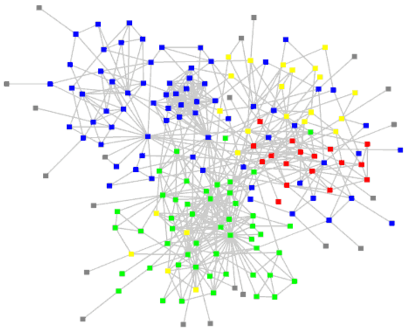
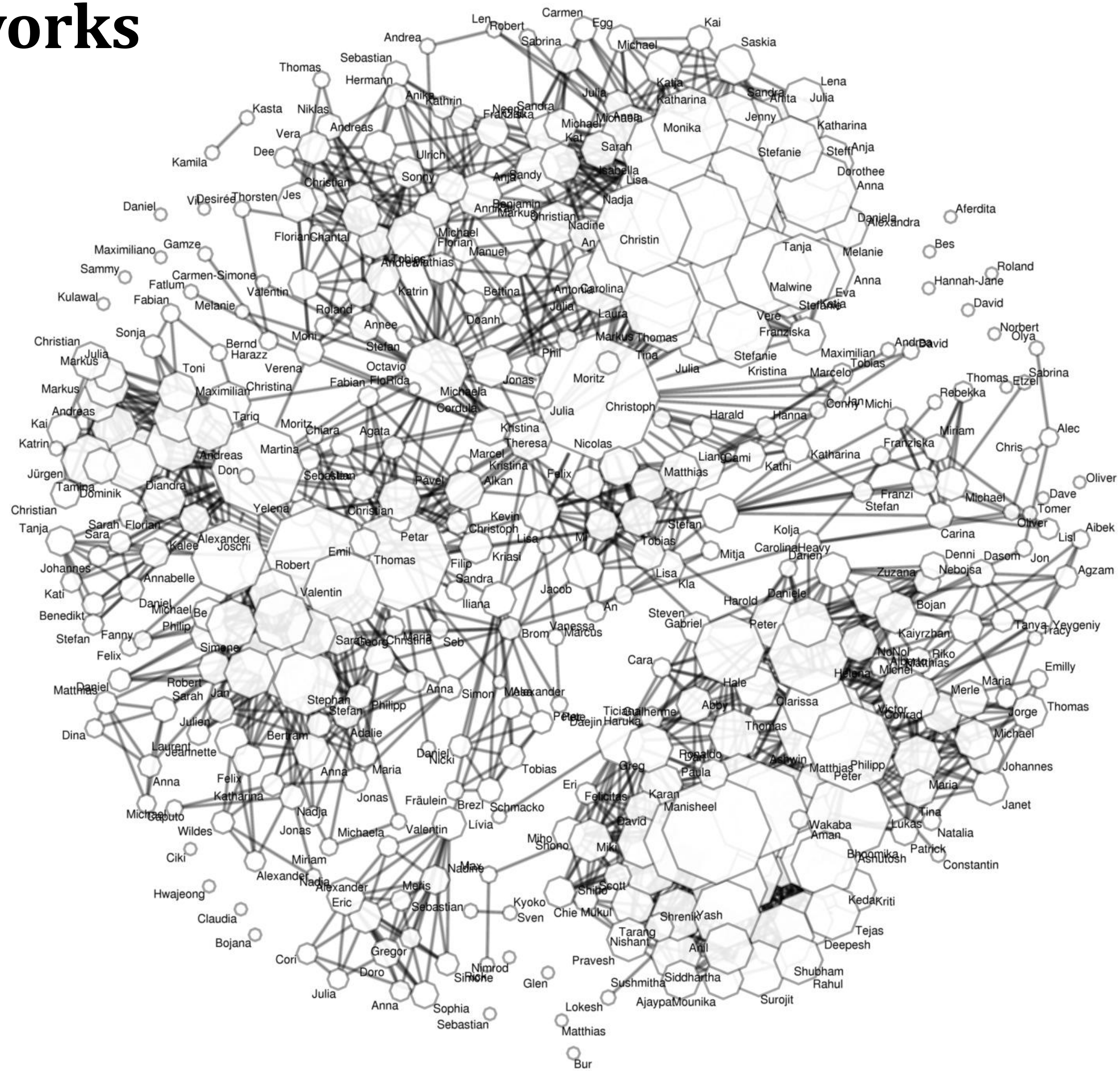


SCAN: A Structural Clustering Algorithm for Networks

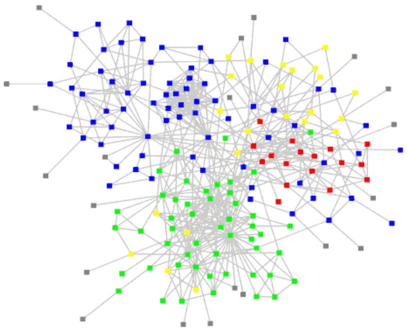
Xiaowei Xu, Nurcan Yuruk, Zhidan Feng (University of Arkansas at Little Rock)
Thomas A. J. Schweiger (Acxiom Corporation)

