Academic Year 2023/2024

Social Media Analytics (SMA)

Community Detection

Part 2

Marco Viviani

University of Milano-Bicocca

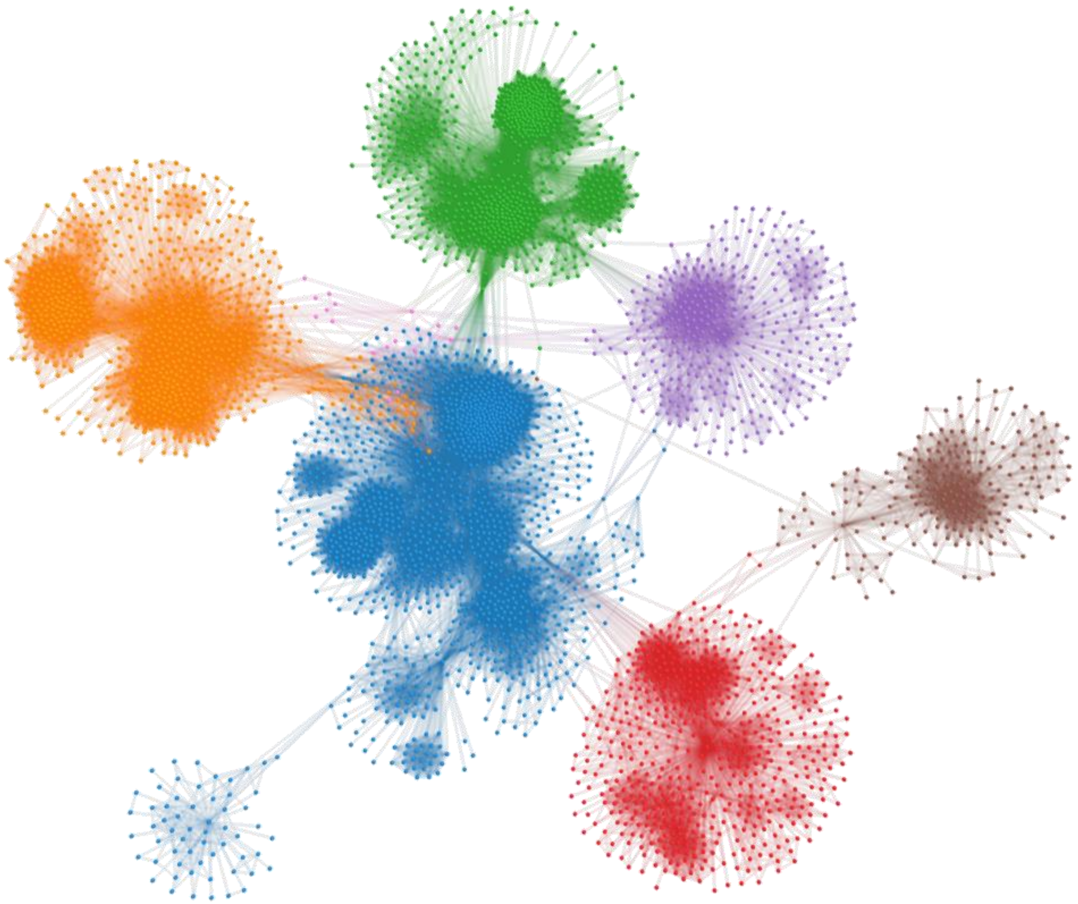
Department of Informatics, Systems, and Communication

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

DIPARTIMENTO DI INFORMATICA, SISTEMISTICA E COMUNICAZIONE

Outline

- Degree correlation
    - Small and giant components
    - Network percolation
- Community detection
    - Granularity
    - Overlapping communities
- Community detection algorithms
    - Flat VS Hierarchical algorithms
    - Modularity-based algorithms
    - Label propagation-based algorithms
    - Random walk-based algorithms
    - Graph partitioning-based algorithms



Impact of Degree Correlation

- Most real networks are characterized by some **degree correlations**.
  - Social networks are assortative (mostly).
  - Biological networks display disassortativity.

- These correlations raise an **important question**:
  - Why do we care?
  - In other words, do degree correlations alter the properties of a network? And which network properties do they influence?

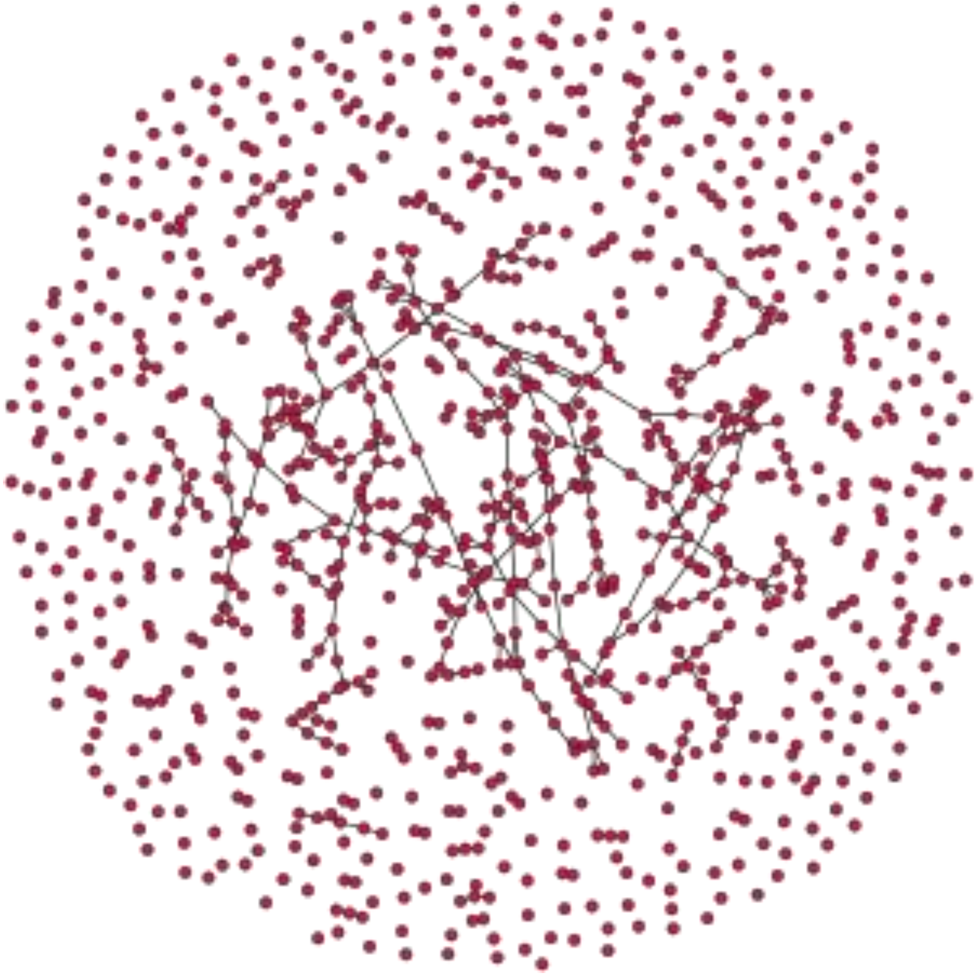
→ Impact on the giant component.

