Analysis of Massive Data Sets

http://www.fer.hr/predmet/avsp

Prof. dr. sc. Siniša Srbljić

Doc. dr. sc. Dejan Škvorc

Doc. dr. sc. Ante Đerek

Faculty of Electrical Engineering and Computing
Consumer Computing Laboratory

Community Detection in Social Network Graphs

Goran Delač, PhD

Outline

- **Social Networks**
  - Social graph
- **Detecting Communities**
  - Traditional approaches
- **Affiliation Graph Model**
  - Detecting communities using AGM
- **BigCLAM Approach**

Social Networks

- Modern social interactions

Social Networks

- **Modern social interactions**

- **Massive communities (Q4 2015)**
  - Google+ 2,200,000,000 (profiles, low activity)
  - Facebook 1,591,000,000
  - Instagram 400,000,000
  - Twitter 320,000,000

Social Networks

- **Vast amounts of data – immense opportunities**
  - Trend analysis, information cascades
  - Sentiment analysis
  - Social search
  - Recommendations
  - …
  - **Detecting communities**

Social Networks

- **What are social networks?**

1. Collection of entities (usually people, but not necessary)

2. At least one relationship exists between entities (friendship, follower, …)
    - Unidirectional/Bidirectional
    - Binary/Weighted

Social Networks

- **What are social networks?**
  3. Assumption of nonrandomness (locality)
     - If entity A is related to both B and C, there is higher then average probability that B and C are related

Facebook social graph visualization

Social Networks

- **Social Network Representation**
  - Social graph
    - Entities are nodes
    - Connections are edges
  - Connections can have a degree
    - Labeled edges
  - Connections can have a direction
    - Directed (G+, Twitter)
    - Undirected (Facebook)

Social Networks

- **Social Network Representation**
  - Social graph
    - Entities are nodes
    - Connections are edges
  - Entities can have different types
    - e.g. users, pages, tags
      - users can be related if they tag the same pages
    - k-partite social graph

Social Networks

- **Relationships other than "friendship"**
  - Telephone networks
  - Email networks
  - Collaboration networks

    ⋮

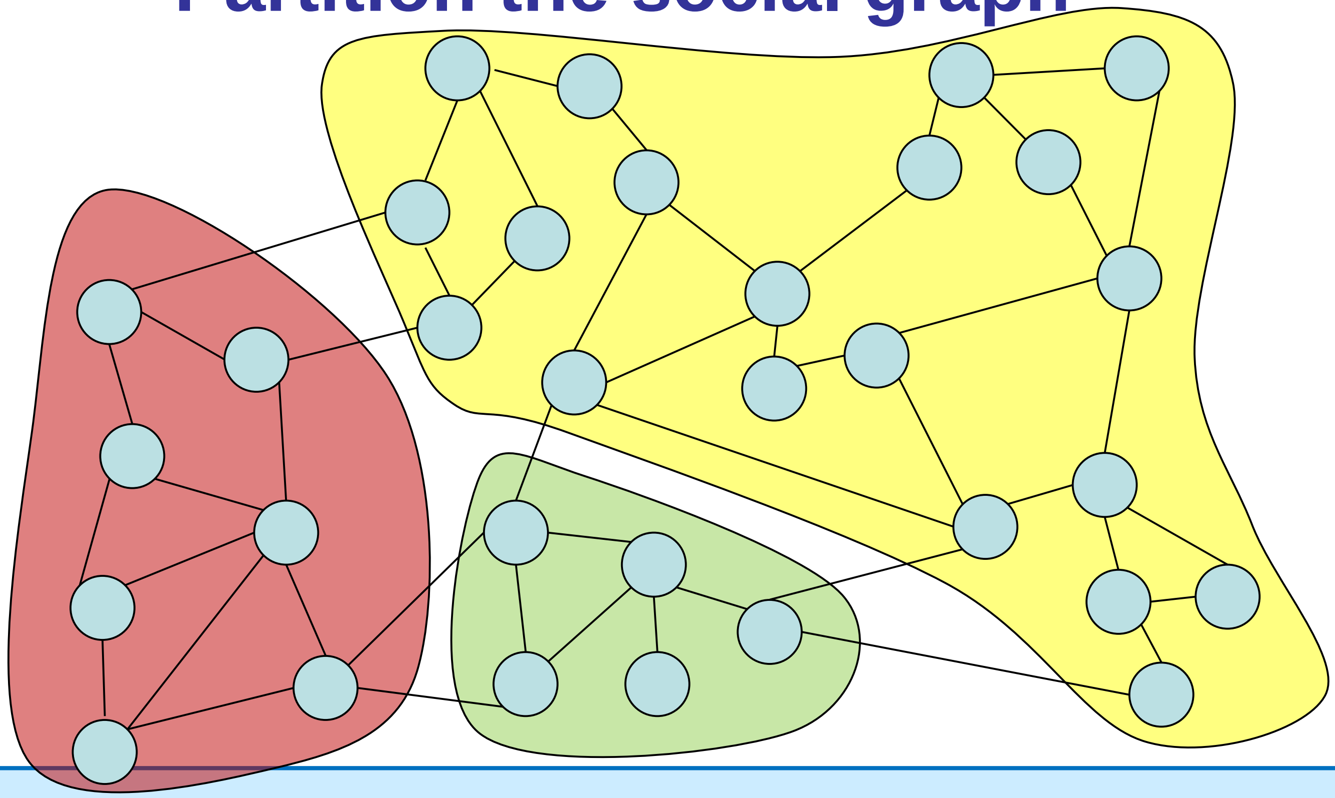
  - Information nets, infrastructure nets, …

