Overview of the Lecture

Part 1: Information retrieval

- Natural language as mechanism to share information

Part 2: Mining unstructured data

- Inferring structure from data aggregated from distributed sources
- Social graph mining (clustering), Document classification, Recommender systems (prediction), Association rule mining

Part 3: Knowledge graphs

- Creating and using shared formal models

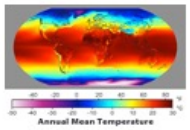
This summarizes the contents and the underlying conceptual framework of the lecture.

Part 2: Mining Unstructured Data

Overview

1. Data Mining
2. Mining Social Graphs
3. Document Classification
4. Recommender Systems
5. Association Rules

Information Management Tasks



model M

Model Usage: given a model, find some specific data

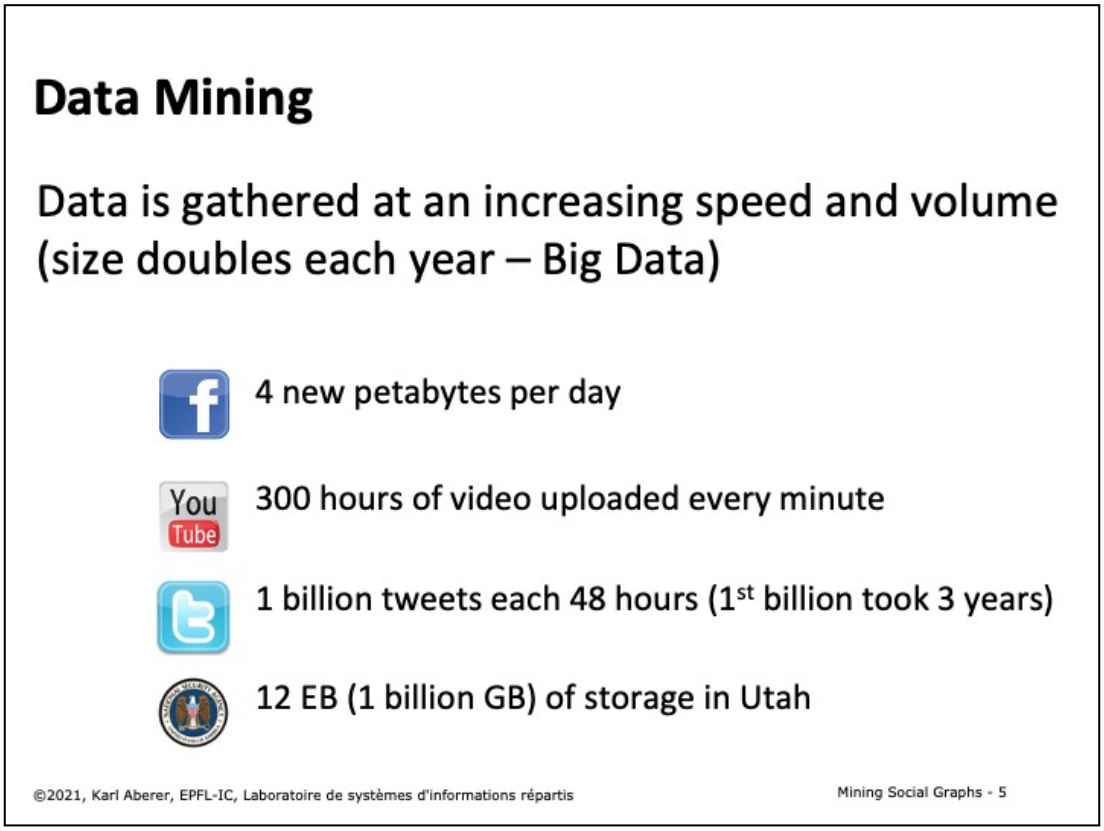
Example:
Temperature model allows to find temperature at selected locations

Model Building: given data, find a model the matches the data

Example:
Given temperature measurements, find a function that interpolates for all locations

data

Since the central role of an information system is to create a model of reality based on data, the key information management tasks are related to the interplay between data and models. We can identify two directions for this interplay: from models to data, and from data to models.



The growing use of information systems in many domains produces rapidly growing data collections in business, science and on the Web. Only recently the rate at which digital data is produced started exceeding the rate at which storage space is growing.

These data collections contain a wealth of hidden information, that needs to be discovered. Businesses can learn from their transaction data more about the behavior of their customers and therefore can become more efficient by exploiting this knowledge. Science can obtain from observational data (e.g. satellite data, sensor data) new insights on scientific problems. Web usage data can be analyzed and used to optimize information access, but as well to implement novel business models, such as for advertising, on the Web. The task of extracting useful information from large datasets is called data mining (or data analytics) and has in the recent years become one of the most dynamically evolving areas in computer science in general.

Data Mining

Most of this data is useless


pictures of drunk people

videos of cats


royal baby, Justin Bieber


SMS to girlfriend

Having increasing amounts of data does not imply having more useful information.

Data Mining Challenge

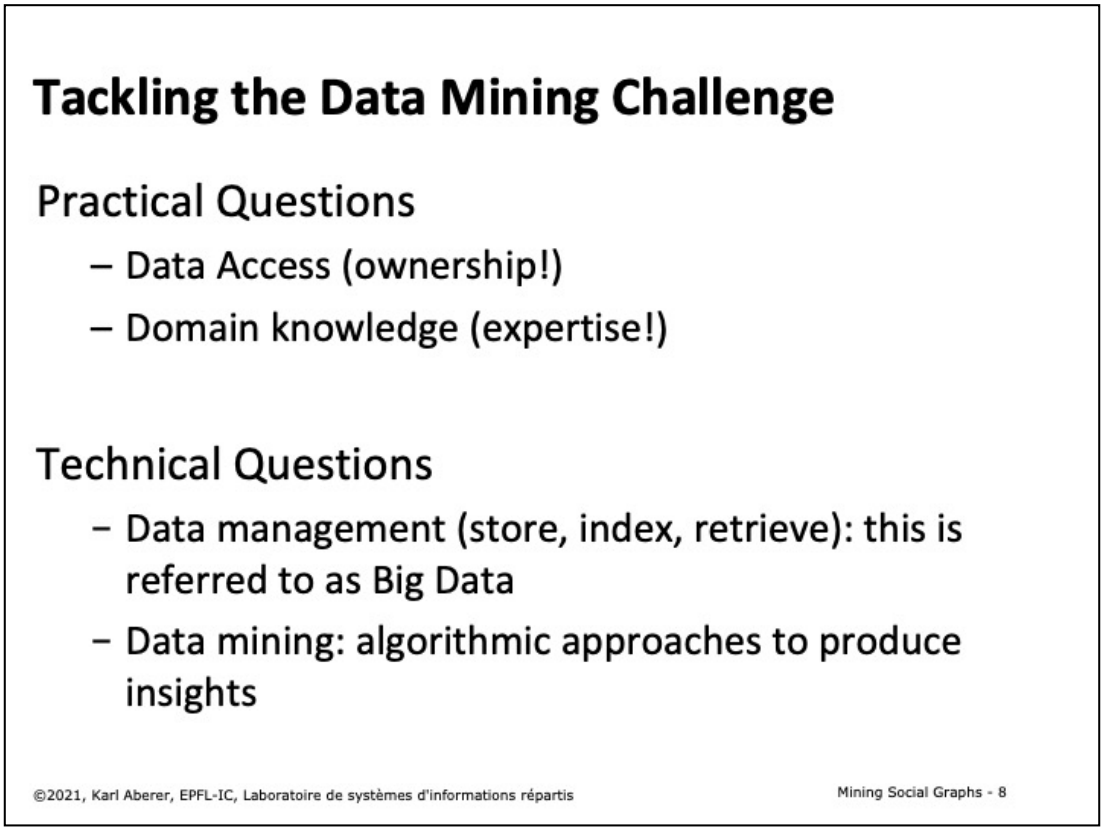
Transform masses of data into actionable intelligence

- From transaction data to market insights
- From observational data (satellite, sensors) to new scientific hypothesis
- From web usage data to better user interfaces
- ...