The Giant Component

- In a complex network, the **giant component** refers to the largest connected subgraph within the network.
  - It contains a significant proportion of the entire nodes in the network.
  - Typically, as the network expands the giant component will continue to have a significant fraction of the nodes.

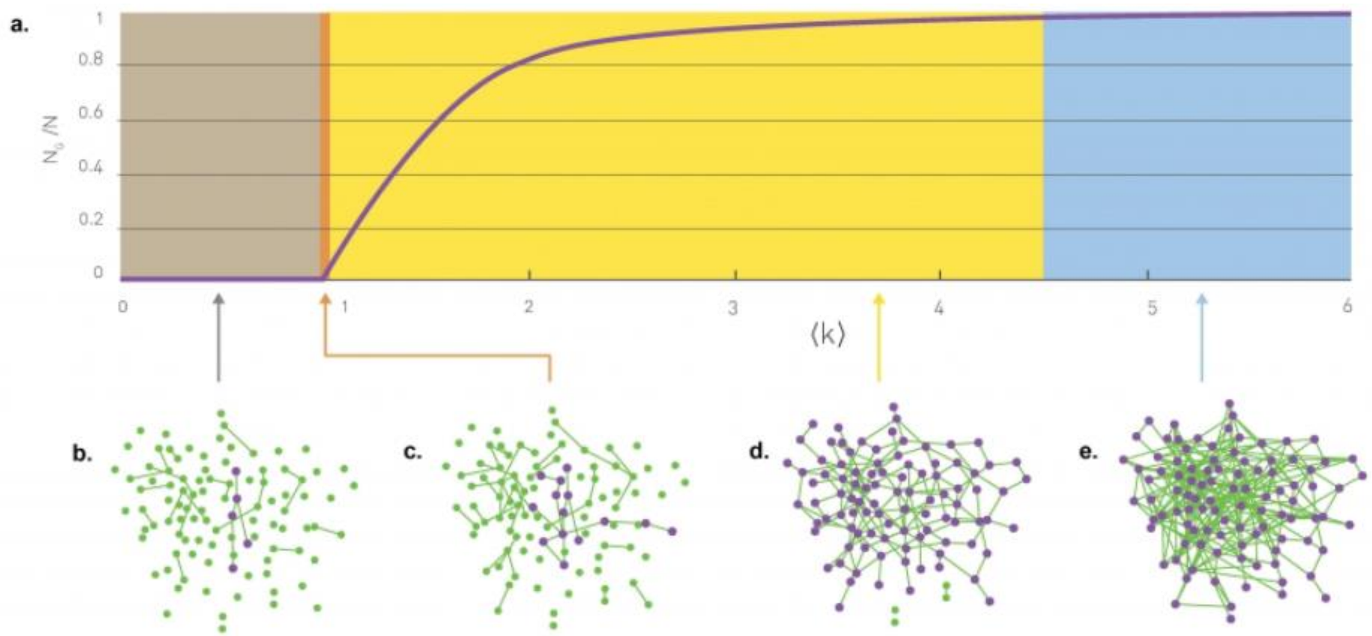
- Random networks
  - In sufficiently dense graphs distributed according to the Erdős–Rényi model, a giant component exists with high probability.
  - A giant component is a connected component whose fraction of the overall number of vertices is bounded away from zero.

The Giant Component: Example



Degree Correlation and Random Networks

- An important property of a random network is the emergence of a phase transition at $\langle k \rangle = 1$, marking the appearance of the **giant component**.



$\langle k \rangle$ indicates the average degree.

Degree Correlation and Real Networks

- **Assortative networks**
  - The phase transition point moves to a lower $\langle k \rangle$, hence a giant component emerges for $\langle k \rangle < 1$.
    - The reason is that it is easier to start a giant component if the high-degree nodes seek out each other.

- **Dissassortative networks**
  - The phase transition is delayed since in these networks the hubs tend to connect to small degree nodes.
    - Disassortative networks have difficulty forming a giant component.

- The importance of communities within a network, whether they are on the giant component or in smaller connected components, depends on the specific context and goals of the considered analysis.

Giant Component Communities

- **Connectivity**: communities within the giant component are generally more connected and have more direct interactions with a larger portion of the network.
  - This can be important if the goal is to disseminate information or influence a significant part of the network.

- **Impact**: if the aim is to have a broad impact on the network or leverage the network's collective influence, then communities within the giant component might be more important.