Networks

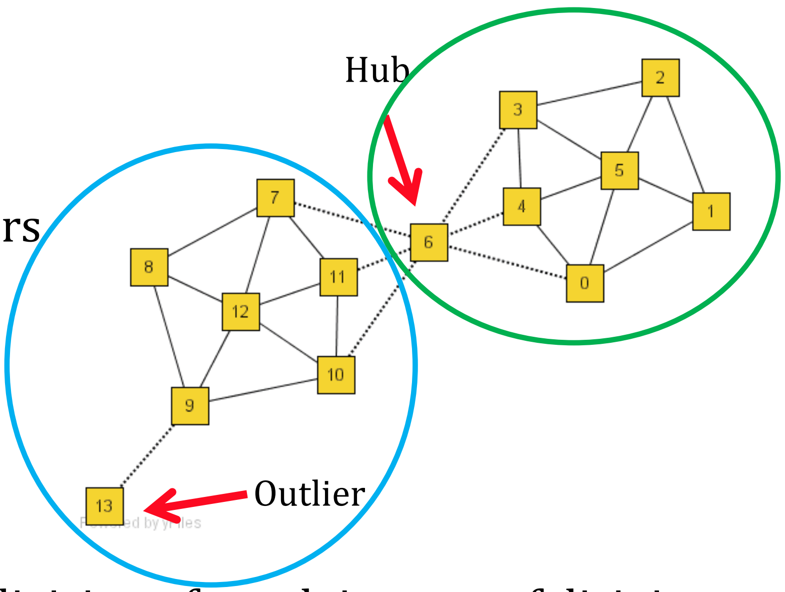


scaling: #edges connected to respective vertex

Network basics

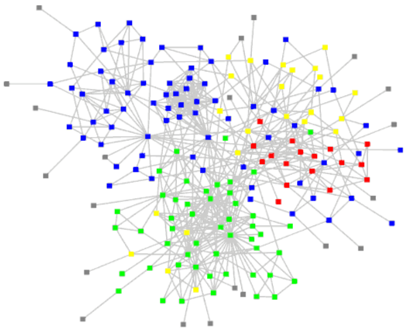


- Network graph *G = {V, E}*
- *V* is set of vertices, *E* is set of edges connecting the vertices



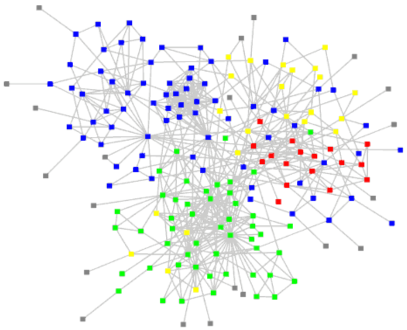
- **Hubs**: vertices that bridge (many) clusters
- **Outliers**: vertices marginally connected to clusters featuring a weak association
- network clustering/graph partitioning: division of graph into set of disjoint sub-graphs in order to find hidden structures
- existing approaches aim at finding clusters based on the number of edges between vertices or clusters

SCAN – A Structural Clustering Algorithm for Networks



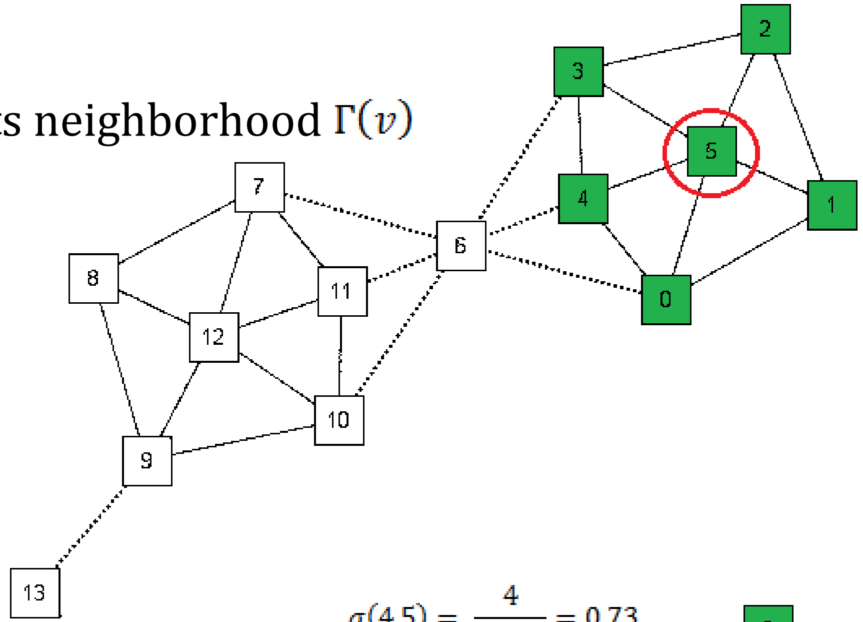
- based on definitions of DBSCAN
- detects clusters, hubs and outliers using structure and connectivity of the vertices as a clustering criterion
- vertices sharing a certain quantity of neighbors should be grouped into one cluster, hubs and outliers should be isolated
- runtime complexity wrt. n vertices and m edges in a network: $O(m)$
- focusses on simple, undirected and unweighted graphs

Definitions for structure-connected clusters (I)