Analysis of large data sets to find relationships and to summarize the data in novel ways that are both understandable and useful

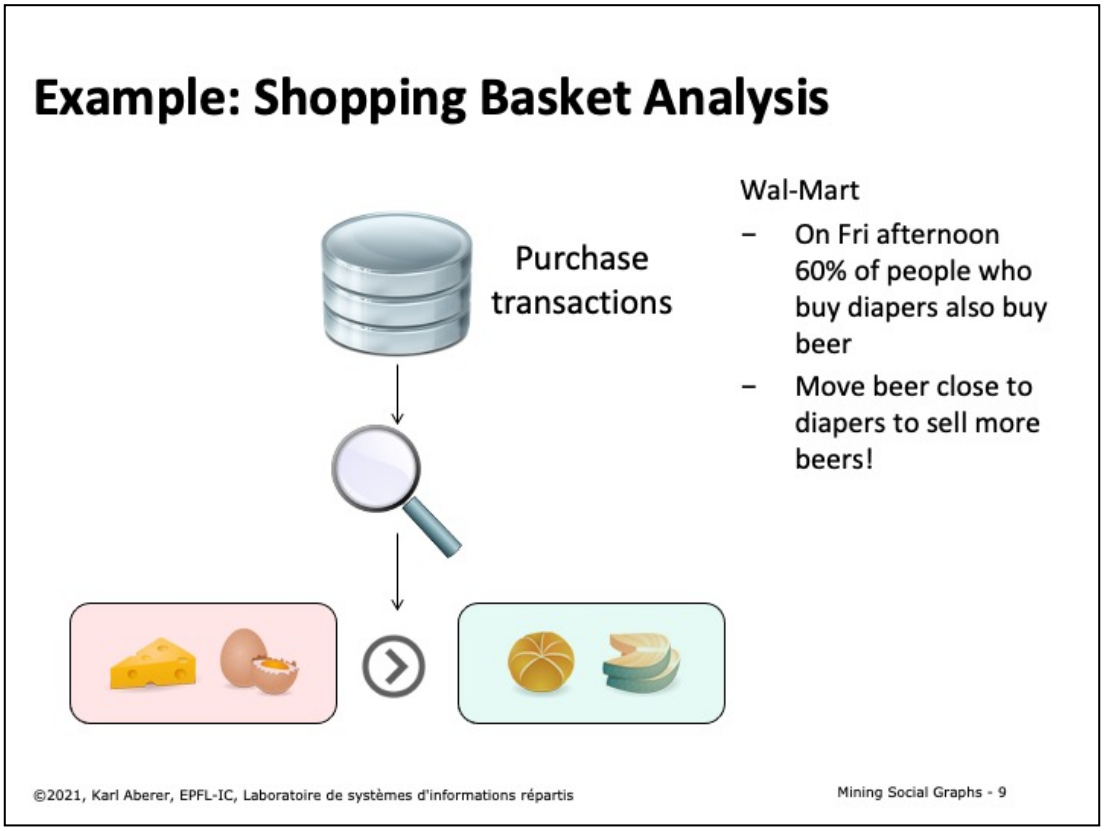
The real challenge in data analytics is extracting masses of data into what is usually called insights or actionable insights. Actionable is related to different aspects;

- understandable: the insights need to be interpreted by a human
- Useful: this implies that the insights help to take decision that have practical impact and utility; this also implies that insights that are not « surprising » but just reproduce existing knowledge are also not very useful.



The challenges to achieve this are manifold, but at least the following three key ingredients are required:

1. The data: of course massive amounts of data exist, but often they are not easily accessible, protected for legal, organizational, economic or political reasons, and spread out in many different systems.
2. The questions: searching for insights into data requires to have at least a general idea of what we are looking for, or what could be an interesting and useful insight. Without having an objective in mind it is hard to find answers.
3. The algorithms: Extracting interesting information out of data, requires efficient and smart algorithms. This is what we will study in the following. Once we master and understand such algorithms, the real challenges will be point 1 and 2.
4. Systems for handling Big Data: this is an area that has made huge progress in the recent years, in particular due to the needs of the big Internet companies. Many of the tools are now available as open source and within the cloud.



The classical example of a data mining problem is "market basket analysis". Retail stores gather information on what items are purchased by their customers. The expectation is, by finding out what products are frequently purchased together (i.e., are associated with each other), identifying ways to optimize the sales of the products by better targeting certain groups of customers (e.g., by planning the layout of a store or planning advertisements). A well-known example was the discovery that people who buy diapers also frequently buy beers (probably exhausted fathers of small children). Therefore nowadays one finds frequently beer close to diapers in supermarkets, and of course also chips close to beer. Similarly, amazon exploits this type of associations in order to propose to their customers books that are likely to match their interests. This type of problem was the starting point for one of the best known data mining techniques: association rule mining.

Other Examples

Amazon

Analysis of Query Logs (Query Expansion)

- Users that search for “Obama President” often also search for “Obama President Elections”

The original term of “market basket analysis” does not imply that this problem is limited to analyzing shopping behaviors. The same techniques can and have been applied in many other contexts, including recommender systems, search systems etc.

Classes of Data Mining Problems

Local properties

- Patterns that apply to part of the data
  - e.g., buy diapers ➔ buy beers

Global model

- Descriptive **structure** of the data
  - e.g., 3 types of customer behaviour
- Predictive **function** of the data
  - e.g., dist(beer, diapers)=1 ➔ +10% beer sales

Association rule mining is just one example of a data mining algorithm, out of a wide variety. Data mining algorithms can in general by classified according to the goals they pursue and the type of results they provide.

A basic distinction is made among data mining algorithm that identify global structures of a data sets, either in the form of summaries of the data or as globally applicable rules, and data mining algorithms that provide models that apply only locally, i.e., to some subset of the data set, in the form of sparsely occurring patterns or exceptional and unexpected dependencies in the data set.

The example of association rule mining is a typical case of discovering local patterns. The rules obtained are unexpected patterns and typically relate to small parts of the database only.

Data mining algorithms for finding global models of the data are further distinguished into techniques that are used to "simply" describe the data and into techniques that allow to make predictions on data that has not yet been seen. Descriptive modeling techniques provide compact summaries of the data sets, typically by identifying clusters of similar data items. Predictive modeling techniques provide globally applicable rules for the database, typically, allowing to predict some properties of the data, from some other data in the dataset.

Data Mining Overview

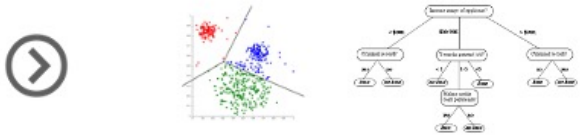
Local properties		Global model	
Patterns		Descriptive model	Predictive model
Association rules	Pattern mining	Clustering	Classification
		Information retrieval	Regression
Exploratory data analysis			
Interactive tools		Visualisation	

To complete the picture we can include into the field of data mining also exploratory data analysis, a special form of data mining which is used when no clear idea exists, what is being sought for in a given database. It may serve as a preprocessing step for performing more specific data mining tasks.

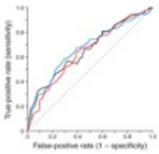
Information retrieval is usually also considered as a special case of data mining, where query patterns are searched for in the database. It can be understood as a clustering technique that distinguishes relevant from non-relevant data items.

Components of Data Mining Algorithms

1. Pattern structure/model representation
   - What we look for?



2. Scoring function
   - How well the model fits the data set?



Each data mining algorithm can be characterized by four aspects:

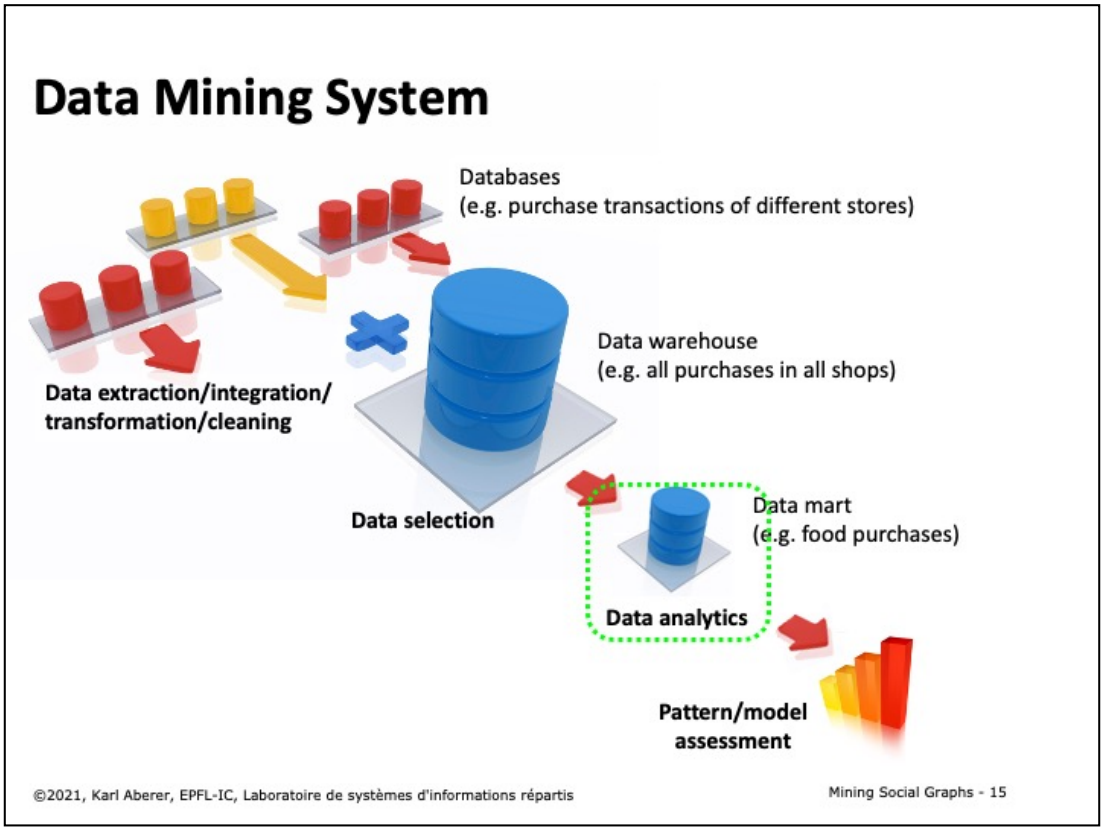
- The models or patterns that are used to describe what is searched for in the data set. Typical examples of models are dependencies, clusters and decision trees.
- The scoring functions that are used to determine how well a given data set fits the model. This is comparable to the similarity functions used in information retrieval.
- The method that is applied in order to find data in the data set that scores well with respect to the scoring function. Normally this requires efficient search algorithms that allow to identify those models that fit the data well according to the scoring functions.
- Finally the scalable implementation of the method for large data sets. Here indexing techniques and efficient secondary storage management are typically required. This corresponds to the use of inverted files in information retrieval.

Components of Data Mining Algorithms

3. Optimisation and search
   - How to tune the parameters of the model? [opt]
   - How to find data satisfying a pattern? [search]

4. Data management
   - How to implement the algorithm for very large data sets?

In particular the aspects of optimization and data management differentiate data mining from related areas such as statistics and machine learning: data mining algorithms can be understood as statistical or machine learning techniques that scale well to large data sets.



Data mining algorithms are part of larger data mining system that support the pre- an post-processing of the data. A data mining systems performs the following typical tasks:

- First, the data needs to be collected from the available data sources. Since these data sources can be distributed and heterogeneous databases, database integration techniques are applied. The integrated data is kept in so-called data warehouses, databases replicating and consolidating the data from the various data sources. An important concern when integrating the data sources is data cleaning, i.e., removing inconsistent and faulty data as far as possible. The tasks of data integration and data cleaning are supported by so-called data warehousing systems.
- Once the data is consolidated in a data warehouse, subsets of the data can be selected from the data warehouse for performing specific data mining tasks, i.e., tasks targeting a specific question. This task-specific data collections are called data-marts. The data-mart is the database to which the specific data mining algorithm is applied.
- The data mining task is the process of detecting interesting patterns in the data. This is what generally is understood as data mining in the narrow sense. We will introduce in the following examples of the most common techniques that are used to perform this task (e.g. association rule mining).
- Once specific patterns are detected they can be further processed. Further processing may include the evaluation of the "interestingness" of patterns for the specific problem at hand and the implementation of actions to react on the

discovery of a pattern.

Each of the steps described can influence the preceding steps. For example, patterns or outliers detected during data mining may indicate the presence of erroneous data, rather than interesting features in the source databases. This may imply adaptations of the data cleaning process during data integration.

Data Mining ≠ Machine Learning

What is in common

- In DM for data analytics frequently typical ML methods are used, though not always (e.g. visual mining, simple statistics)

What is different

- Data: DM is always applied to large datasets, ML not (e.g. reinforcement learning)
- Scope: DM comprises of the whole process of data integration, cleaning, analysis
- Goal: DM aims at detecting unsuspected patterns, ML may have other goals (e.g. winning a game)

Often data mining and machine learning are used like synonyms. Though both are exploiting often the same algorithmic techniques, they have different scope and goals. In particular, data mining is specifically targeted at analysis of large datasets and concerned with the resulting performance questions.

An insurance company wants to find typical, but unknown, causes for specific injuries from earlier cases. This likely is based on …

A. local rule discovery
B. predictive modelling
C. descriptive modelling
D. exploratory data analysis

2. MINING SOCIAL GRAPHS

Mining Graphs

Data Mining can be performed on different types of data

- Structured data: tables, graphs
- Unstructured data: text, images, sensor data

Graph data is increasingly important

- Social networks and Web
- Knowledge Graphs
- Scientific data on networks
- Graph structure can also be inferred from distance measures (e.g. for documents)

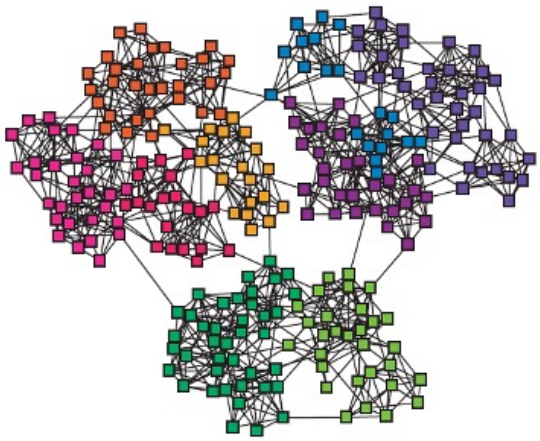
Data mining is applied for a wide range of both structured and unstructured data. Whereas it has traditionally evolved from analyzing large transactional datasets (e.g. from business), typically represented as relations, for structured data graph data is playing an increasingly important role, due to a growing number of graph data sources. One area where graph data play an obvious role is in social networks, where social interactions and relationships are modelled as graphs. This availability of graph data and the need to understand the structure of large graphs have been driving factors in the development and wide-spread application of graph mining algorithms.

It is also worthwhile to note that graph mining algorithms can be applied to graphs that are generated from other data types, e.g. by applying similarity measures on entities. For example, document similarity measures based on text embeddings could be used to generate graph structures for document collections.

Graphs and Clustering

Graphs often contain structure

– Clusters (also called communities, modules)

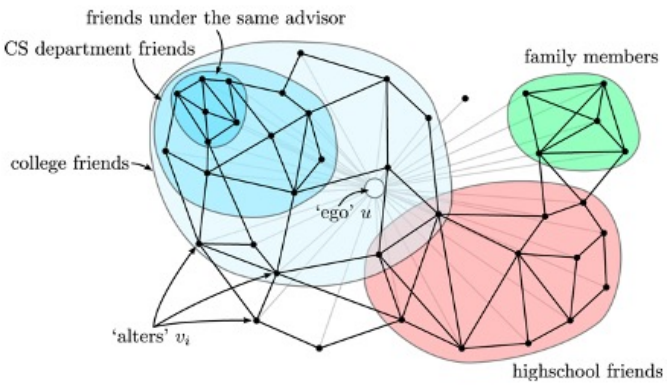


It is widely observed that natural networks contain structure. This is true for social networks (e.g. on social media platforms, citation networks), as well as for many natural networks as we find them in biology. Graph-based clustering aims at uncovering such hidden structures.