Small Component Communities

- **Niche influence**: communities within smaller connected components might represent specialized or niche areas of interest within the network.
  - They can be crucial for addressing specific needs or interests that are not well-represented in the giant component.
  - In social networks, a small community might be incredibly important for individuals within it, even if it is not part of the giant component.
  - Conversely, in a transportation network, a small community might be essential for local travel but not have much impact on long-distance travel.

Degree Correlation and Network Resilience

- Alterations of the giant component have implications for **network resilience**.
  - Effects of node and edge removal.

- In assortative networks hub removal makes less damage because the hubs form a core group, hence many of them are redundant.

- Hub removal is more damaging in disassortative networks, as in these the hubs connect to many small-degree nodes, which fall off the network once a hub is deleted.

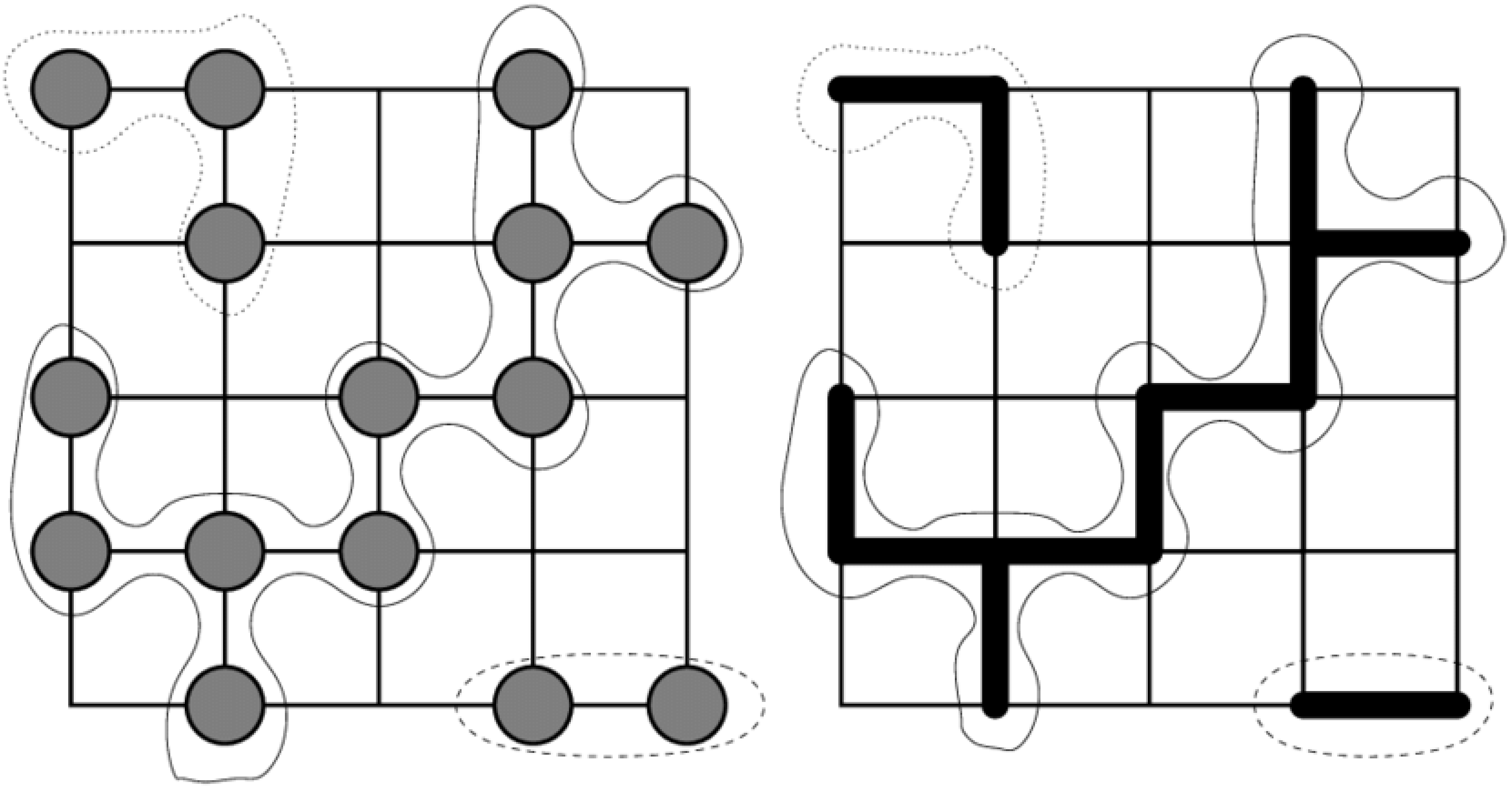
Network Resilience and Network Percolation

- **Network percolation** is a modeling and analysis technique used to study how network connectivity changes as nodes or links are systematically removed or disrupted.

- It focuses on the critical point – or **percolation threshold** – at which a network transitions from a connected state to a fragmented state as nodes or links are removed based on some probabilistic rule.
  - Site percolation where nodes are removed with a certain probability.
  - Bond percolation where links are removed.