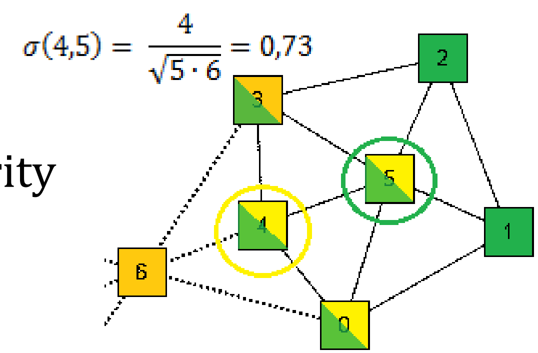
- **VERTEX STRUCTURE**
  $v \in V$, the structure of v is defined by its neighborhood $\Gamma(v)$

$$\Gamma(v) = \{w \in V | (v,w) \in E\} \cup \{v\}$$

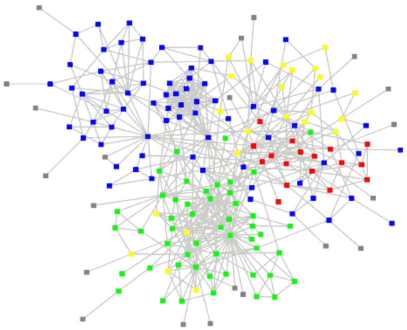


- **STRUCTURAL SIMILARITY**
  two objects sharing a similar structure in terms of their neighborbood, will have a large structural similarity

$$\sigma(v,w) = \left\{\frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}\right\}$$

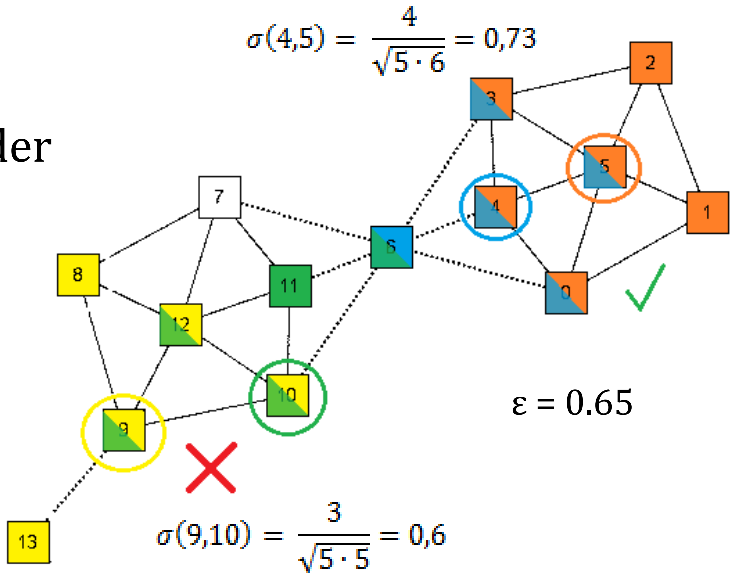


Definitions for structure-connected clusters (II)



- ***ε* – NEIGHBORHOOD**
  apply threshold to structural similarity in order to assign cluster memberships

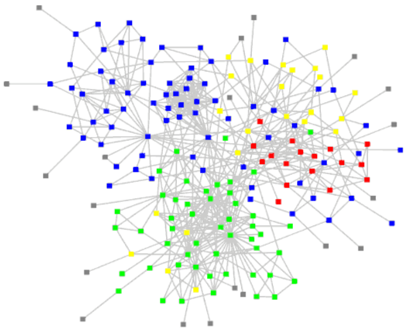
  $$N_{\varepsilon}(v) = \{w \in \Gamma(v) \mid \sigma(v,w) \geq \varepsilon\}$$



- **CORE VERTEX**
  core vertex shares *structural similiarity* of at least ε with at least μ neighbors

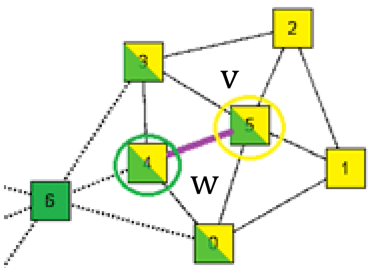
  $$Core_{\varepsilon,\mu}(v) \Leftrightarrow |N_{\varepsilon}(v)| \geq \mu$$

Definitions for structure-connected clusters (III)



- **DIRECT STRUCTURE REACHABILITY**
  if vertex is in ε – Neighborhood of a *core vertex*, they should be in the same cluster

  $$DirReach_{\varepsilon,\mu}(v,w) \Leftrightarrow Core_{\varepsilon,\mu}(v) \wedge w \in N_{\varepsilon}(v)$$

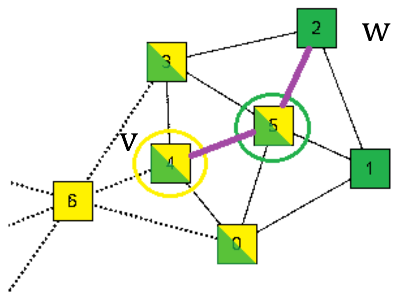


ε = 0.65,
μ = 2

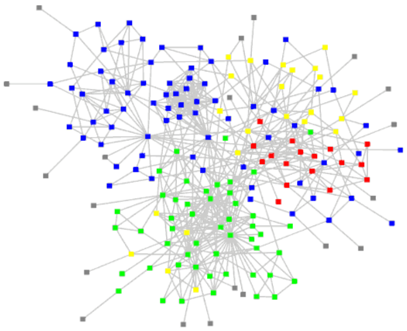
- **STRUCTURE REACHABILITY**
  a vertex $w \in V$ is *structure-reachable* from $v \in V$, if there is a chain of vertices $v_1, \ldots, v_n \in V, v_1 = v, v_n = w$ such that v_{i+1} is *directly structure-reachable* from v_i

  $$Reach_{\varepsilon,\mu}(v,w) \Leftrightarrow \exists v_1, \ldots, v_n \in V: v_1 = v \wedge v_n = w \wedge \forall i \in \{1, .., n\}: DirReach_{\varepsilon,\mu}(v_i, v_{i+1})$$