They all exhibit locality of "friendship"

Social Networks

- **Goal**
  - **Extract** knowledge of social communities

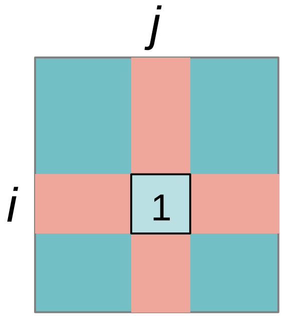
Partition the social graph



Detecting Communities

Traditional methods

- Minimal cut
  - One of the oldest methods
  - Minimize number of edges between communities
  - Problem: Will find communities even if they do not manifest
- Hierarchical clustering
  - Apply similarity measure on the adjacency matrix
    - Cosine similarity, Jaccard similarity, Hamming distance



  - Single-linkage clustering
    - All node pairs in different communities have similarity below a certain threshold

Detecting Communities

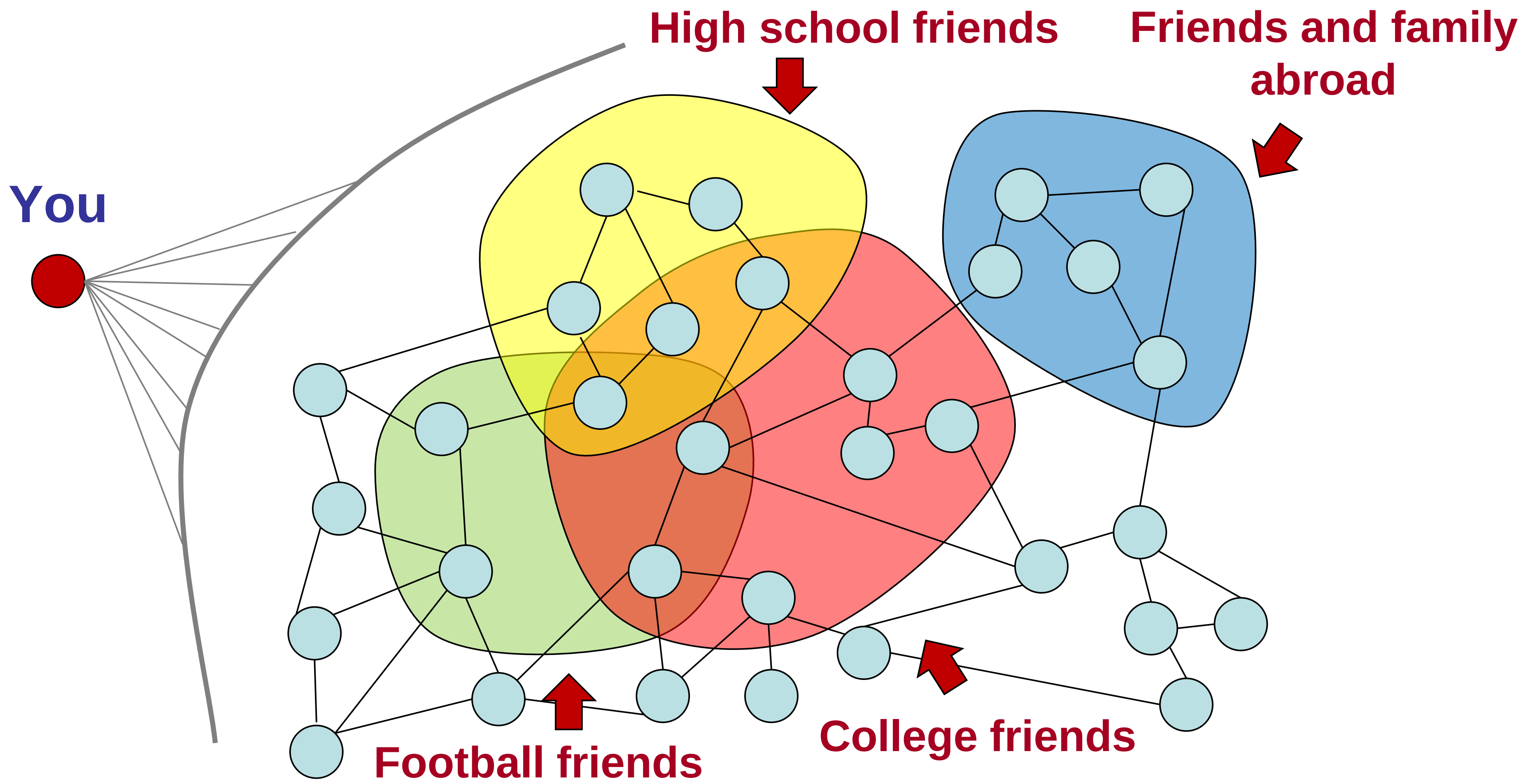
- **Traditional methods**
  - Girvan-Newman Algorithm
    - Identifies edges that lie between the communities and removes them
    - Measure: **betweenness**
    - Betweenness – number of times a node acts as a connection along the shortest path between other nodes in the graph
    - Edge betweenness – number of times an edge is within a shortest path between any two nodes in a graph