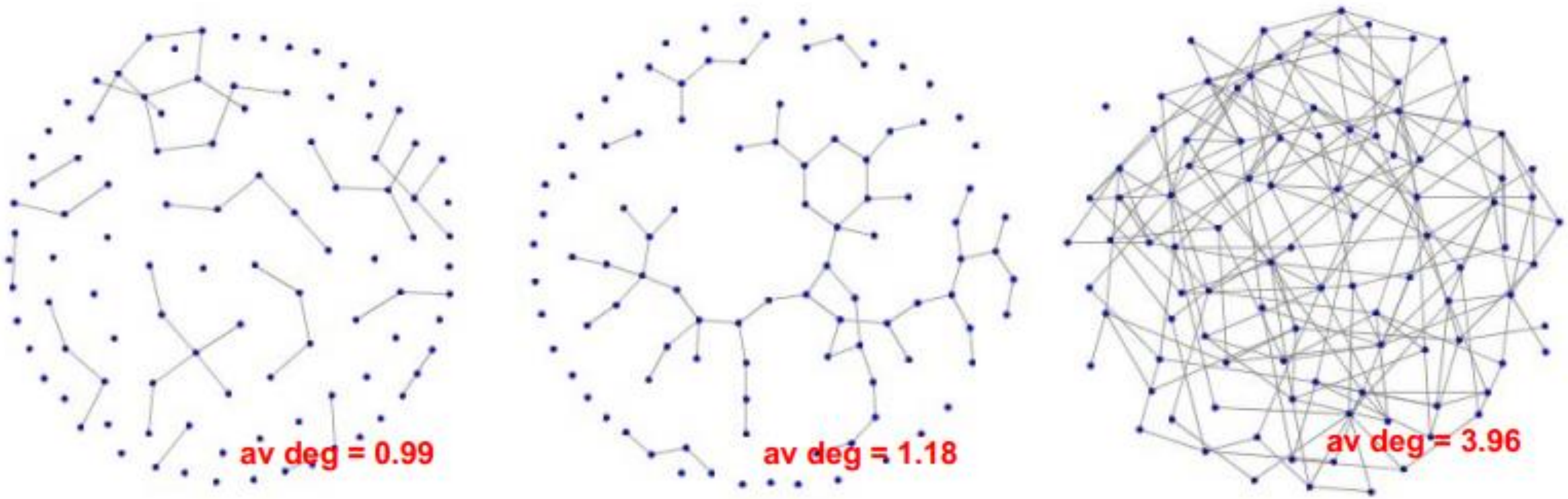
- The percolation process helps researchers understand the network's phase transition and the impact of random or targeted failures on network connectivity.

Site and Bond Percolation



Percolation Threshold and Giant Component

- **Percolation threshold**: it can be interpreted as the point at which the giant component emerges.



Community Detection

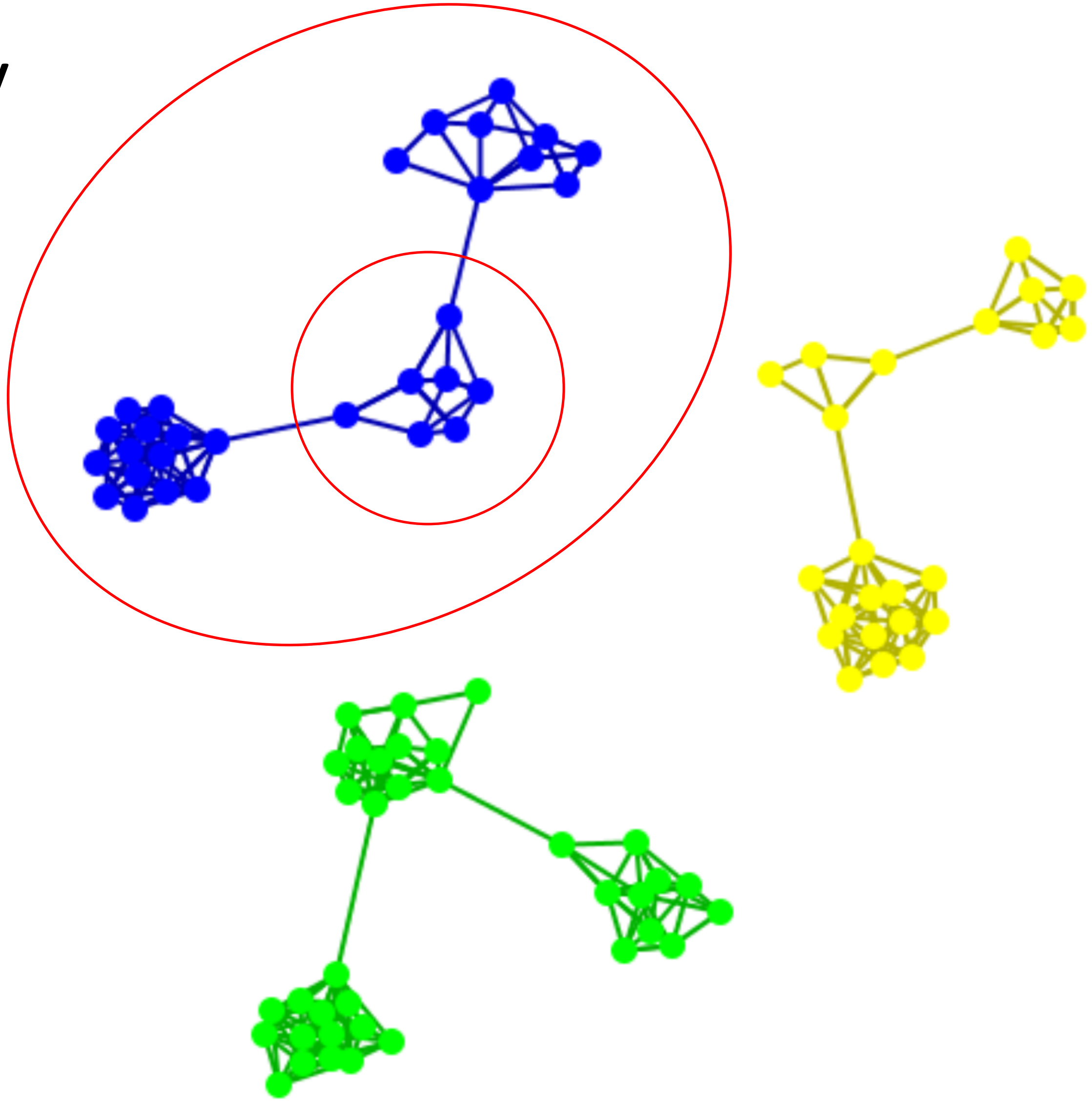
- **Community detection** is the process of identifying subgroups or communities within a network, where nodes within a community have more connections with each other than with nodes outside the community.
  - Recent approaches also consider aspects other than just topology.