ε = 0.65,
μ = 2



Definitions for structure-connected clusters (IV)

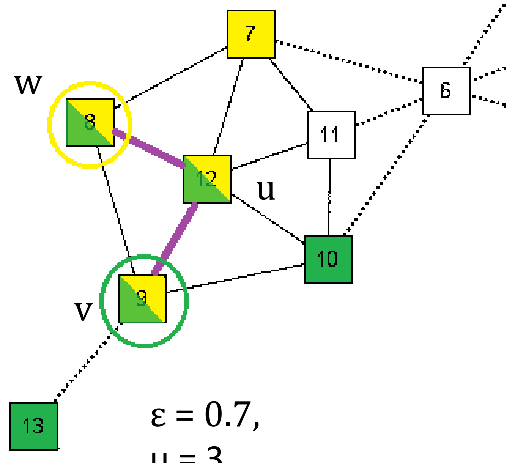


- **STRUCTURE CONNECTIVITY**

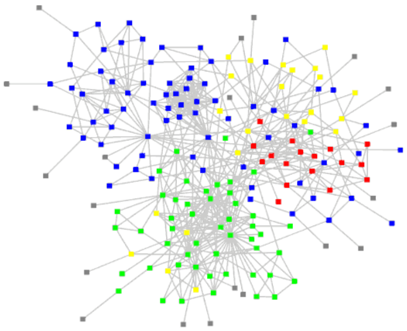
  two non-core vertices $v, w \in V$ belonging to the same cluster may not be *structure-reachable*, as the *core-condition* does not hold for them.

  However, they still belong to the same cluster, if they are *structure-reachable* from the same *core vertex* $u \in V$. This is called *structure-connectivity*.

$$Connect_{\varepsilon,\mu}(v,w) \Leftrightarrow \exists u \in V: Reach_{\varepsilon,\mu}(u,v) \wedge Reach_{\varepsilon,\mu}(u,w)$$



Definitions for structure-connected clusters (V)



- **STRUCTURE-CONNECTED CLUSTER**

  a non-empty subset $C \subseteq V$ is called a *structure-connected cluster* , if all vertices in C are *structure-connected* and C is maximal wrt. *structure reachability*

  $$Cluster_{\varepsilon,\mu}(C) \Leftrightarrow$$

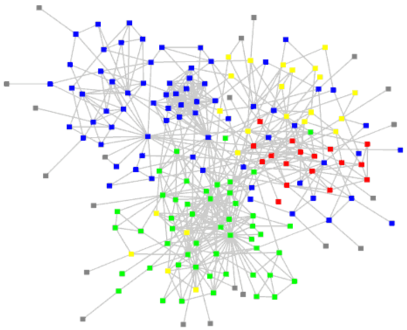
  1. Connectivity: $\forall v, w \in C: Connect_{\varepsilon,\mu}(v,w)$
  2. Maximality: $\forall v, w \in V: v \in C \wedge Reach_{\varepsilon,\mu}(v,w) \Rightarrow w \in C$

- **CLUSTERING**

  a *clustering P* consists of all *structure-connected clusters*

  $$Clustering_{\varepsilon,\mu}(P) \Leftrightarrow P = \{C \subseteq V \mid Cluster_{\varepsilon,\mu}(C)\}$$

Definitions for structure-connected clusters (VI)



If a vertex is not a member of any cluster, it is either a hub or an outlier, depending on its neighborhood.

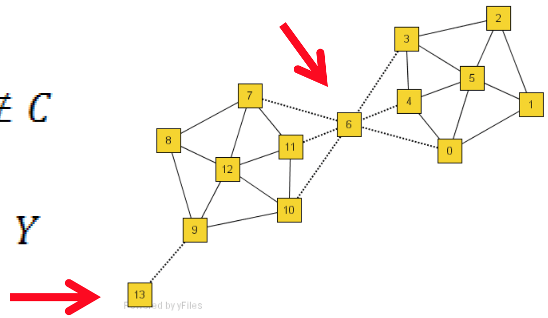
- **HUB**

  if an isolated vertex $v \in V$ has neighbors belonging to two or more different clusters, it is a *hub*

  $Hub_{\varepsilon,\mu}(v) \Leftrightarrow$

  1. v is not a member of any cluster: $\forall C \in P: v \notin C$
  2. v bridges different clusters:

     $\exists p, q \in \Gamma(v): \exists X, Y \in P: X \neq Y \wedge p \in X \wedge q \in Y$



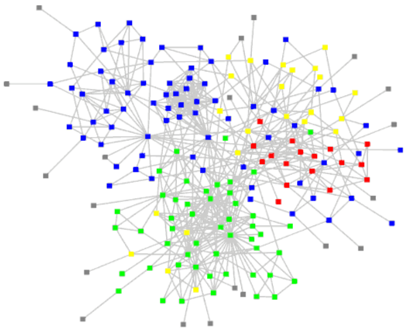
- **OUTLIER**

  an isolated vertex $v \in V$ is an *outlier,* if and only if all its neighbors either belong to only one cluster or do not belong to any cluster

  $Outlier_{\varepsilon,\mu}(v) \Leftrightarrow$

  1. v is not a member of any cluster: $\forall C \in P: v \notin C$
  2. v does not bridge different clusters: $\neg\exists p, q \in \Gamma(v): \exists X, Y \in P: X \neq Y \wedge p \in X \wedge q \in Y$