Detecting Communities

- **Traditional methods**
  - Girvan-Newman Algorithm
    1. Calculate edge betweenness
    2. Remove edge with highest betweenness
    3. Recalculate betweenness
    4. Repeat steps 1-3 until graph brakes down into communities

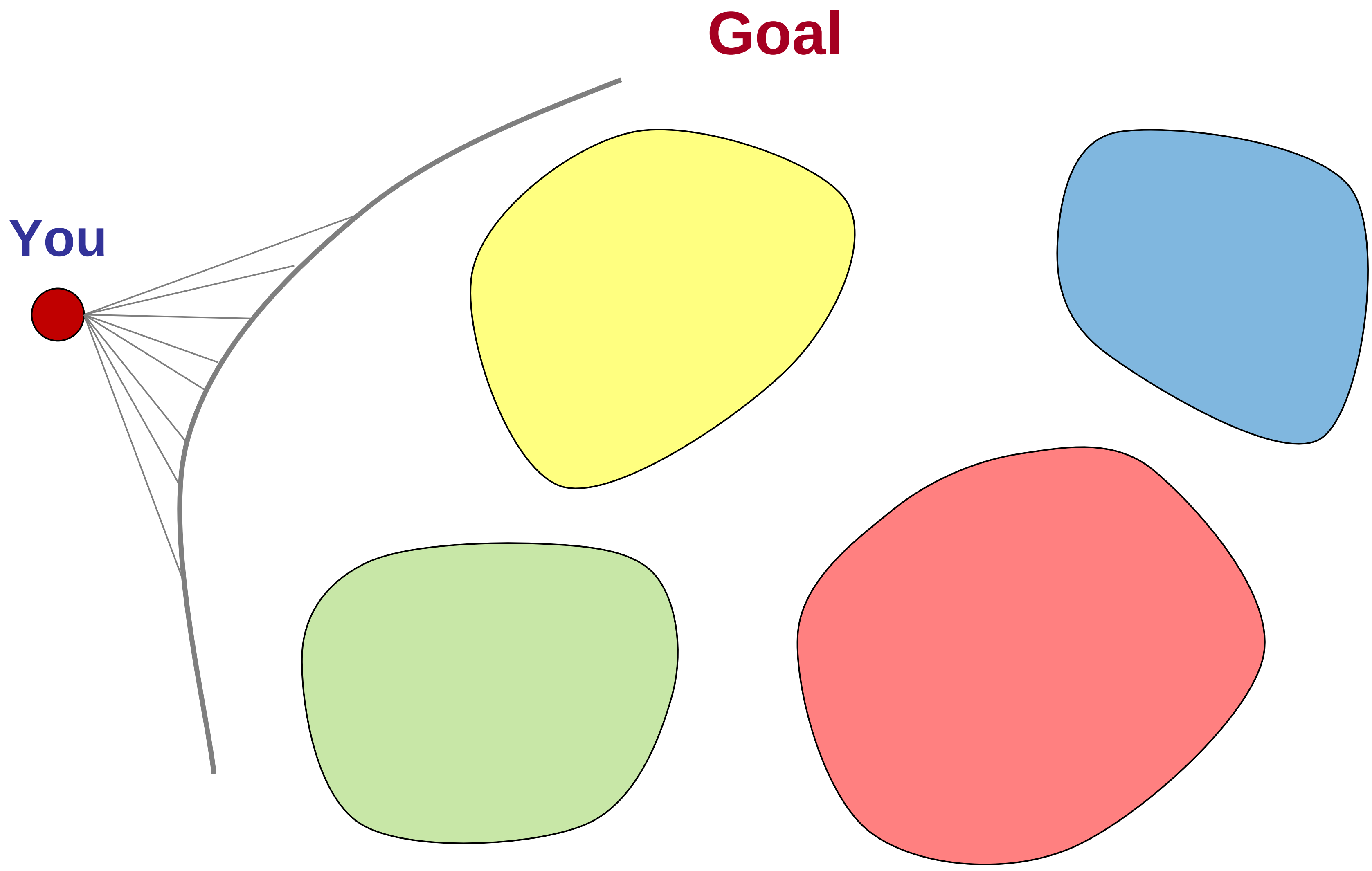
    - Effective but computationally heavy: $O(m^2n)$, m - edges, n - vertices

Detecting Communities



Communities can overlap!

Detecting Communities



Detecting Communities

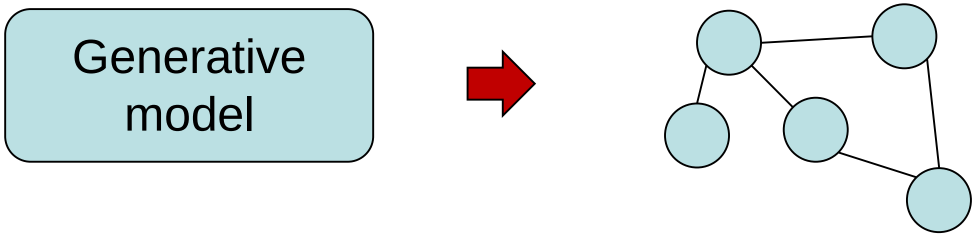
- **Problem**
  - Communities can overlap!