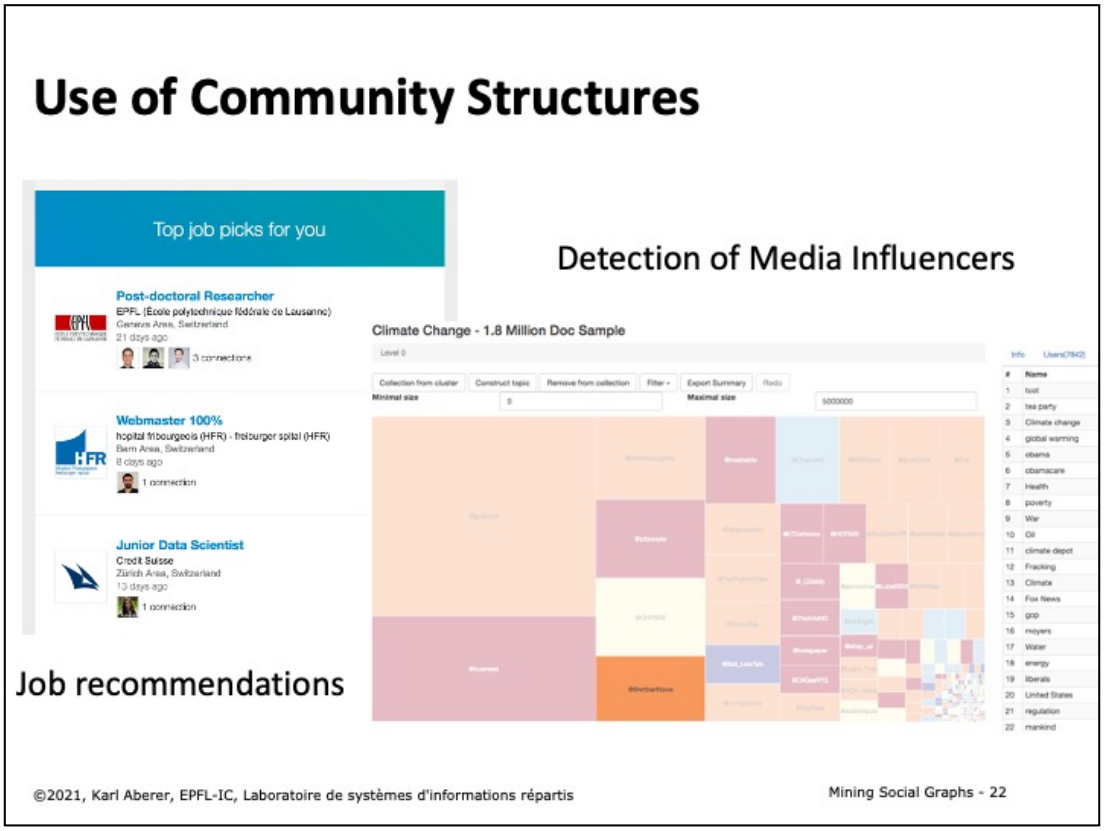
Social Network Analysis

Clusters (communities) in social networks (Twitter, Facebook) related to

- Interests
- Level of trust



In social networks mining community structures is particularly popular. Consider the social network neighborhood of a particular social network user, i.e. all other social network accounts that are connected to the user, either through explicit relationships, such as a follower graph, or through interactions such as likes or retweets. Typically the social neighborhood would decompose into different groups, depending on interests and social contexts. For example, the friends from high school would have a much higher propensity to connect to each other, than with members of the family of the user. Thus they are likely to form a cluster. Similarly, other groups with shared interest or high mutual level of trust would form communities.



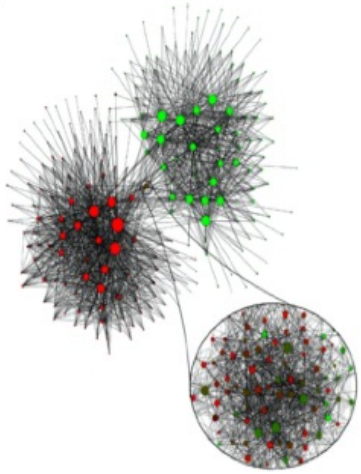
Many tasks can benefit from a reliable community detection algorithm. Online Social Networks rely on their underlying graph to recommend content, for example relevant jobs. Knowing to which community a user belongs can improve dramatically the quality of such recommendations.

Another typical use of community detection is to identify media influencers. Community detection can first be used to detect communities that share in social networks common interests or beliefs (e.g. for climate change we might easily distinguish communities that are climate deniers and climate change believers), and then the main influencers of such communities could be identified.

Use of Community Structures: Social Science

Call patterns in Belgium mobile phone network

– Two almost separate communities

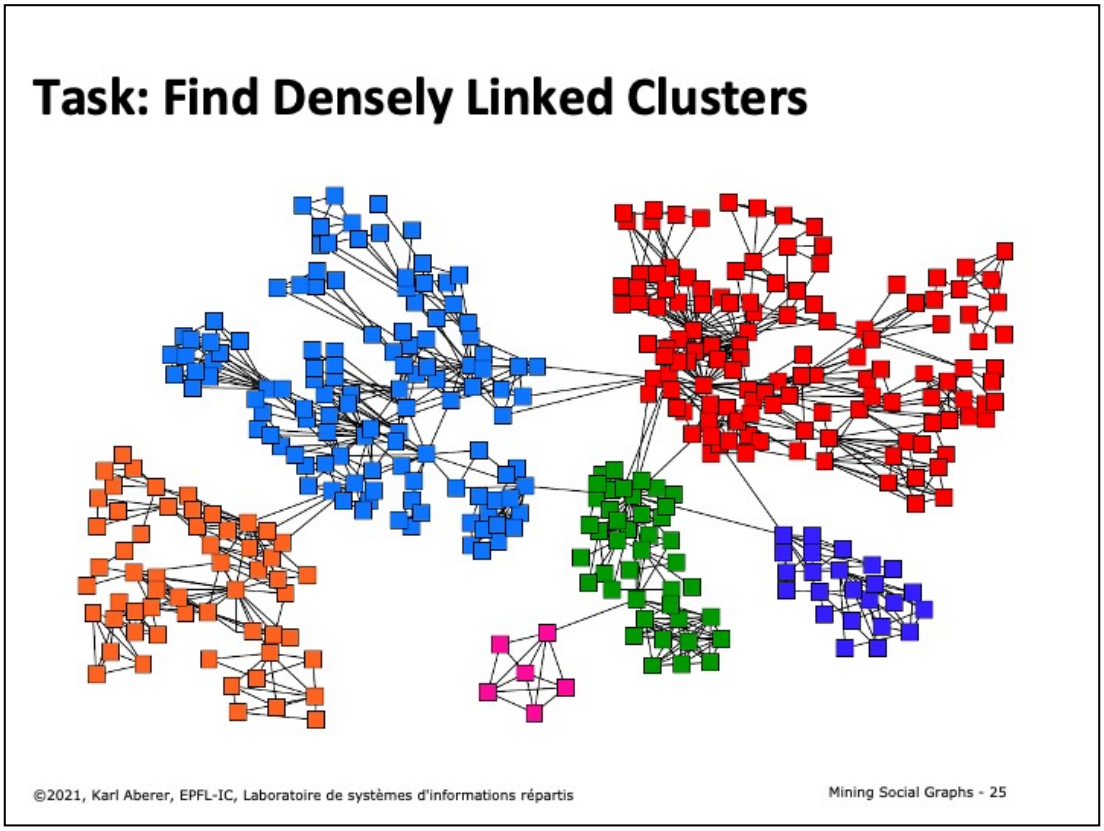


V.D. Blondel et al, *J. Stat. Mech.* P10008 (2008).

In 2008 Vincent Blondel and his students have started applying a new community detection algorithm to the call patterns of one of the largest mobile phone operators in Belgium. It was designed to identify groups of individuals who regularly talk with *each other* on the phone, breaking the whole country into numerous small and not so small communities by placing individuals next to their friends, family members, colleagues, neighbors, everyone whom they regularly called on their mobile phone. The result was somewhat unexpected: it indicated that Belgium is broken into two huge communities, each consisting of many smaller circles of friends. Within each of these two groups the communities had multiple links to each other. Yet, these communities never talked with the communities in the other group (guess why?). Between these two mega-groups was sandwiched in a third, much smaller group of communities, apparently mediating between the two parts of Belgium.

Which of the following graph analysis techniques do you believe would be most appropriate to identify communities on a social graph?

A. Cliques
B. Random Walks
C. Shortest Paths
D. Association rules



The intuition behind community detection is that the heavily linked components of the graph belong to the same community.

Similarly as earlier for clustering in multidimensional spaces, therefore, the goal of a community detection algorithm is to identify communities that are heavily intra-linked (high intra-cluster similarity) and scarcely inter-linked (low inter-cluster similarity).

Types of Community Detection Algorithms

Hierarchical clustering

- iteratively identifies groups of nodes with high similarity

Two strategies

- **Agglomerative algorithms** merge nodes and communities with high similarity
- **Divisive algorithms** split communities by removing links that connect nodes with low similarity

In general, community detection algorithms are based on a hierarchical approach. The idea is that within communities sub-communities can be identified, till the network decomposes into individual nodes. In order to produce such a hierarchical clustering, two approaches are possible: either by starting from individual nodes, by merging them into communities, and recursively merge communities into larger communities till no new communities can be formed (agglomerative algorithms), or by decomposing the network into communities, and recursively decompose communities till only individual nodes are left (divisive algorithms).