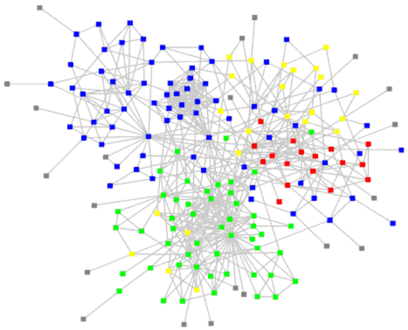
Algorithm SCAN



- for each vertex ***v*** not yet classified, it is checked whether vertex is a **core**
  - **if so**, a new cluster is expanded from this vertex:
  1. the algorithm generates a **new cluster id**, looks for **all unclassified vertices** in the **neighborhood** of the core and inserts them into a **queue**
     - *vertices which are directly structure-reachable from core vertex v*
  2. it traverses the queue until it is empty and establishes all vertices which are **directly structure-reachable** from the respective vertex **w**
     - *vertices which are structure-reachable from core vertex v (via vertex w)*
  3. the same cluster id is assigned to all those vertices and if they are not labeled as a non-member yet, they are also inserted into the queue
  - **if not**, it is labeled as a non-member
- non-member vertices which have edges to two or more clusters, are classified as hubs. Otherwise, they are classified as outliers.

- ε-value between 0.5 and 0.8, μ = 2

Experimental Evaluation (I)



Comparison to **FastModularity** algorithm:

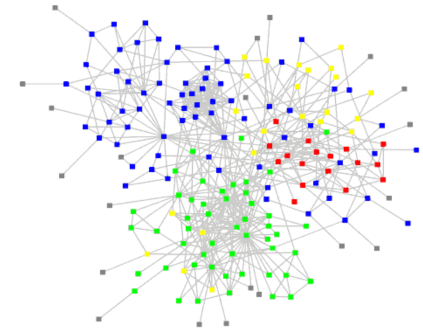
- hierarchical network clustering algorithm optimizing the modularity

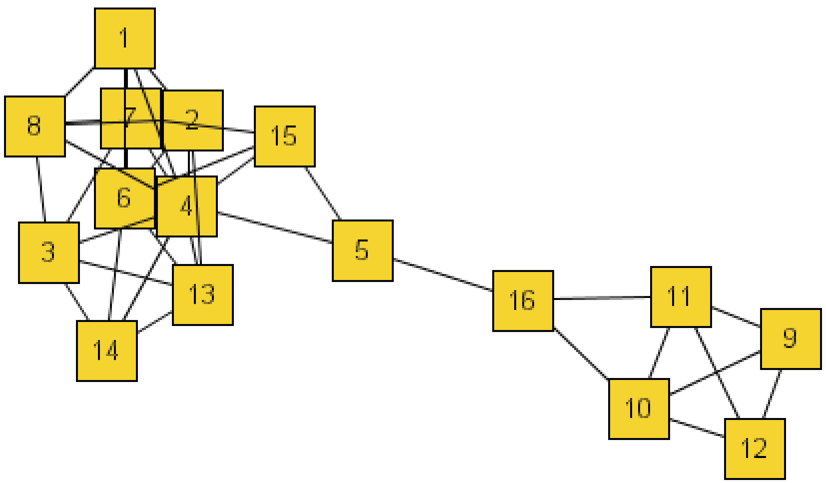
- *modularity* measures whether division of a network into communities is a good one in terms of many edges within communities and preferably little edges between communities

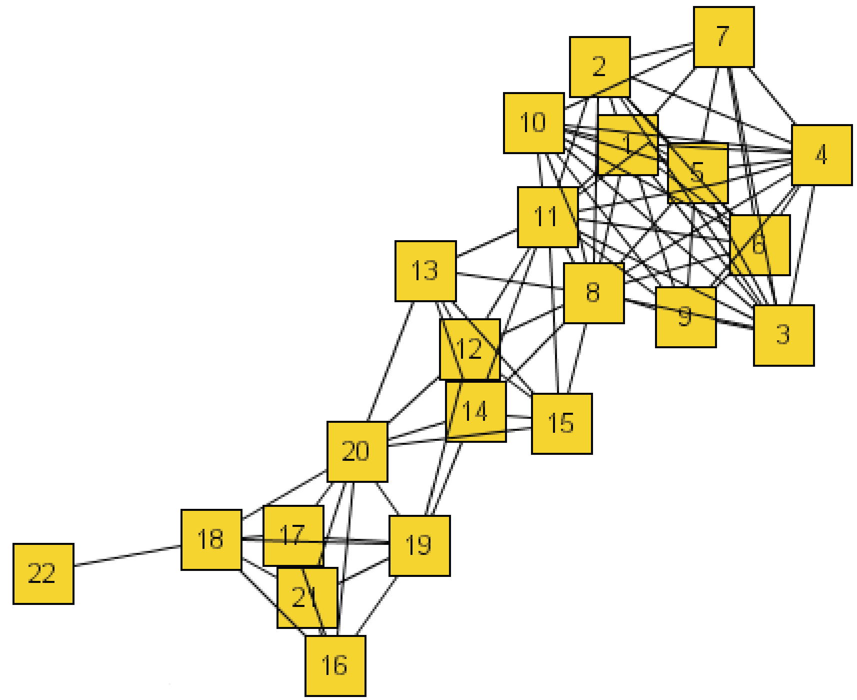
1. initally, each vertex is the only member of a community
2. iteratively, the algorithm greedily merges the two communities causing the largest increase of modularity, until all vertices are members of the same community