- **Solution**
  - Clique detection methods
    - All nodes within in a clique are directly connected (dense graph)
    - Clique percolation method (CPM)
  - Affiliation graph model (AGM)
    - Yang, Leskovec 2012 [3]

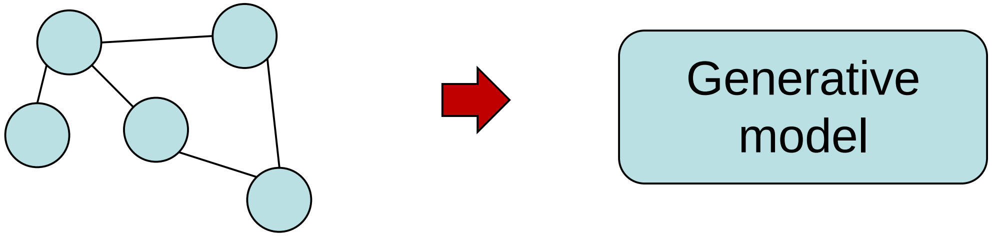
Affiliation Graph Model

- **Approach: Utilize generative models**

1. Using a model generate the social network

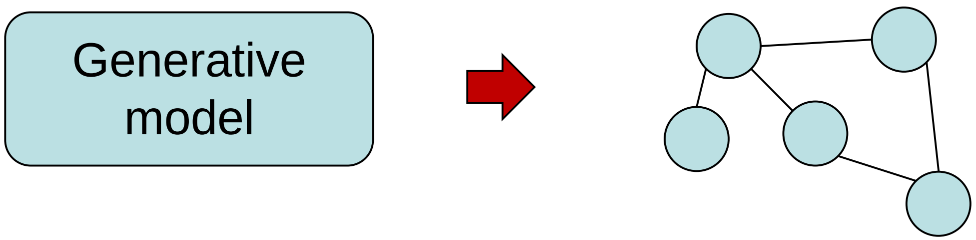


2. Given a social network, derive the most appropriate model that describes it



Affiliation Graph Model

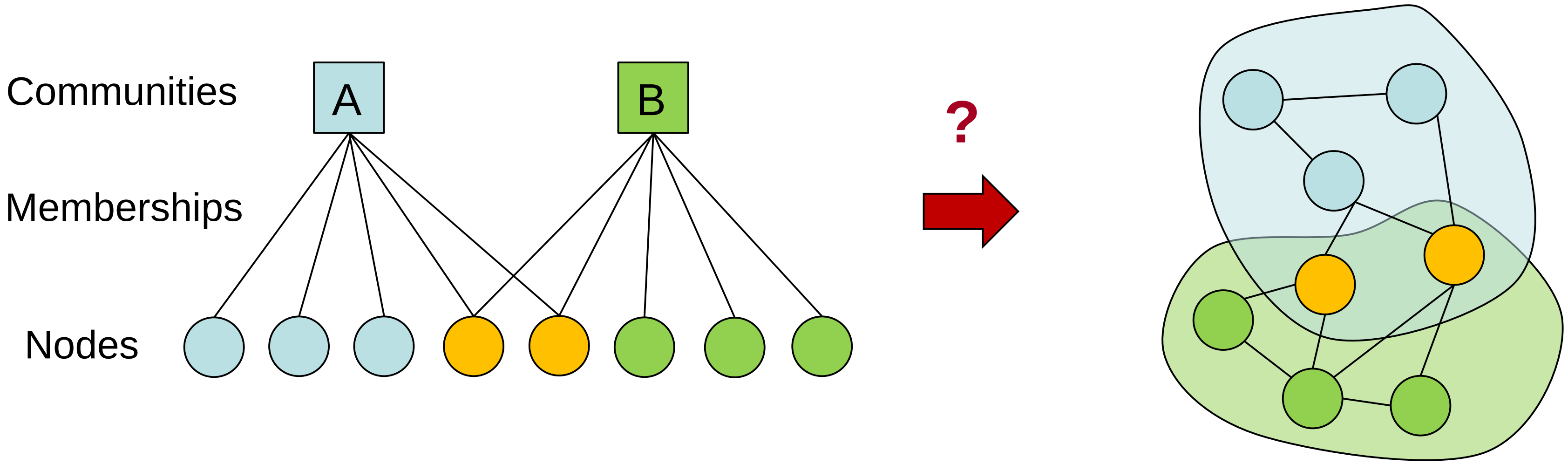
- **Goal: Derive a model that generates social networks**
  - The model has a set of parameters that need to be estimated
    - In doing so we in effect detect communities



  - Given network nodes (entities), how do communities (defined by AGM) generate the edges of the network?