- In scale-free networks, the presence of hubs (preferential attachment, small-world phenomenon, assortativity) can influence the structure of communities.
  - Some communities might be centered around hubs, while others may be more isolated.

Community Granularity

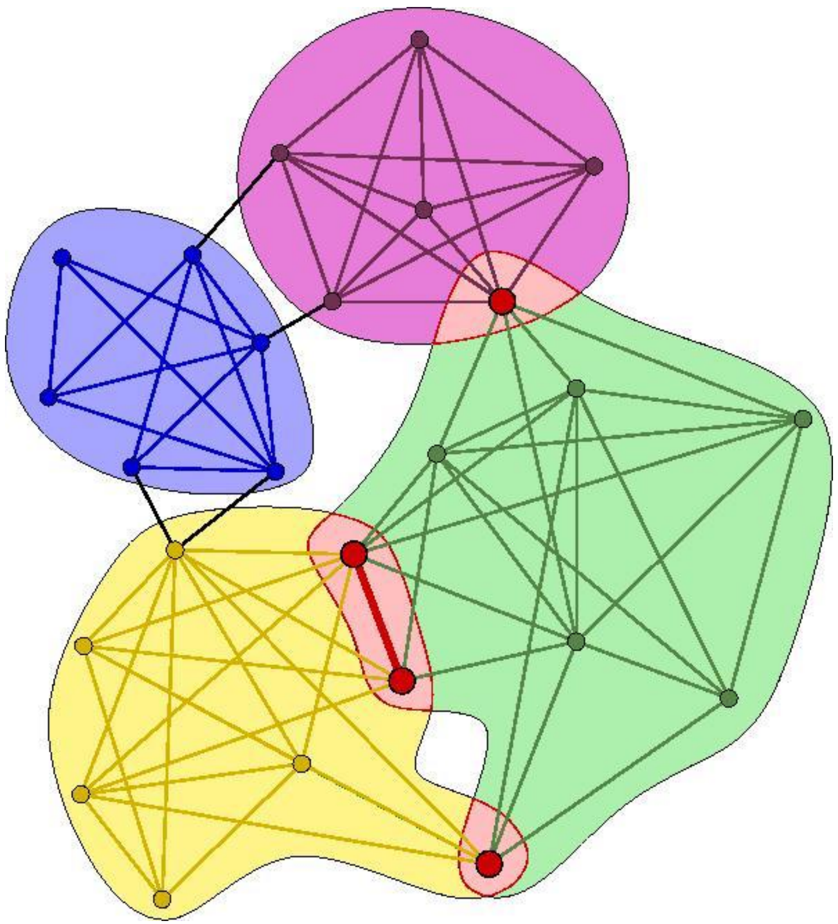
- The level of granularity of a community can vary and can depend on subjectivity (and the aim for which the analyses are performed).

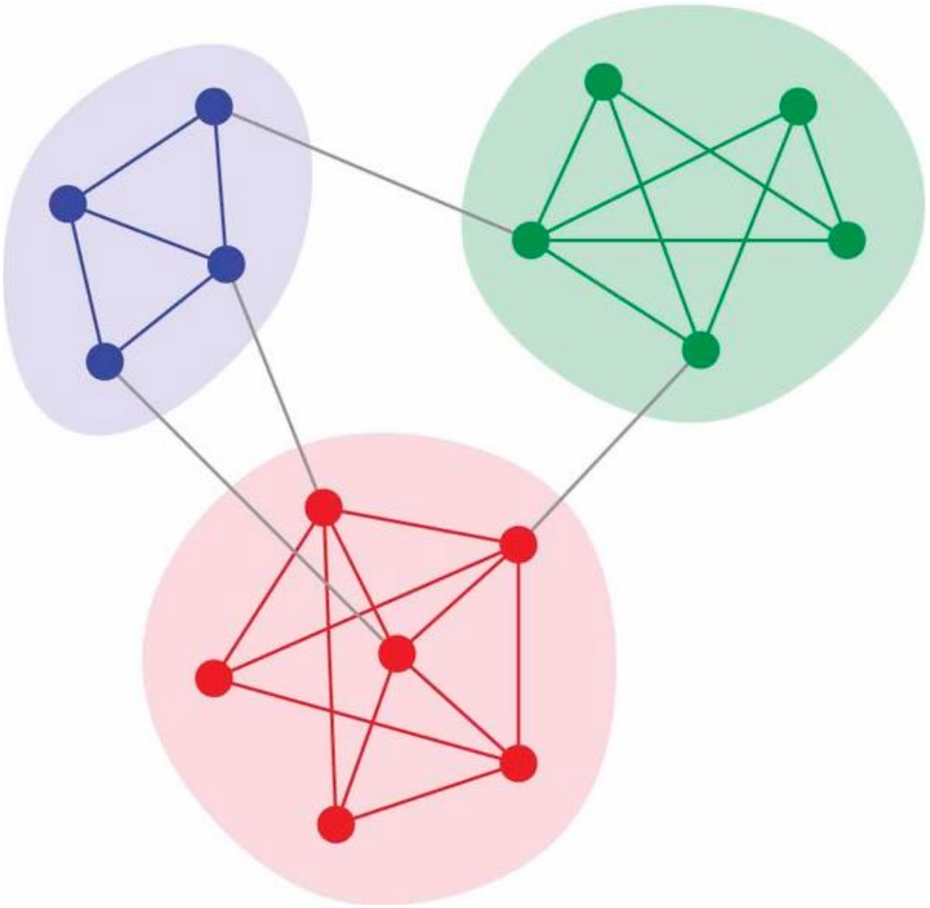
- From the analysis of small, independent communities to the analysis of communities within the giant component.



powered by ORA, CASOS Center @ CMU

Degree of Overlapping





Clustering, Community Detection, and Partitioning

- Clustering, community detection, and partitioning are three **related but distinct concepts** in the field of data analysis and graph theory.
  - They are all methods used to identify groups or structures within data, but they have different objectives and approaches.