In the following we will present one representative of each of the two categories of algorihtms:

1. The Louvain Algorithm, an agglomerative algorithm
2. The Girvan-newman algorithm, a divisive algorithm

Louvain Modularity Algorithm

Agglomorative Community Detection

- Based on a measure for community quality (**Modularity**)
- greedy optimization of modularity

Overall algorithm

- **first** small communities are found by optimizing modularity **locally** on all nodes
- then each small community is **grouped** into one new community node
- **Repeat** till no more new communities are formed

The Louvain algorithm is essentially based on a the use of a measure, modularity, that allows to assess the quality of a community clustering. The algorithm performs greedy optimization of this measure. It is fairly straightforward: initially every node is considered as a community. The communities are traversed, and for each community it is tested whether by joining it to a neighboring community, the modularity of the clustering can be improved. This process is repeated till no new communities form anymore.

Measuring Community Quality

Communities are sets of nodes with many mutual connections, and much fewer connections to the outside

Modularity measures this quality: the higher the better

$$\sum_{C \in \text{Communities}} (\#\text{edges within } C - \text{expected } \#\text{edges within } C)$$

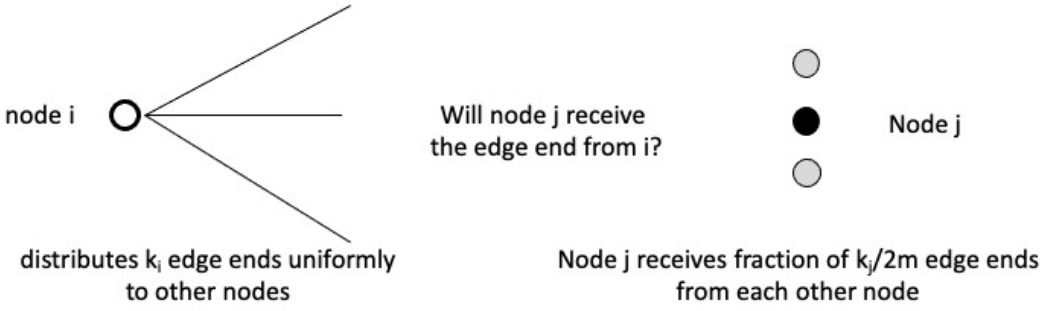
In order to measure the quality of a community clustering, modularity compares simply the difference between the number of edges that occur within a community with the number of edges that would be expected if the edges in the graph would occur randomly. The random graph model is used as what is called the null model, a graph that has the same distribution of node degrees, but randomly assigned connections.

Expected Number of Edges

Graph with unweighted edges

- m = total number of edges
- k_i = number of outgoing edges of node i (degree)

Observation: there exist 2m "edge ends"



distributes k_i edge ends uniformly to other nodes

Node j receives fraction of $k_j/2m$ edge ends from each other node

The null model requires to determining the number of edges that we would expect among a set of nodes, of which we know the degrees, but not the connectivity in detail. Assume that for all nodes i in a community we would know their degree ki (number of edges leaving the node). How many edges would we then observe if the connections on the graph were generated randomly? To answer this question we reason as follows: select one of the nodes i with degree ki. What is now the probability (or fraction of weight) an arbitrary other node j with weight kj would receive? If there are a total of m edges in the network, there are 2m edge ends (since each edge ends in two nodes). If the edge ends of ki are uniformly distributed, node j would thus receive kj/2m out of all edge ends. Thus it will receive a fraction ki*(kj/2m) fo all edge ends that are distributed from node ki.

Modularity

Modularity measure Q

- A_{ij} = effective number of edges between nodes i and j
- C_i, C_j = communities of nodes i and j

$$Q = \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(C_i, C_j)$$

Expected number of edges

Effective number of edges

Properties

- Q in [-1,1]
- 0.3-0.7 < Q means significant community structure

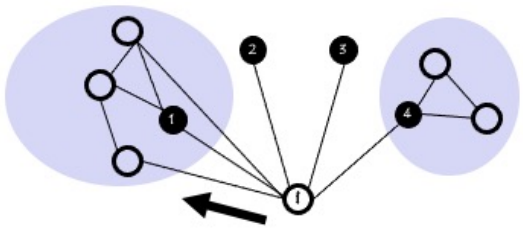
Given the expected number of edges (the null model) we can now formulate the modularity measure as the total difference of number of expected and effective edges for all pairs of nodes from the same cluster. The delta function assures that only nodes belonging to the same community are considered, since it returns 1 when ci = cj.

Due to normalization the measure returns calues between -1 and 1. In general, if modularity exceeds a certain threshold (0.3 to 0.7) the clustering of the network is considered to exhibit a good community structure.

Locally Optimizing Communities

What is the modularity gain by moving node i to the communities of any of its neighbors?

– Test all possibilities and choose the best

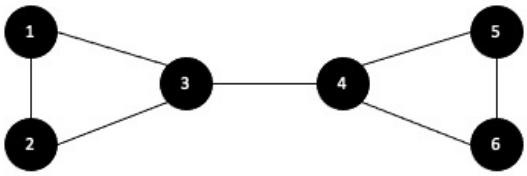


Given the modularity measure the next question is now how to use it to infer the communities. This is performed through local optimization. The algorithm sequentially traverses all nodes of the network, and for each node checks how the modularity can be increased maximally by having the node joining the node of a neighboring community. It then decides to join that node to the best community.

Example

Initial modularity Q = 0

Start processing nodes in order



We illustrate the algorithm for a simple example. Initially, the modularity is zero since all nodes belong to different communities. We start now to process the nodes in some given order, e.g. size of identifier.

Example: Processing Node 1

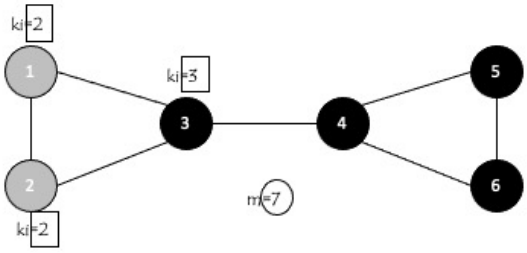
Joining node 1 to node 2

- Q = 1/2*7 (1 – 2*2/2*7) = 1/14 * 10 /14 > 0

Joining node 1 to node 3

- Q = 1/2*7 (1 – 2*3/2*7) = 1/14 * 8/14 > 0

New modularity: 1/14 * 10 /14



For node 1 we can join it either to node 2 or 3, it's two neighbors. To decide which is better, we have to compute modularity after the join. We see that it is better to join node 2, as the resulting modularity will be higher. We can think of this as follows: since node 3 has more connections, it is more likely to be randomly connected to node 1, this connecting to node 2 is more "surprising".

Example: Processing Nodes 2 and 3

Joining node 2 to node 3 (leaving community of node 1)

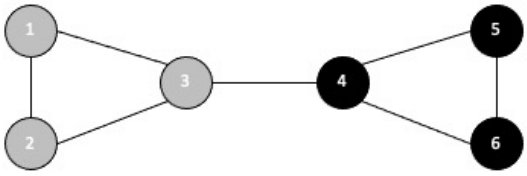
- no improvement

Joining node 3 to community {1,2} (via node 1 or 2)

- $Q = 1/14\ (3 - 4/14 - 6/14 - 6/14) = 1/14 * 26 / 14$

Joining node 3 to node 4

- $Q = 1/14\ 10/14 + 1/14\ (1 - 9/14) = 1/14 * 15/14$



For node 2 there will be no change, as the only alternative would be node 3. But for the same reasons node 1 did not join node 3, also node 2 does not. Next is node 3: here we have two choices, either to join community {1,2}, or to join node 4. In the first case we obtain a larger community, and in the second case two smaller communities. Computation of modularity reveals that joining 1 and 2 gives a much better community structure.

Example: Processing Nodes 4, 5 and 6

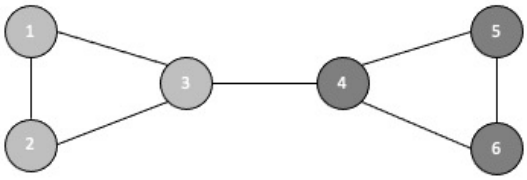
Joining node 4 to community {1,2,3} via 3

- Q = 1/14 (4 - 4/14 - 6/14 - 6/14 – 6/14 – 6/14 - 9/14) = 1/14 * 19/14

Joining node 4 to node 5

- Q = 1/14 * 26/14 + 1/14 * (1 – 6/14) = 1/14 * 34/14

Finally, also node 6 will join the second community



Similar arguments as before will then join node 4 to 5 and finally node 6 to the community consisting of nodes 4 and 5. Then the first round of processing is over.