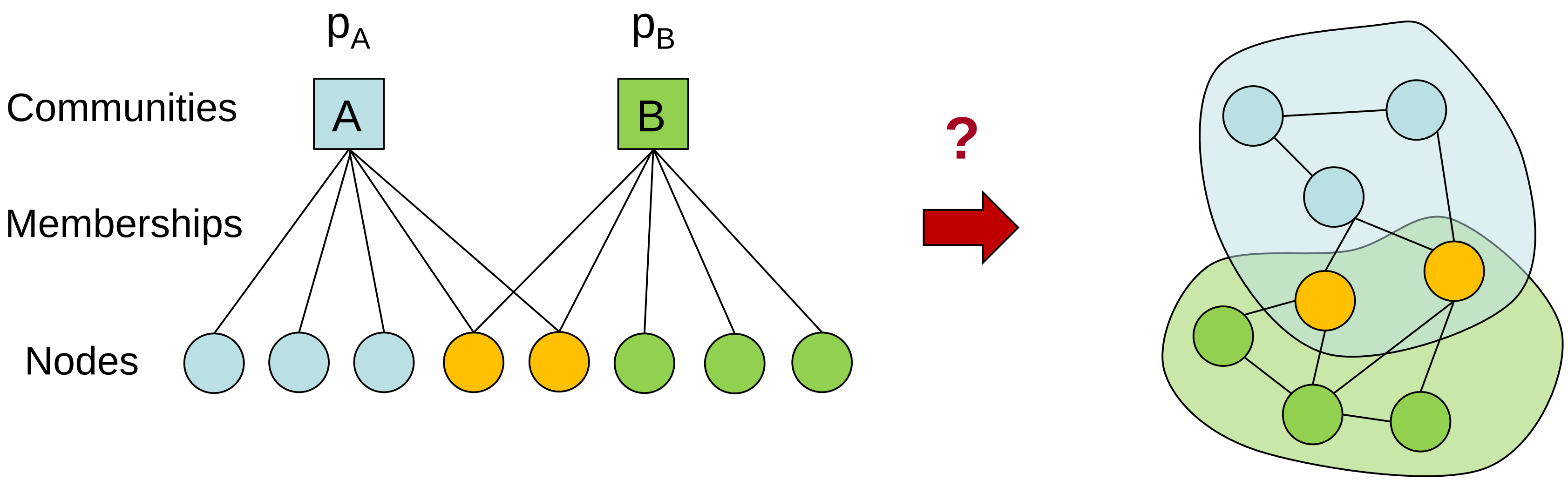
Affiliation Graph Model

- **Model**
  - Two types of nodes
    - Social network entities (nodes)
    - Communities
  - Edges
    - Community membership



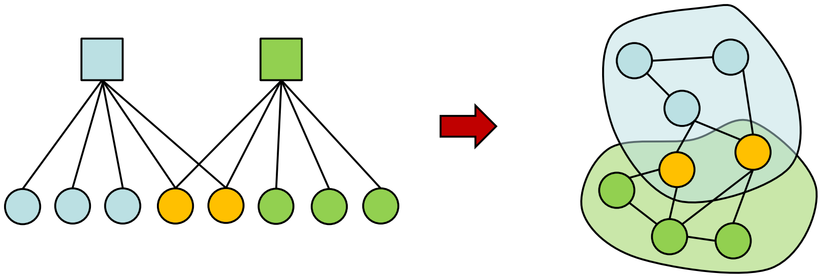
Affiliation Graph Model

- **Model**
  - Probability that a node links to other nodes in community C
    - p_c
  - AGM(N, C, M, $\{p_c\}$)



Affiliation Graph Model

- **Generative process**



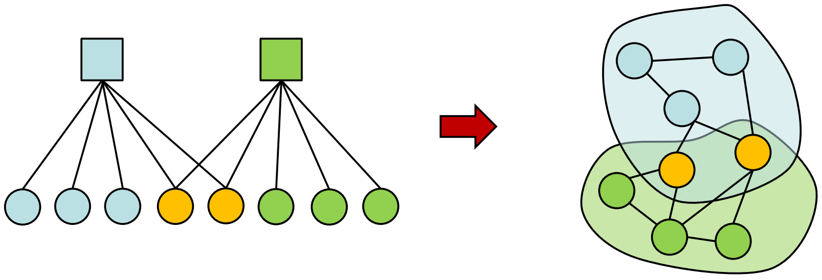
  - Each pair of nodes in community C is connected with probability p_c

- **Total probability**
  - Nodes ***u*** and ***v*** are connected

$$P(u,v) = 1 - \prod_{c \in M_u \cap M_v} (1 - p_c)$$

Affiliation Graph Model

- Generative process

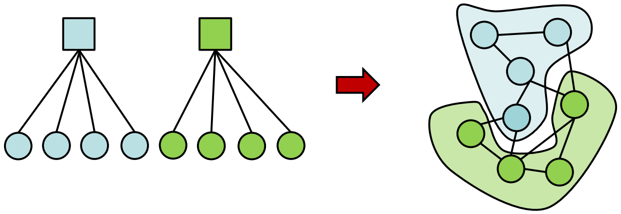


  - Go through all node pairs ***(u, v)*** and generate a connection (edge) with probability:

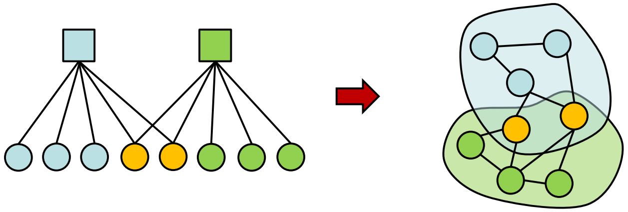
$$P(u,v) = 1 - \prod_{c \in M_u \cap M_v} (1 - p_c)$$

