage

- If a worker is corrupt, the master reassigns graph partitions to the workers being alive; they reload their partition state from the most recently available checkpoint

Giraph



universität koblenz
weiter:denken

- Open-source implementation of Pregel, started by Yahoo, now worked on by developers from Facebook, LinkedIn, Twitter, etc.
- **MAP phase** to run computations
- Zookeeper to ensure waiting
- Classic M/R for dense graphs will require to spend most of the time moving data across nodes in the cluster
  - Heuristics: e.g. webpages from the same domain to the same MAP and use combiners

http://giraph.apache.org/

Who uses Giraph?

universität koblenz
weiter:denken

One Trillion Edges: Graph Processing at Facebook-Scale

Avery Ching
Facebook
1 Hacker Lane
Menlo Park, California
aching@fb.com

Sergey Edunov
Facebook
1 Hacker Lane
Menlo Park, California
edunov@fb.com

Maja Kabiljo
Facebook
1 Hacker Lane
Menlo Park, California
majakabiljo@fb.com

Dionysios Logothetis
Facebook
1 Hacker Lane
Menlo Park, California
dionysios@fb.com

Sambavi Muthukrishnan
Facebook
1 Hacker Lane
Menlo Park, California
sambavim@fb.com

ABSTRACT

Analyzing large graphs provides valuable insights for social networking and web companies in content ranking and recommendations. While numerous graph processing systems have been developed and evaluated on available benchmark graphs of up to 6.6B edges, they often face significant difficulties in scaling to much larger graphs. Industry graphs can be two orders of magnitude larger - hundreds of billions or up to one trillion edges. In addition to scalability challenges, real world applications often require much more complex graph processing workflows than previously evaluated. In this paper, we describe the usability, performance, and scalability improvements we made to Apache Giraph, an open-source graph processing system, in order to use it on Facebook-scale graphs of up to one trillion edges. We also describe several key extensions to the original Pregel model that make it possible to develop a broader range of production graph applications and workflows as well as improve code reuse. Finally, we report on real-world operations as well as performance characteristics of several large-scale production applications.

a project to run Facebook-scale graph applications in the summer of 2012 and is still the case today.

Table 1: Popular benchmark graphs.

Graph	Vertices	Edges
LiveJournal [9]	4.8M	69M
Twitter 2010 [31]	42M	1.5B
UK web graph 2007 [10]	109M	3.7B
Yahoo web [8]	1.4B	6.6B

Many specialized graph processing frameworks (e.g. [20, 21, 32, 44]) have been developed to run on web and social graphs such as those shown in Table 1. Unfortunately, real world social network are orders of magnitude larger. Twitter has 288M monthly active users as of 3/2015 and an estimated average of 208 followers per user [4] for an estimated total of 60B followers (edges). Facebook has 1.39B active users as of 12/2014 with more than 400B edges. Many of the performance and scalability bottlenecks are very different when considering real-world industry workloads. Several studies [14, 24] have documented that many graph frameworks fail at much smaller scale mostly due to inefficient

Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. 2015. One trillion edges: graph processing at Facebook-scale. *Proc. VLDB Endow.* 8, 12 (August 2015), 1804–1815.

Giraph

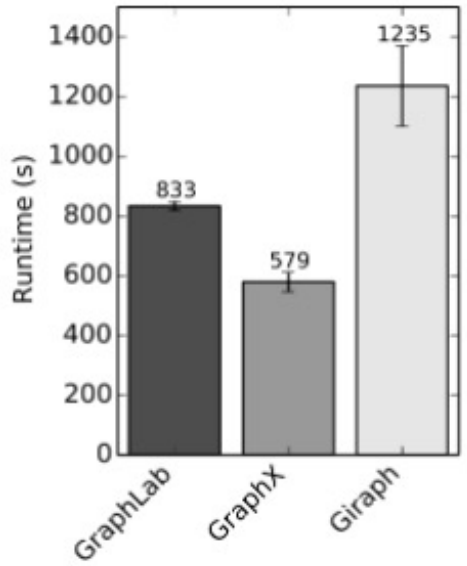
universität koblenz
weiter:denken

- Built on top of Hadoop
- Only MAP (no REDUCE -> no moving data)
  - Operation made on each node
  - Node-centric computation
- Input graph loaded only once
  - Input: nodes, edges with scores attached
- Computation in memory
  - receives messages sent to the node
  - Compute function using messages, node value, outgoing edge values
  - Send messages to other nodes (read at the next step)