Example: Merging Nodes

Now that all nodes have been processed, we merge nodes of the same community in a single new node and restart processing

Will the two remaining nodes merge? Answer: yes

For the next round of processing the resulting community nodes are collapsed in new nodes for the complete community, and the algorithm is re-run with the new resulting graph. Obviously, now the two remaining nodes will merge into a community, as the modularity will move from zero to positive. Then the algorithm terminates.

Modularity clustering will end up always with a single community at the top level?

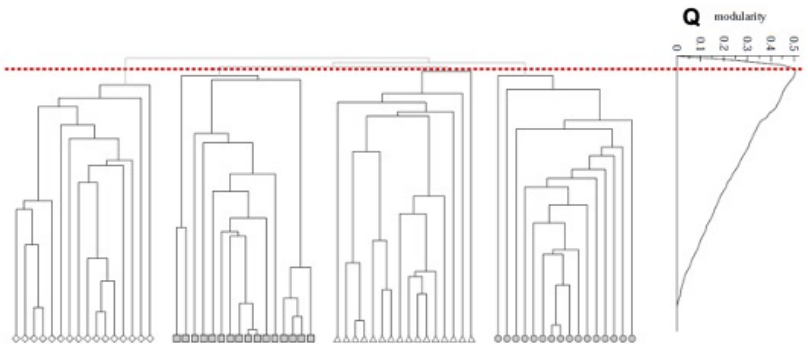
A. true
B. Only for dense graphs
C. Only for connected graphs
D. never

Modularity clustering will end up always with the same community structure?

A. true
B. Only for connected graphs
C. Only for cliques
D. false

Modularity to Evaluate Community Quality

Modularity can also be used to evaluate the best level to cutoff of a hierarchical clustering



Apart from constructing the communities, the modularity measure can also be used to evaluate the quality of communities in hierarchical clustering. This can be done independently of how the clustering has been constructed. In fact, there is an optimal level of clustering when moving from one level of the hierarchy to the next. Initially increasing the number of communities increases their quality, by separating distinct communities. At a certain point, splitting of communities in smaller communities worsens the quality of the community structure. Thus there exists an optimum point of clustering that can be selected using modularity.

Louvain Modularity - Discussion

Widely used in social network analysis and beyond

- Method to extract communities from very large networks very fast

Complexity: O(n log n)

Louvain modularity clustering is today the method of choice for social network clustering, mainly because of its good computational efficiency. It runs in n log n, which makes it applicable for very large networks, as they occur today, in particular for social networks resulting from large platforms, as social network sites, messaging services or telephony.

Girvan-Newman Algorithm

Divisive Community Detection

- Based on a **betweenness measure** for edges, measuring how well they separate communities
- Decomposition of network by splitting along edges with highest separation capacity

Overall algorithm

- Repeat until no edges are left
  - Calculate betweenness of edges
  - Remove edges with highest betweenness
  - Resulting connected components are communities
- Results in hierarchical decomposition of network

We now introduce a second algorithm fro community detection, that belongs to the class of divisive algorithms. Also this algorithm is based on a measure, this time on a measure on edges. The betweenness measure gives an indication which edges are likely to connect different communities, and thus are good splitting points, to partition larger parts of the network in to communities. The algorithm recursively decomposes the network, by removing edges with the highest betweenness measure, till no edges are left. Also this algorithm results in a hierarchical clustering.

Edge Betweenness

Edge betweenness: fraction of number of shortest paths passing over the edge

$$betweenness(v) = \sum_{x,y} \frac{\sigma_{xy}(v)}{\sigma_{xy}}$$

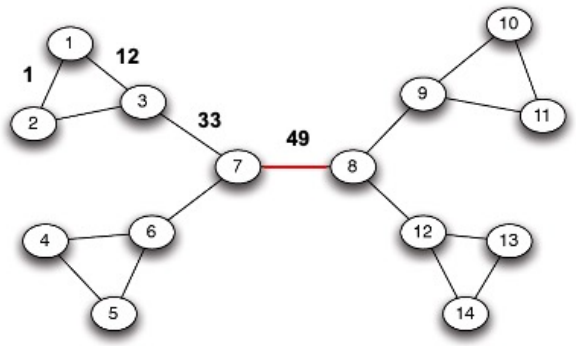
where

σ_{xy}: number of shortest paths from x to y
$\sigma_{xy}(v)$: number of shortest paths from x to y passing through v

Betweenness centrality is an indicator of a node's centrality in a network. It is equal to the number of shortest paths from all vertices to all others that pass through that node. A node with high betweenness centrality has a large influence on the transfer of items through the network, under the assumption that item transfer follows the shortest paths. The concept finds wide application, including computer and social networks, biology, transport and scientific cooperation.

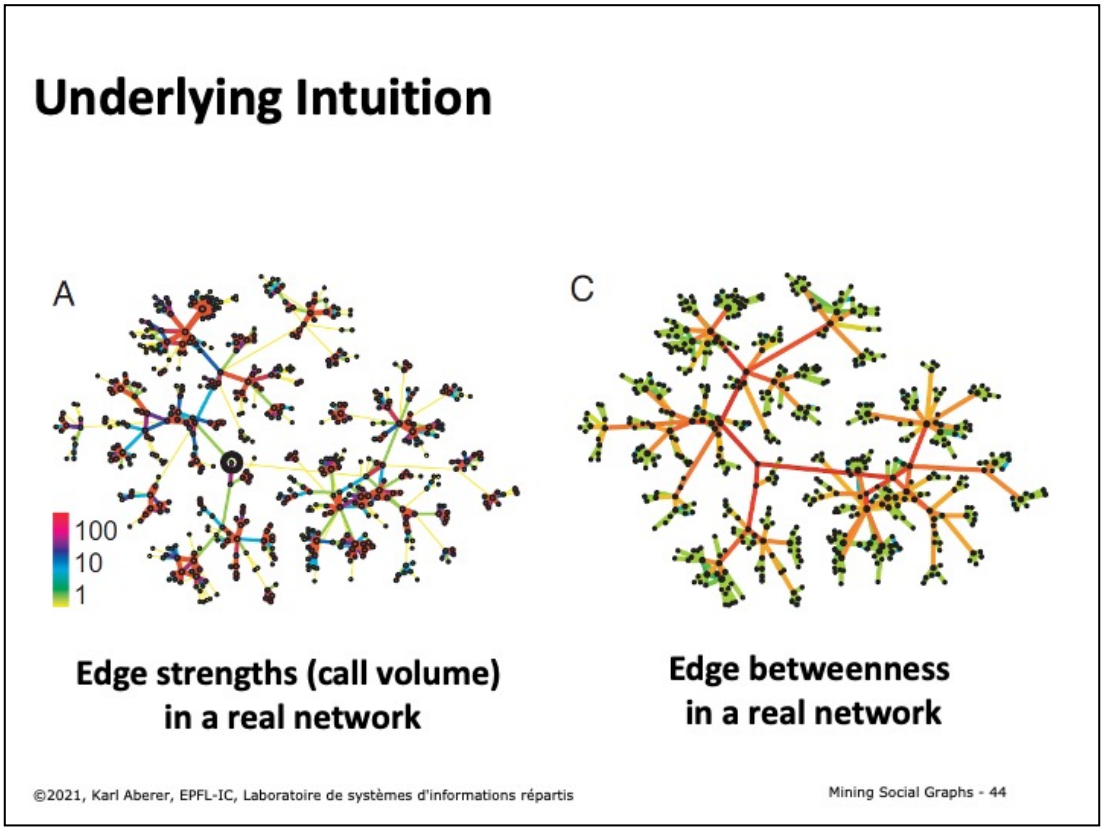
Alternatively, there exist also the concept of *random-walk betweenness*. A pair of nodes m and n are chosen at random. A walker starts at m, following each adjacent link with equal probability until it reaches n. Random walk betweenness x_{ij} is the probability that the link $i \rightarrow j$ was crossed by the walker after averaging over all possible choices for the starting nodes m and n