- **Clustering**: The primary objective of clustering is to group similar data points or objects together based on their similarity or distance metrics.

- **Community Detection**: The primary objective of community detection is to identify densely connected communities within a network or graph.

- **Partitioning**: The primary objective of partitioning is to divide data or structures into two non-overlapping subsets, and it may not necessarily be based on similarity or connectivity.

Clustering, Community Detection, and Partitioning: Differences and Similarities

- **Grouping or division**
  - All three concepts involve some form of grouping or division.

- **Unsupervised methods**
  - All three concepts are typically based on unsupervised techniques, meaning they do not require labeled data for training or partitioning criteria.
    - Nowadays there is the possibility of employing supervised approaches to community detection.
    - Supervised community detection can be particularly useful in situations where you have some prior knowledge about the communities in the network or when you want to incorporate additional information into the detection process.

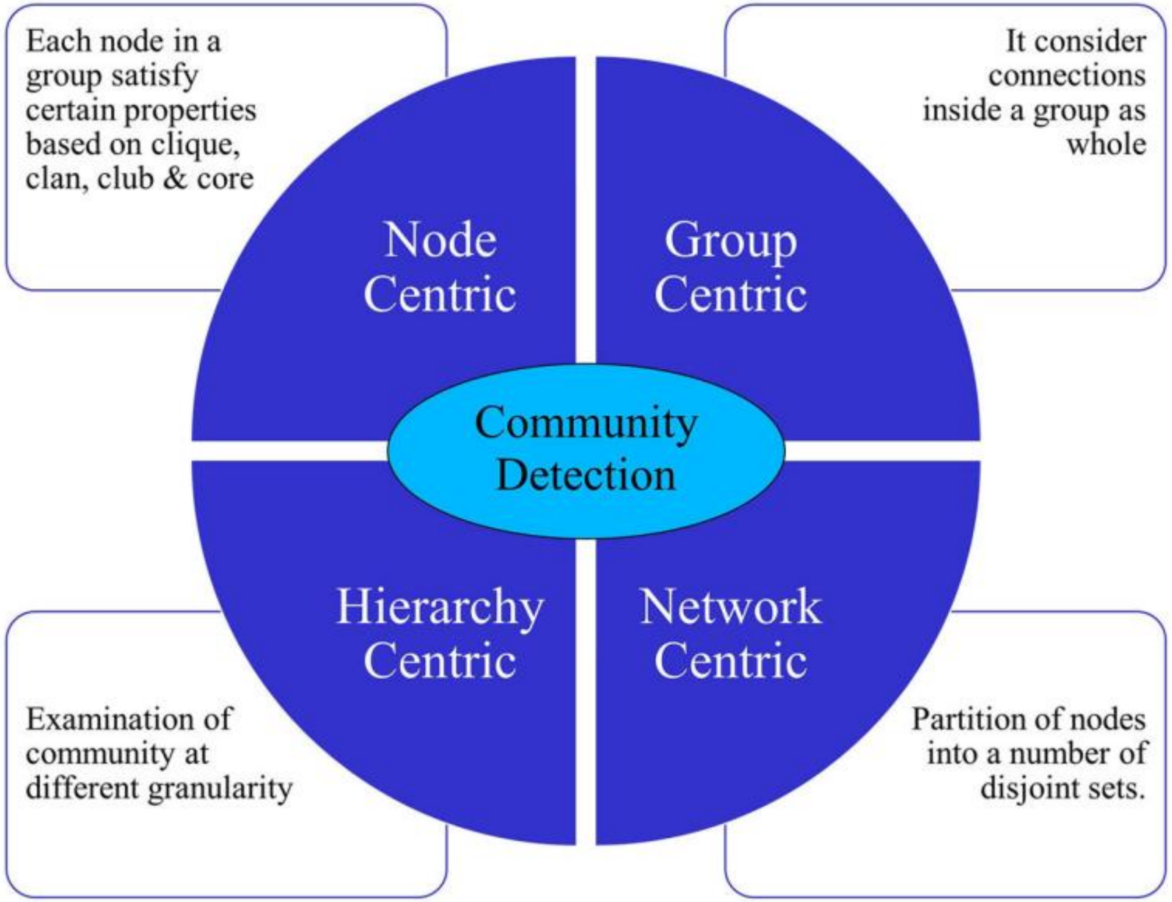
- **Graph theory connection**
  - Both community detection and partitioning often involve the use of graph theory principles when applied to networks or graphs.

Clustering VS Community Detection

- **Clustering**
  - Data is often NOT linked.
  - Clustering works on the distance similarity matrix, e.g., k-means.
  - By using the k-means with adjacency matrix rows, only the ego-centric network is considered.

- **Community detection**
  - Data is linked.
  - Algorithms use the graph property directly.

A Taxonomy of Community Detection (1/2)



https://doi.org/10.1007/s10115-022-01704-6

A Taxonomy of Community Detection (2/2)

- In **node-centric**, all the calculation is done based on a node like as node degree, node similarity, and node reachability.

- **Group-centric** is interested in the communities with particular group properties like balanced, robust, modular, and dense.
  - The group has to satisfy certain properties without zooming into node-level.

- In **network-centric**, partitions are based on the similarity of nodes where all the communities are disjoint.
  - The interest is referred to partition the whole network into several disjoint sets.

- In **hierarchy-centric**, all the groups are divided among levels where in top level all nodes are in same community, while in last level all nodes are in different communities.
  - A hierarchical structure of communities is built.