Spark GraphX

universität koblenz
weiter:denken

- Spark's API for graphs and graph-parallel computation
- Implements Pregel operations
- Keeps data in memory
  - Fast(er)
  - Uses RDDs



End-to-end PageRank performance (20 iterations, 3.7B edges)

https://spark.apache.org/graphx/

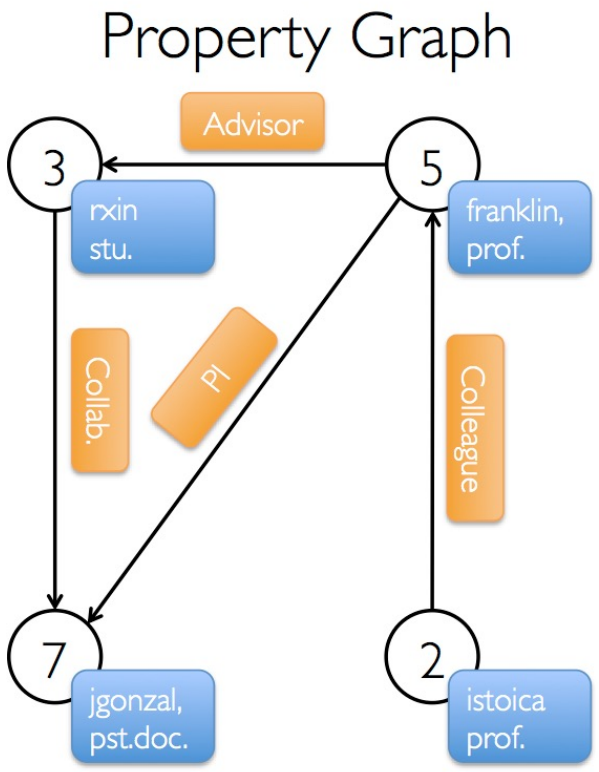
Who uses GraphX?

universität koblenz
weiter:denken

- Alibaba Taobao
- ActNow – a big data real-time predictive analytics platform
- AsiaInfo – Data centre solutions for customers in telco industry
- PanTera – Tool of exploring large data sets

https://spark.apache.org/powered-by.html

A graph in GraphX



Vertex Table

Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

Edge Table

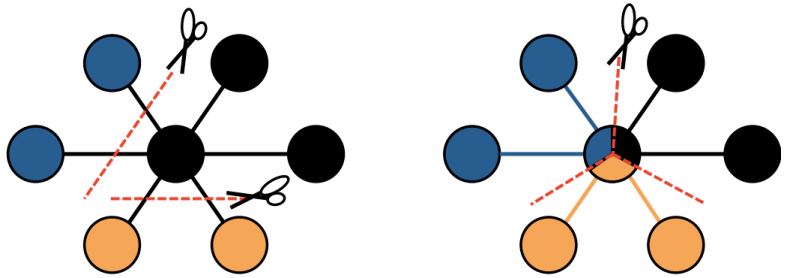
SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI

https://spark.apache.org/docs/latest/graphx-programming-guide.html

Spark GraphX

universität koblenz
weiter:denken

- GraphX adopts a vertex-cut approach to distributed graph partitioning



- Assigning edges to machines and allowing vertices to span multiple machines
  - More edges than vertices
- Vertices in GraphX can only send messages to neighboring vertices

Summary

universität koblenz
weiter:denken

- Network Theory
- Data representation
- Graph Processing Examples
- Distributed Systems for Graph Processing
  - Pregel
  - Giraph
  - Spark GraphX

Next week – Link Analysis at Scale

universität koblenz
weiter:denken

- Single-source Shortest Path
- PageRank
- Community detection