Experimental Evaluation (II)

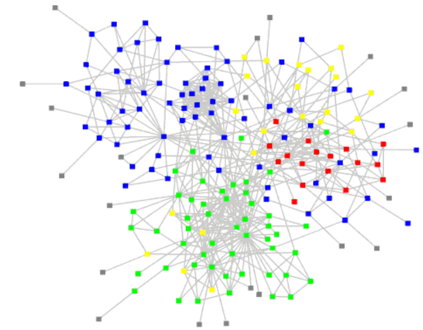
Customer Segmentation dataset

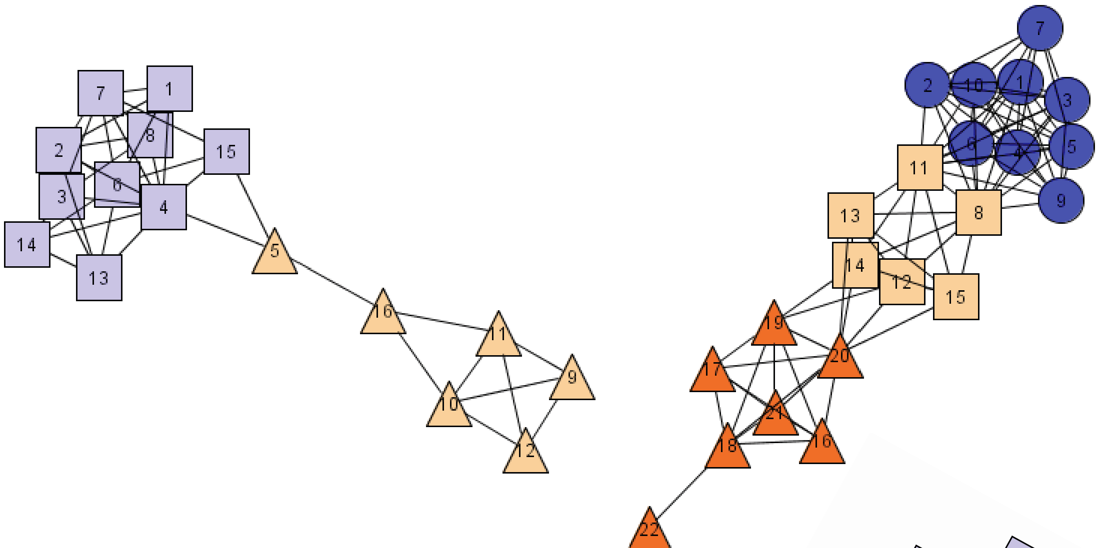




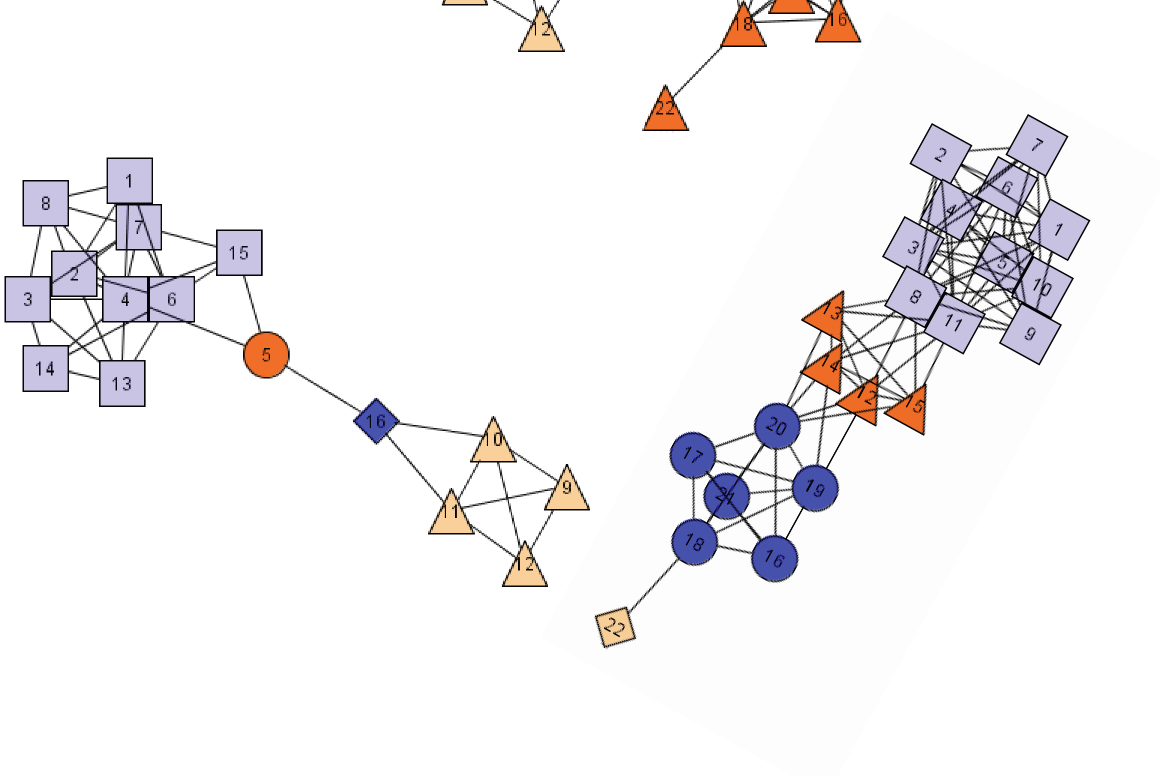


Experimental Evaluation (III)