Affiliation Graph Model

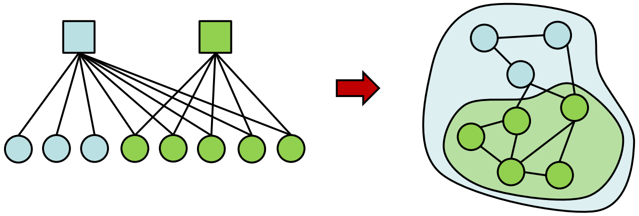
- **Advantage of AGM**
  - Flexible community representation
    - Non-overlapping



    - Overlapping

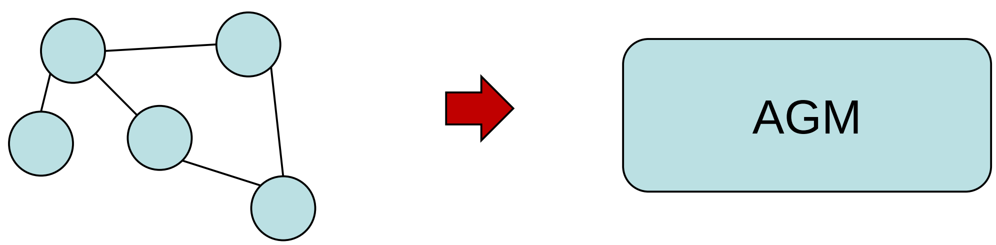


    - Nested



Detecting Communities using AGM

- **Finding the model = finding the communities**



Input: social network

Output: AGM

Community memberships

Detecting Communities using AGM

- **Find AGM parameters**

  - **N** → social net (entities) → **directly from graph**

  - **C** → number of comminutes → **estimate form graph [3]**

  - $\mathbf{p_c}$ → prob. that two nodes in community are connected

  - **M** → community memberships

Detecting Communities using AGM

- **Solution: Maximum Likelihood Estimation**
  - Given a social graph ***G***
  - Given a model ***f(param)***
  - We want to estimate $\boldsymbol{P_f(G \mid param)}$
    - Conditional probability
    - The probability that **AGM** generated ***G*** given parameters ***param***

- **Find the most likely model that generated graph G**

$$\arg\max_{param} \; P_f(G \mid param)$$

Detecting Communities using AGM

- **MLE: Example**
  - Suppose we have a sequence of events (e.g. coin toss, rainy days etc.)
  - X = [0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0]
  - Model ***f*(*Y*) –** return 1 with probability ***Y***
  - What is $\boldsymbol{P_f(X \mid Y)}$ **?**

  - Assuming the events are independent
    - $\boldsymbol{P_f(X \mid Y) =} P_f(0 \mid Y) * P_f(1 \mid Y) * \ldots * P_f(0 \mid Y) = \boldsymbol{Y^5 (1-Y)^7}$
    - $\boldsymbol{P_f(X \mid Y = 5 / 12) = 0.0002886432}$

    - **X was most probably generated if Y = 5 / 12**

Detecting Communities using AGM

- **MLE for AGM**
  - Event = two network nodes are connected
    - $P(u, v)$
  - Likelihood of **AGM** generating some graph **G with** a set of edges **E**:

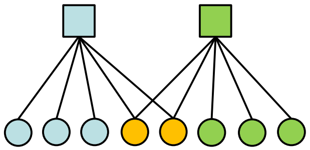
$$P(G \mid param) = \prod_{(u,v) \in E} P(u, v) \prod_{(u,v) \notin E} \left(1 - P(u, v)\right)$$

Detecting Communities using AGM

- **MLE for AGM**
  - **Goal:** find parameters ***param*** = (N, C, M, $\{p_c\}$) such that:

$$\arg\max_{param} \prod_{(u,v) \in E} P(u,v) \prod_{(u,v) \notin E} \left(1 - P(u,v)\right)$$

  - Problem
    - Finding ***param*** is equal to finding bipartite affiliation network
    - Too hard for large data sets!

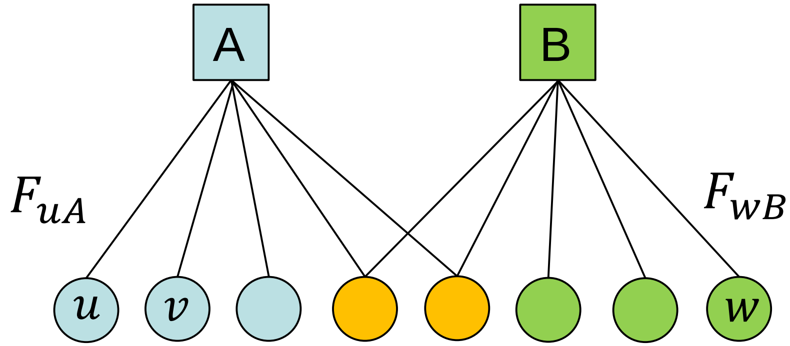


BigCLAM Approach

- **Solution: Relax the model**
- **BigCLAM (Yang, Leskovec 2013) [2]**
  - **C**luster **A**ffiliation **M**odel for **B**ig **N**etworks
  - Idea: avoid discrete memberships
    - Introduce strengths to memberships
    - Strengths are non negative values
    - If strength is **0**, the entity is not a member of the given community
    - If strength is **high**, the entity is a very active community member
  - Implications
    - Moving to continuous domain enables usage of very effective approaches, like gradient descent