Example

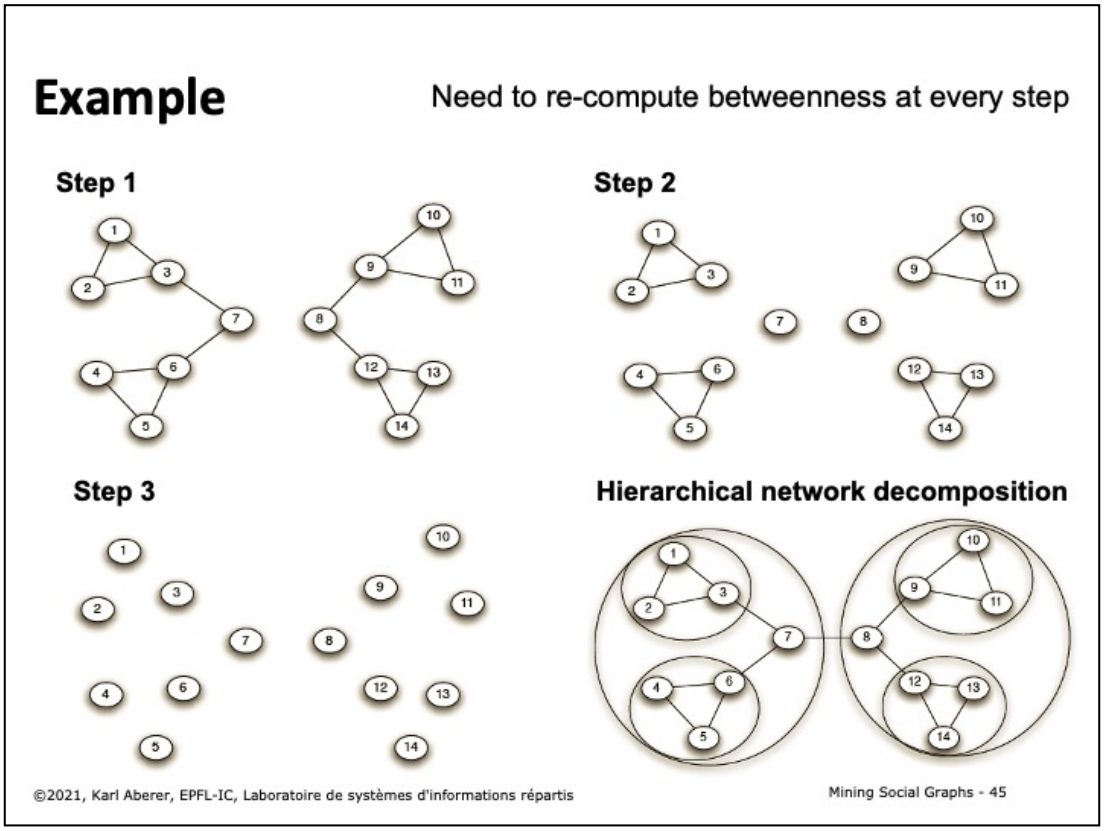


$$\sigma_{xy}(v)$$

In this example network we illustrate of how compute betweenness for some selected edges. This requires first to compute $\sigma_{xy}(v)$. For example, for the edge 1-2 there is one shortest path between 1 and 2 that traverses the edge 1-2, thus the value is 1. For 1-3 there are shortest paths from the remaining 12 nodes in the network (except node 2) to node 1 that have to pass through this edge, thus the betweenness value is 12. For edge 3-4 we have paths going to both nodes 1 and 2, thus the betweenness value of that edge is significantly higher than for 1-3.



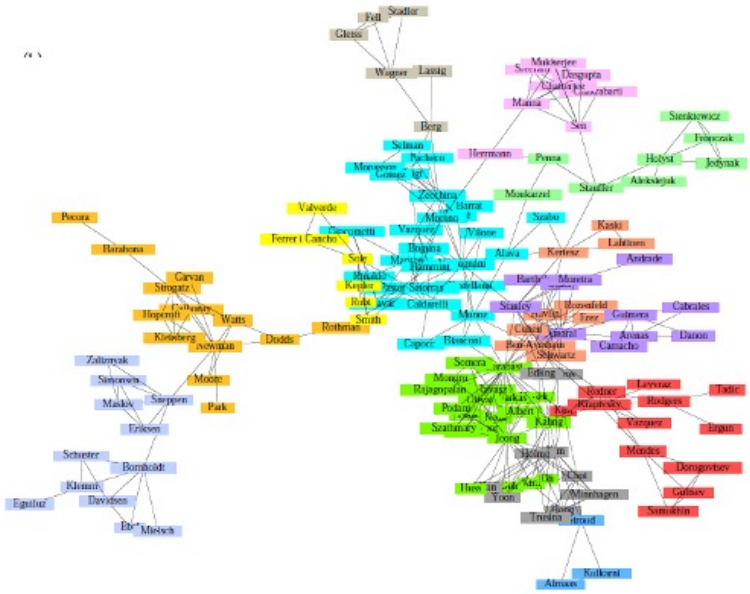
In a sense betweenness is the dual concept to connectivity. This is illustrated by the two graphs that result from real phone call networks. On the left hand side we see the indication of the strengths of connections among the nodes. Communities are tightly connected by such links. On the right hand side we see the betweenness measure. Now the links that are connecting communities have a high strength, since, intuitively speaking, the traffic from one community to another has to traverse over these sparsely available links.



Here we illustrate the execution of the Girvan-Newman algorithm. In Step 1 we remove one edge (in the middle) that had the highest betweenness value, resulting in two communities. Next (by symmetry) the edges connected to nodes 7 and 8 are removed, and in the third and fourth step the network decomposes completely. By overlaying the communities that have resulted from each step we obtain the final hierarchical clustering.

As the graph structure changes in every step, the betweenness values have to be recomputed in every step. This constitutes the main cost of the algorithm.

Girvan-Newman: Sample Results

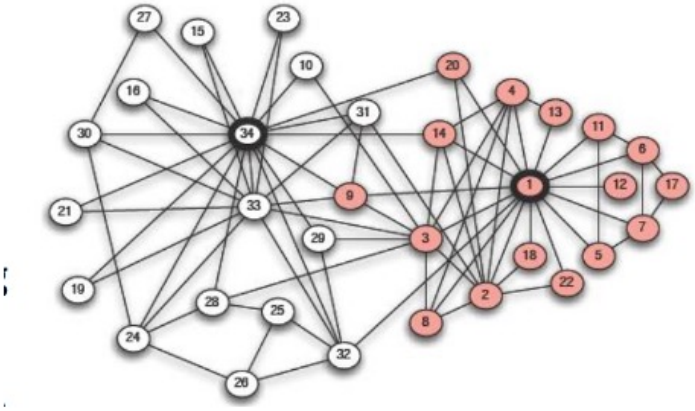


Communities in physics collaborations

The algorithm has been applied in many contexts, in particular on smaller graphs resulting from social science studies.

Girvan-Newman: Sample Results

Zachary's Karate club: Hierarchical decomposition

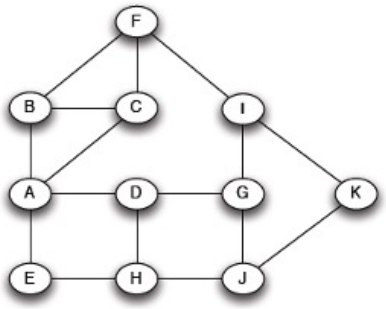


The origins of the algorithm come from studying a network of the 34 karate club members studied by the sociologist Wayne Zachary. Links capture interactions between the club members outside the club. The white and gray nodes denote the two fractions that emerged after the club split into two following a feud between the group's president and the coach. The split between the members closely follows the boundaries of these communities. This karate club has been historically used as a benchmark to test community finding algorithms.

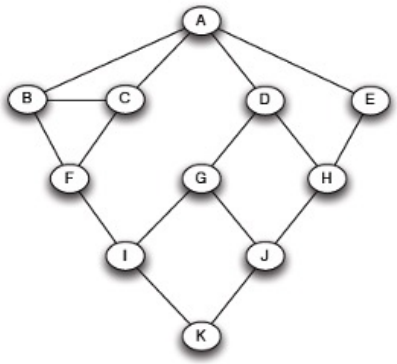
There was one outlier, node number 3, where the algorithm wrongly assigns the member. at the time of conflict, node 9 was completing a four-year quest to obtain a black belt, which he could only do with the instructor (node 34)

Computing Betweenness - BFS

Computing betweenness of paths starting at node A



Perform BFS starting from A

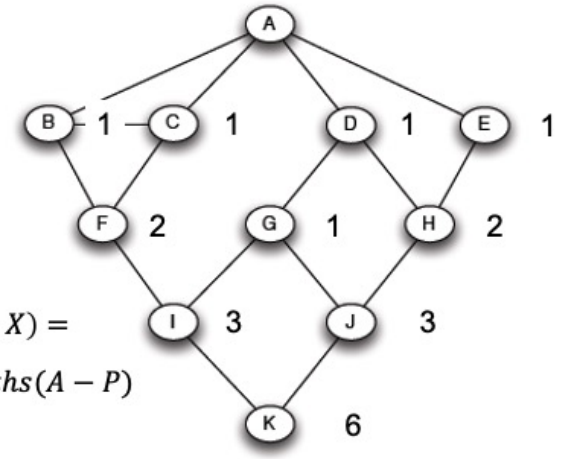


0
1
2
3
4

We describe now the process of computing the betweenness values. The approach is to perform a breadth-first search (BFS) for every node in the graph. The nodes are arranged in increasing levels of distance of the starting node, e.g. node A.