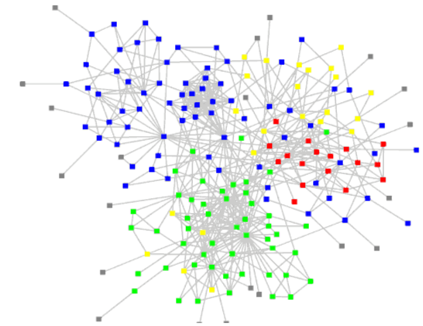
FastModularity clustering result

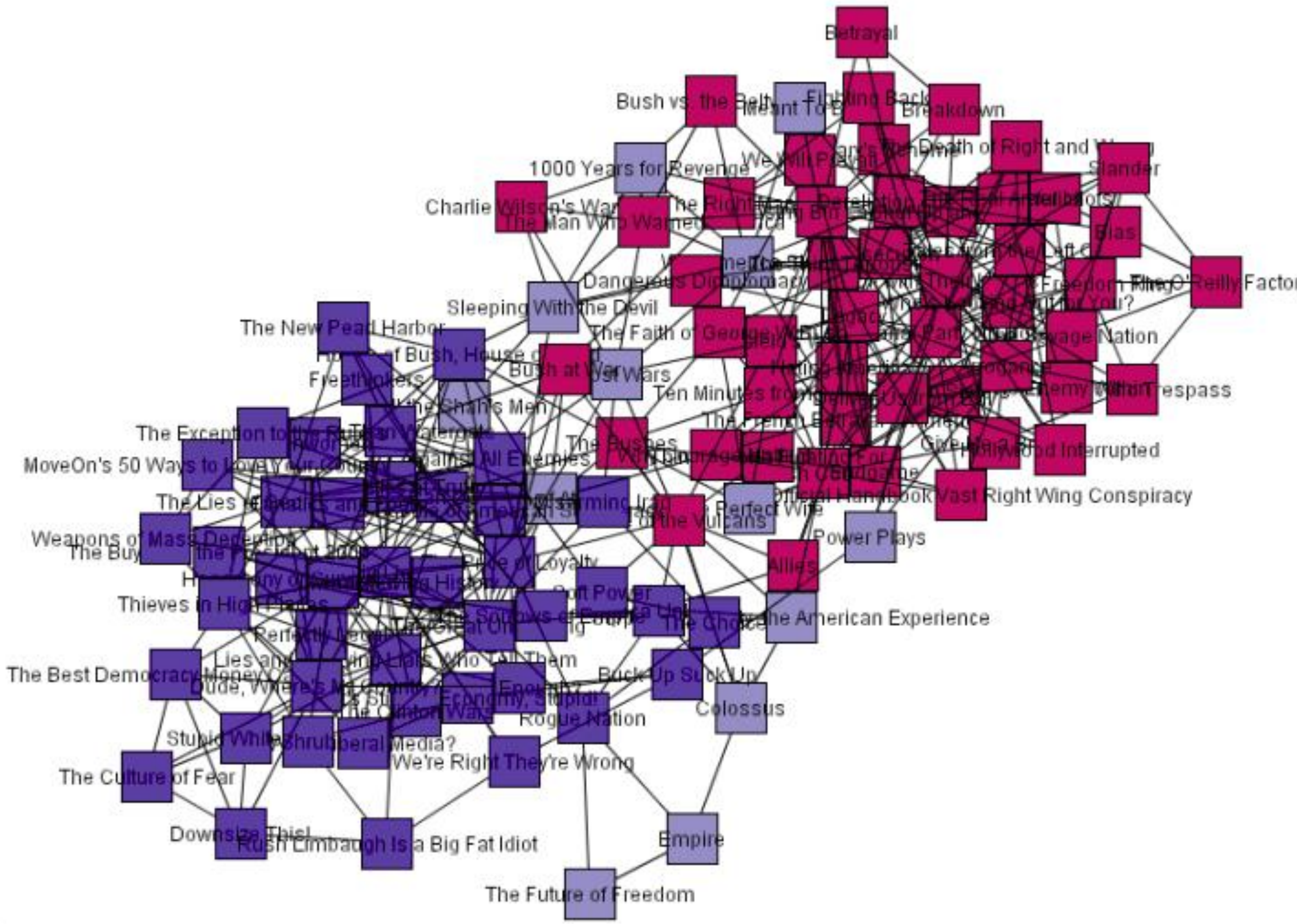


SCAN clustering result

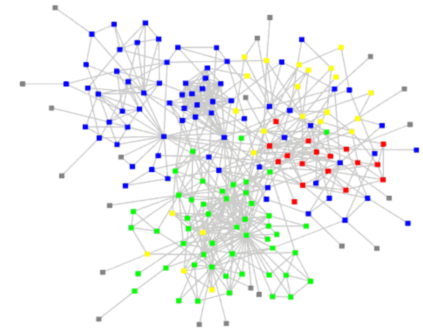
Experimental Evaluation (IV)