BigCLAM Approach

- **Membership strengths**



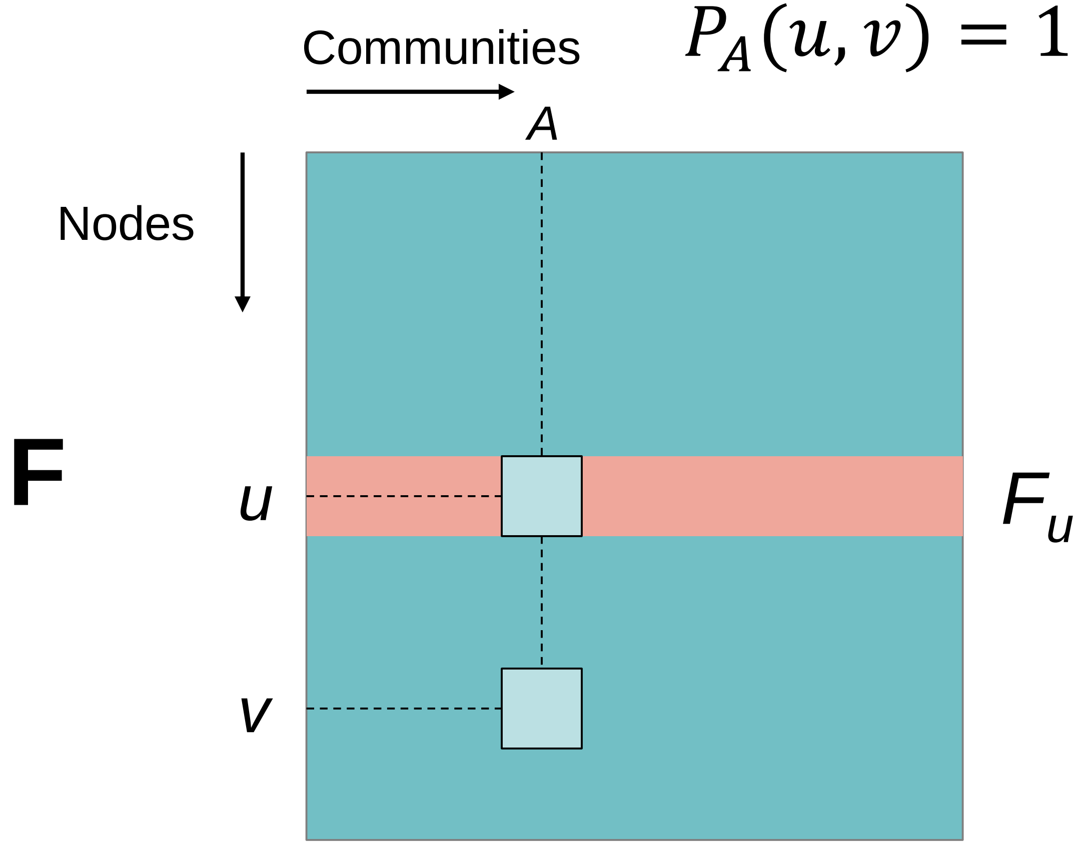
  - $F_{uA} > 0$: membership strength
  - Probability that nodes ***u*** and ***v*** are connected in **A**:

$$P_A(u, v) = 1 - e^{-F_{uA} \cdot F_{vA}}$$

BigCLAM Approach

- **Membership strength matrix F**

$$P_A(u,v) = 1 - e^{-F_{uA} \cdot F_{vA}}$$



BigCLAM Approach

- **Probability that at least one common node connects nodes *u* and *v***

$$P_A(u,v) = 1 - e^{-F_{uA} \cdot F_{vA}}$$

$$P(u,v) = 1 - \prod_{c} (1 - P_c(u,v))$$

$$P(u,v) = 1 - e^{-\sum_C F_{uC} \cdot F_{vC}}$$

$$= 1 - e^{-F_u \cdot F_v^T}$$

BigCLAM Approach

- **Goal: Find such F so that:** $P(u,v) = 1 - e^{-F_u \cdot F_v^T}$

$$\arg\max_{\mathbf{F}} \prod_{(u,v) \in E} P(u,v) \prod_{(u,v) \notin E} \left(1 - P(u,v)\right)$$

$$\arg\max_{\mathbf{F}} \prod_{(u,v) \in E} \left(1 - e^{-F_u \cdot F_v^T}\right) \prod_{(u,v) \notin E} e^{-F_u \cdot F_v^T}$$