Computing Betweenness – Path Counting

Count the number of shortest paths from A to all other nodes of the network



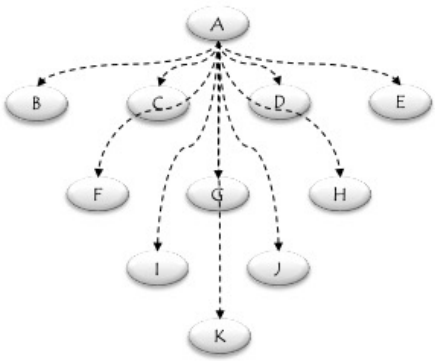
$$\#shortestpaths(A-X) = \sum_{P\ parent\ of\ X} \#shortestpaths(A-P)$$

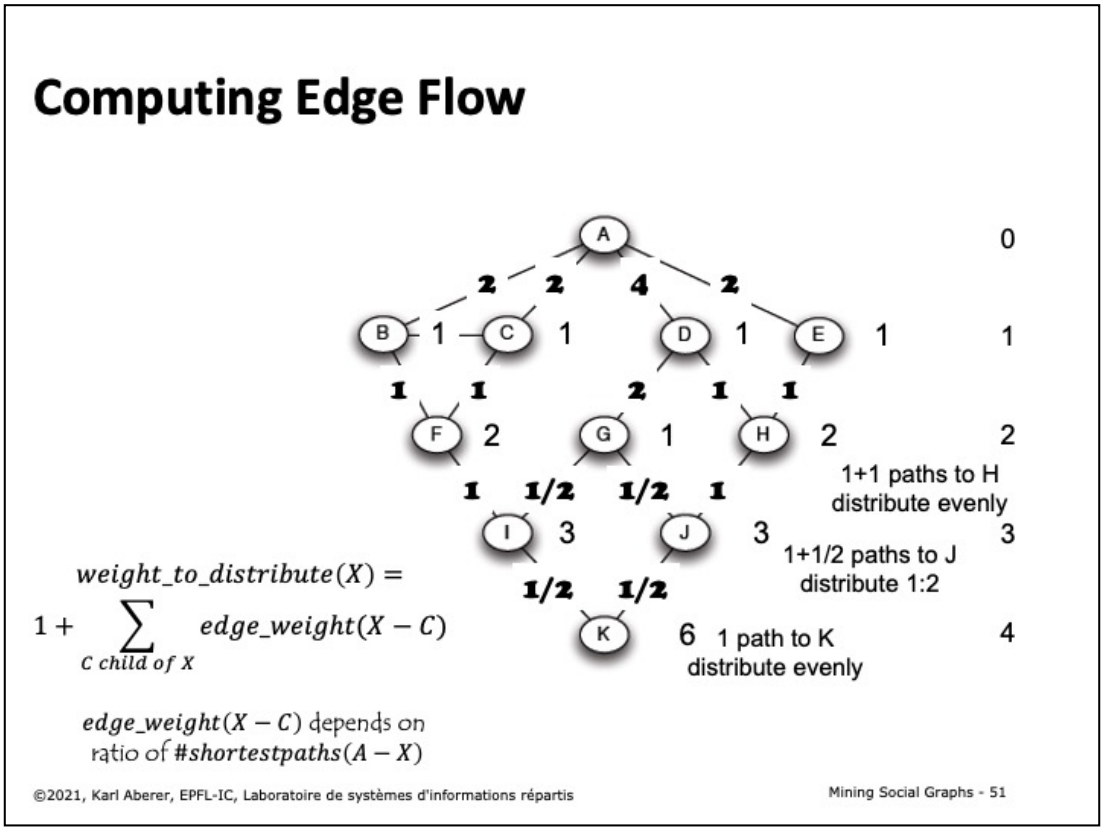
In a first phase we count the number of shortest paths that are leading to each node, starting from node A. To do so we can simply reuse the data that has been computed at the previous level, summing up the number of paths that have been leading to each parent of a given node.

Computing Betweenness – Edge Flow

Edge Flow

- 1 unit of flow from A to each node
- Flow to be distributed evenly over all paths
- Sum of the flows from all nodes equals the betweenness value





In a second phase we compute edge flow values (that are the betweenness values related to node A). We proceed in a bottom-up approach. The flow that arrives at every node is 1. In addition, the node receives also all the flows that are passing on to the children. Thus, the total flow weight a node receives (or has to pass through) is 1 plus the flows to all of its children. This weight is distributed over the parents, proportionally to the number of paths that are leading to those parents (what has been computed in phase 1).

This is a description of the initial steps in detail:

Node K: it receives a total flow of 1 from A. Since it has 2 incoming edges this flow is evenly distributed over the two incoming edges.

Node J: it receives a total flow of 1 from A plus transfers a flow of ½ to K. Thus the flow to distribute is: $1 + \sum_{C\ child\ of\ X} edgeweight(X - C)$ = 1.5. It's parent nodes G and H have a ratio of incoming shortest paths of 1:2, thus the flow is accordingly distributed over the incomgin edges, ½ and 1

Node H: it receives a total flow of 1 from A pls transfers a flow of 1 to J. This the flow do distributed is 2. The two parent nodes D and E have the same number of incoming shortest paths, this the flow is equally distributed.

The remaining steps proceed analogously.

Algorithm for Computing Betweenness

1. Build one BFS structure for each node
2. Determine edge flow values for each edge using the previous procedure
3. Sum up the flow values of each edge in all BFS structures to obtain betweenness value
   - Flows are computed between each pairs of nodes → final values divided by 2

Once the flows specific to a node have been computed for every node, the last step is to aggregate for each edge all the flow values that have been computed for all the nodes. In this way we compute each flow twice (flow into both direction), therefore, the final betweenness value corresponds to the aggregate flow value divided by 2.

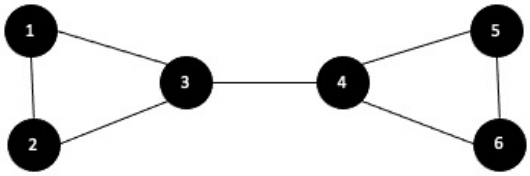
$\sigma_{xy}(v)$ of edge 3-4 is …

A. 16

B. 12

C. 9

D. 4



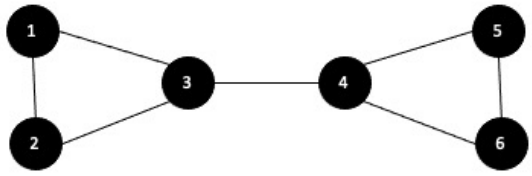
When computing path counts for node 1 with BFS, the count at 6 is …

A. 1

B. 2

C. 3

D. 4



Girvan-Newman Discussion

Classical method

- Works for smaller networks

Complexity

- Computation of betweenness for one link: $O(n^2)$
- Computation of betweenness for all links: $O(L\ n^2)$
- Sparse matrix: $O(n^3)$

The Girvan-Newman algorithm is the classical algorithm for community detection. It's major drawback is its scalability. The flow computation for one link has quadratic cost in the number of nodes (it has to be computed for each pair of nodes). If we assume sparse networks, where the number of links is of the same order as the number of nodes, the total cost is cubic. This was also one of the motivations the inspired the development of the modularity based community detection algorithm

References

The slides are loosely based also on:

- http://barabasilab.neu.edu/courses/phys5116/

Papers

- Blondel, Vincent D., et al. "Fast unfolding of communities in large networks." *Journal of statistical mechanics: theory and experiment* 2008.10 (2008): P10008
- Girvan, Michelle, and Mark EJ Newman. "Community structure in social and biological networks." *Proceedings of the national academy of sciences* 99.12 (2002): 7821-7826.