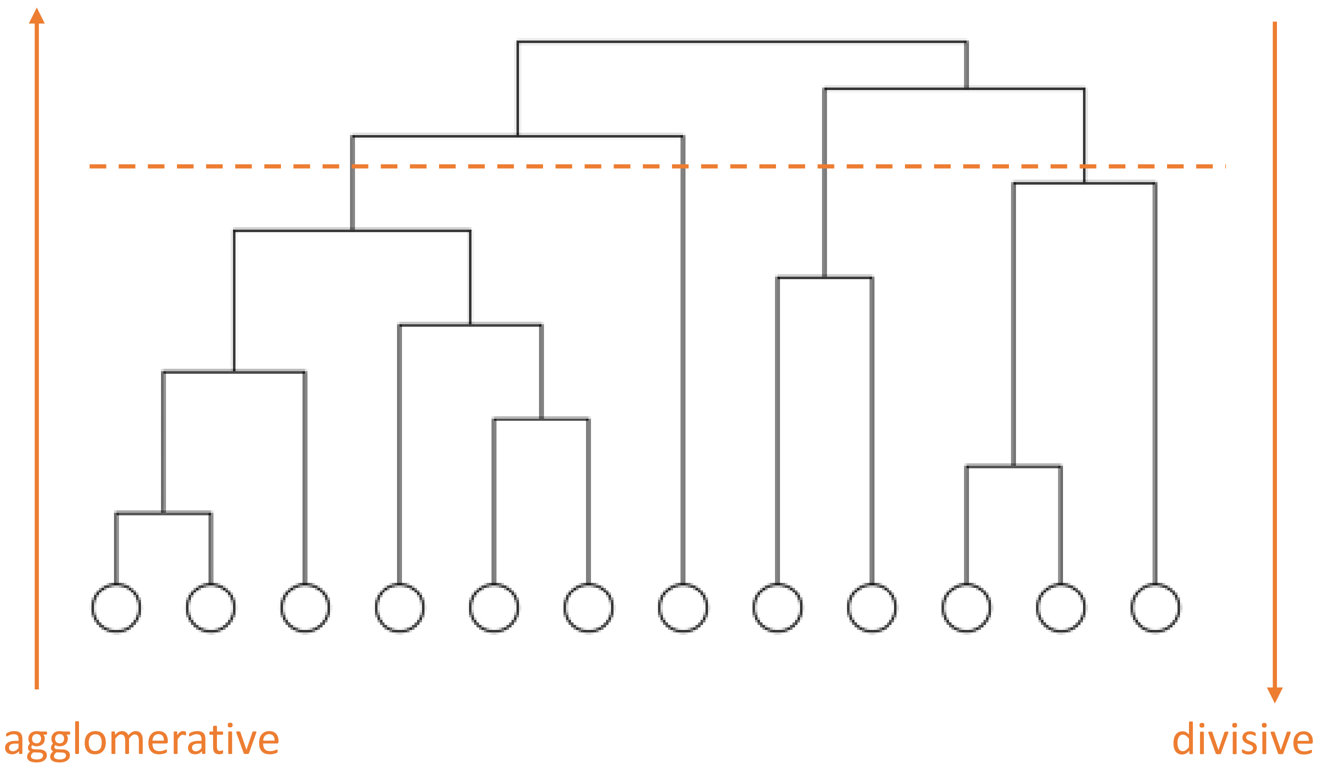
https://doi.org/10.1007/s10115-022-01704-6

Another Taxonomy of Community Detection

- **Hierarchical algorithms**
  - Agglomerative and divisive clustering
- **Modularity-based algorithms**
  - The Louvain and the Girvan-Newman algorithm
- **Label propagation-based algorithms**
  - Label propagation itself and LabelRank
- **Random walk-based algorithms**
  - Walktrap and Infomap
- **Graph partitioning-based algorithms**
  - Spectral clustering, the Kernighan-Lin algorithm, FluidC, METIS
- **Hybrid algorithms**

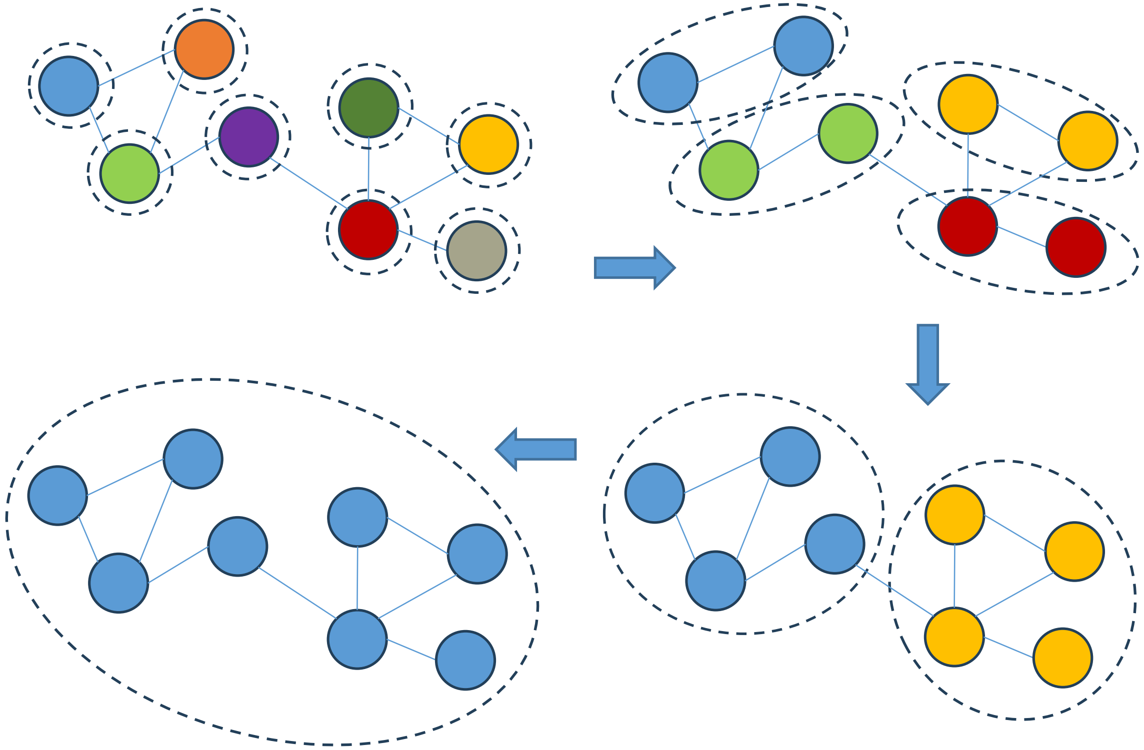
Hierarchical Algorithms

- **Goal**: build a hierarchical structure of communities based on network topology.
- Facilitate the analysis at different resolutions.
- Agglomerative VS divisive clustering.