BigCLAM Approach

- **Modification: log likelihood**
  - Why?
    - Sums instead of products
    - Errors are less pronounced when summing small numbers

$$\log P(X)$$

BigCLAM Approach

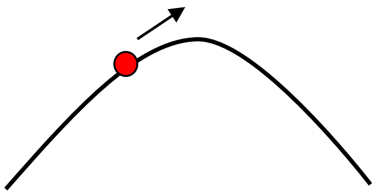
$$l(F) = \log P(G|F)$$

- **Goal**
  - Find ***F*** that maximizes:

$$\boxed{l(F) = \sum_{(u,v) \in E} \log\left(1 - e^{-F_u \cdot F_v^T}\right) - \sum_{(u,v) \notin E} F_u \cdot F_v^T}$$

BigCLAM Approach

- **Gradient descent**
  1. Compute a gradient for a single row
  2. Update row – move in the direction of gradient



  3. Repeat for all rows until F stops changing

BigCLAM Approach

- **Gradient descent**

$$l(F_u) = \sum_{v \in N(u)} \log\left(1 - e^{-F_u \cdot F_v^T}\right) - \sum_{v \notin N(u)} F_u \cdot F_v^T$$

$N(u)$ neighbors of node ***u*** (set of outgoing neighbors)

BigCLAM Approach

- **Gradient descent**

$$\nabla l(F_u) = \sum_{v \in N(u)} F_v \frac{e^{-F_u \cdot F_v^T}}{1 - e^{-F_u \cdot F_v^T}} - \sum_{v \notin N(u)} F_v$$

Update row

$$F_u \leftarrow F_u + \mu \cdot \nabla l(F_u)$$

If $F_{uC} < 0$: $F_{uC} = 0$

BigCLAM Approach

- **Gradient descent**

$$\nabla l(F_u) = \sum_{v \in N(u)} F_v \frac{e^{-F_u \cdot F_v^T}}{1 - e^{-F_u \cdot F_v^T}} - \sum_{v \notin N(u)} F_v$$

  - Computing $\nabla l(F_u)$ is slow!
    - Takes linear time on the size of network

However!

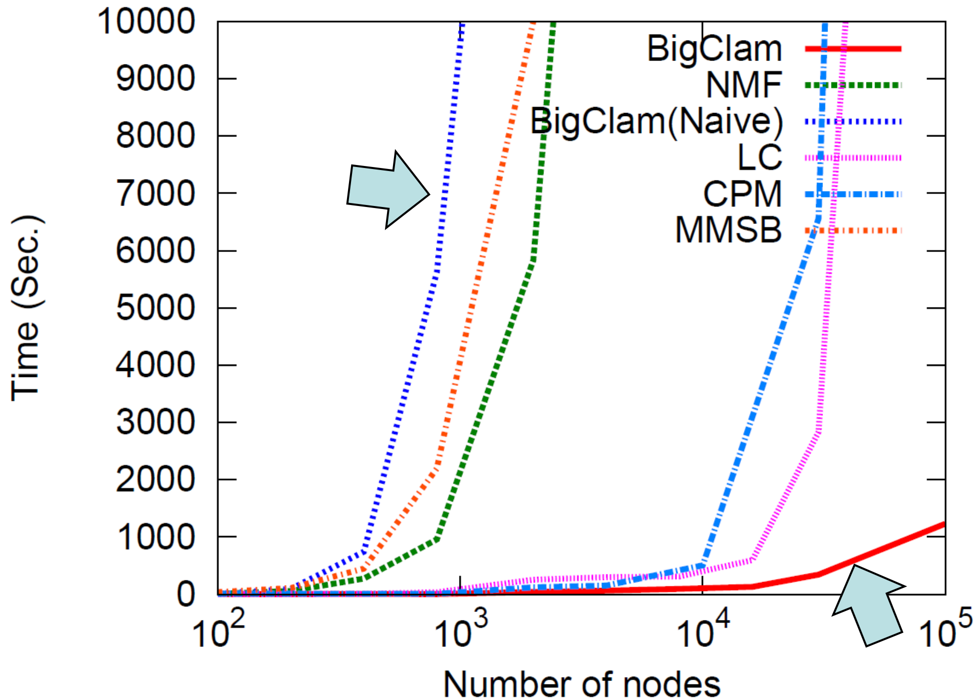
Compute once at the beginning of a pass

$$\sum_{v \notin N(u)} F_v = \sum_{v} F_v - F_u - \sum_{v \in N(u)} F_v$$

Computing $\sum_{v \notin N(u)} F_v$ now takes linear time in the degree of node u ($|N(u)|$)

Node degree is **much smaller** than the total number of nodes in the network!

BigCLAM Approach



[2] Yang, Leskovec 2013

~ 5 min for 300k nodes

~ 1 day for network with 100M edges

Literature

1. J. Leskovec, A. Rajaraman, and J. D. Ullman, "Mining of Massive Datasets", 2014, Chapter 6 : "Mining Social-Network Graphs" (link)

2. J. Yang, J. Leskovec: "Overlapping community detection at scale: a nonnegative matrix factorization approach ", WSDM '13 Proceedings of the sixth ACM international conference on Web search and data mining, Rome, Italy, February 2013, pp. 587 – 596. (link)

3. J. Yang, J. Leskovec: "Community-Affiliation Graph Model for Overlapping Network Community Detection", ICDM '12 Proceedings of the 2012 IEEE 12th International Conference on Data Mining, Brussels, Belgium, December 2012, pp. 1170–1175. (link)