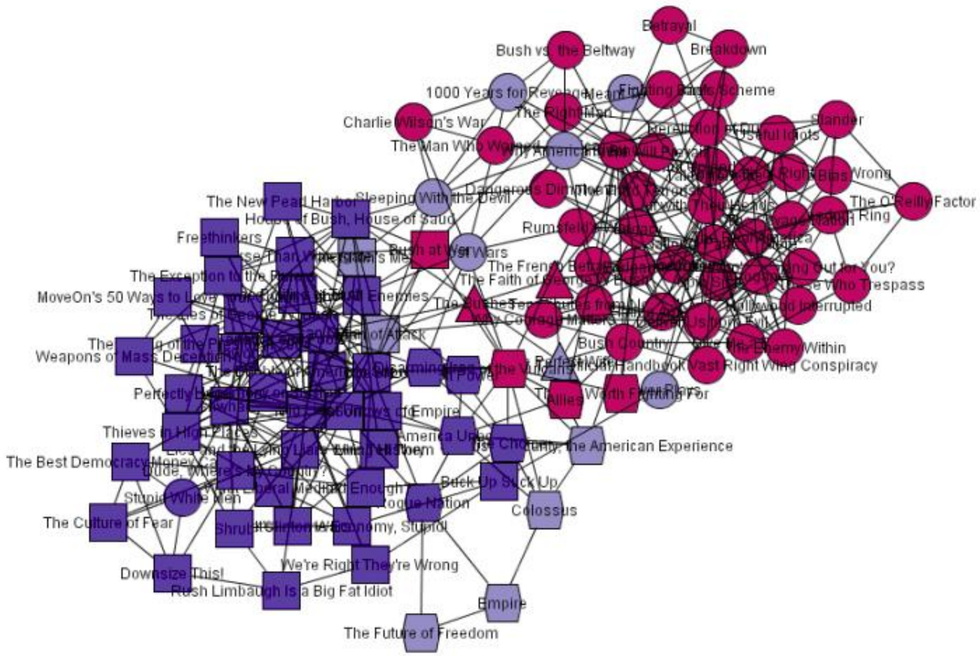
Books about US politics dataset



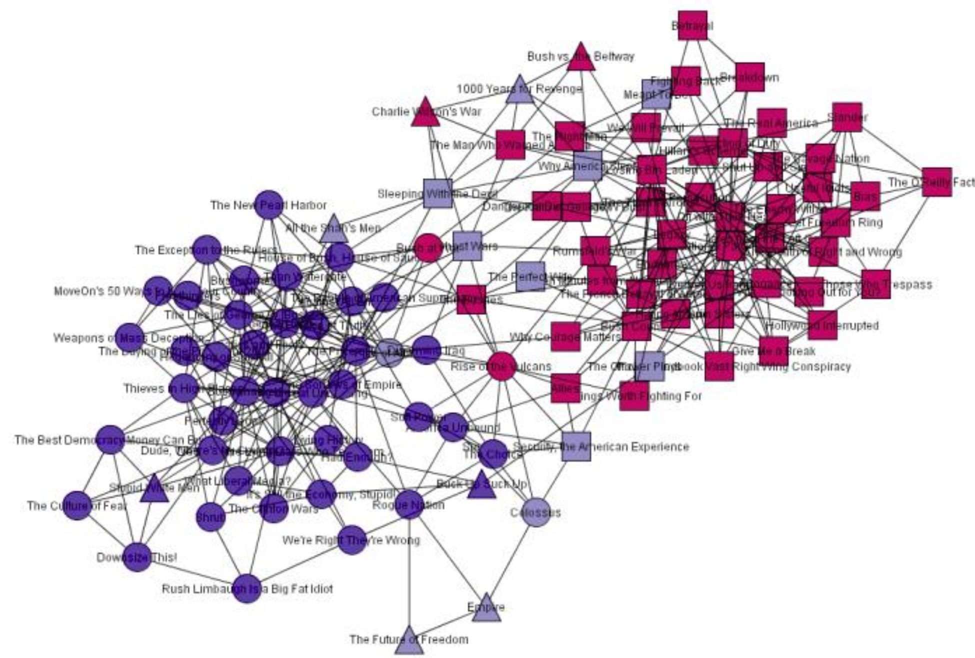


Experimental Evaluation (V)



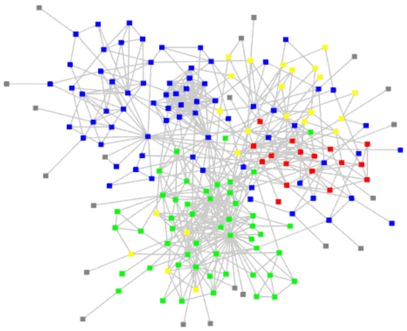


FastModularity clustering result



SCAN clustering result

Discussion



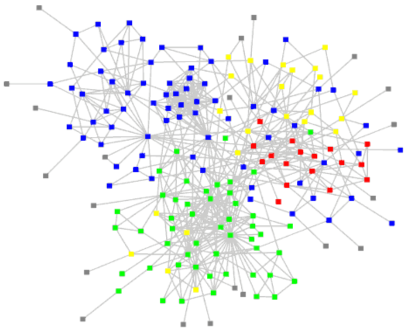
+ identifies not only clusters, but also outliers and hubs

+ linear run-time complexity wrt. # edges, each vertex is visited only once

- performance highly depends on sensitive input parameters

- ignores domain knowledge in clustering attributes

- assumes that network is homogeneous and adjacency matrix is already defined



Q & A

♥