Agglomerative Clustering

- Initialize each node as a community.
- Choose two communities satisfying certain criteria and merge them into larger ones.
  - Maximum node similarity.
  - Maximum modularity increase → Next slides.

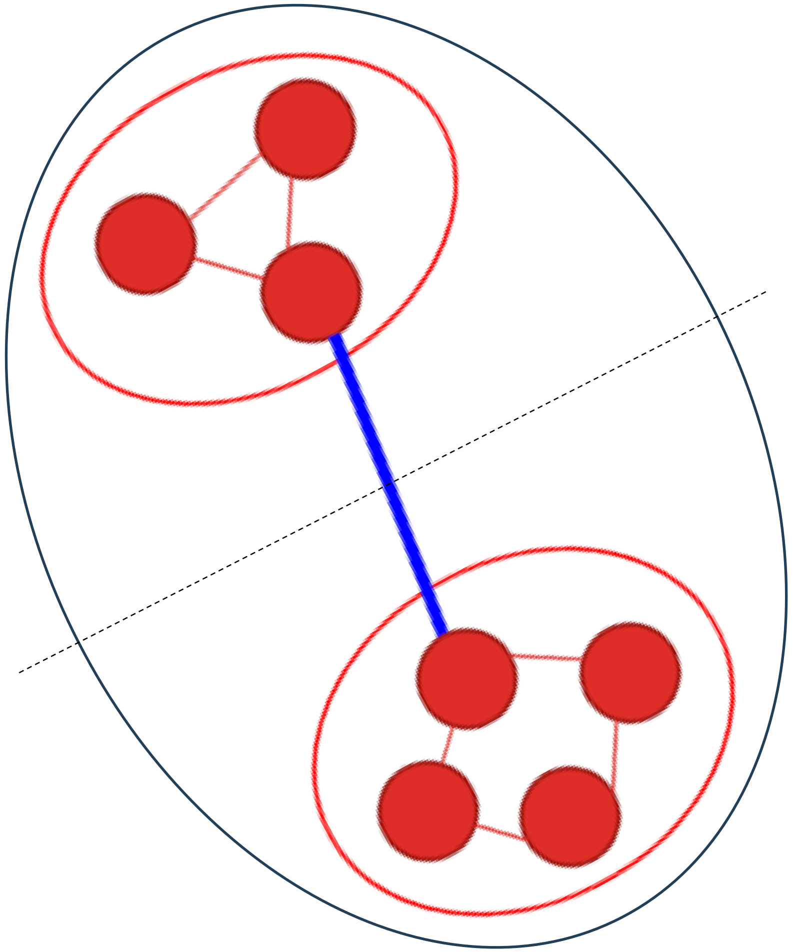


Divisive Clustering

- We start from a **unique community** and we partition it into smaller communities.

- Network-centric methods can be applied for partitioning.
  - The Girvan-Newman algorithm → Next slides.

- One particular example is based on edge-betweenness.
  - Edge-betweenness: Number of shortest paths between any pair of nodes that pass through the edge.
  - Between-group edges tend to have larger edge-betweenness.



Modularity-based Algorithms

- **Modularity** measures the strength of the community structure in a network by comparing the observed number of edges within communities to the expected number of edges if connections were distributed randomly.

- Well-known modularity-based algorithms include:
  - Louvain method (Louvain algorithm)
  - Girvan-Newman modularity (Girvan-Newman algorithm)

Modularity: Background

- Given a degree distribution, we know the expected number of edges between any pairs of vertices.
  - Equation in the next slides.

- We assume that real-world networks should be far from random.
  - Therefore, the more distant they are from this randomly generated network, the more structural they are.
  - Modularity defines this distance and modularity maximization tries to maximize this distance.

Modularity: Formal Definition (1/2)

- Let us consider two nodes i and j, with node degrees k_i and k_j, for a random network.

- The **expected number of edges** between two nodes, where m is the number of edges in the graph is:

$$\frac{k_i k_j}{2m}$$

- Hence, the difference between the actual number of edges A_{ij} between node i and j and the expected number of edges between them is:

$$A_{ij} - \frac{k_i k_j}{2m}$$

Modularity: Formal Definition (2/2)

- Summing over all node pairs gives the **equation for modularity**, denoted as Q:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

where

$$\delta(c_i, c_j) = \begin{cases} 1, & (i, j) \in c \\ 0, & \text{otherwise} \end{cases}$$

Modularity and Community Detection

- Modularity measures the quality of a community partition by comparing the actual number of connections within communities to what would be expected by chance.
  - When the observed connections within communities are significantly higher than expected, the modularity score is positive, indicating the presence of meaningful communities.
  - If it is close to zero, the network’s community structure is not well-defined.

- The modularity score (Q) typically falls within a range of -1 to 1. In particular:
  - $Q \approx 1$: A high positive modularity score indicates that the network's community structure is very pronounced and meaningful.
  - $Q \approx 0$: A modularity score close to zero means that the network's community structure is not significantly different from what you would expect by random chance.
  - $Q < 0$: A negative modularity score indicates that the network's community structure is less pronounced than what would be expected by random chance.

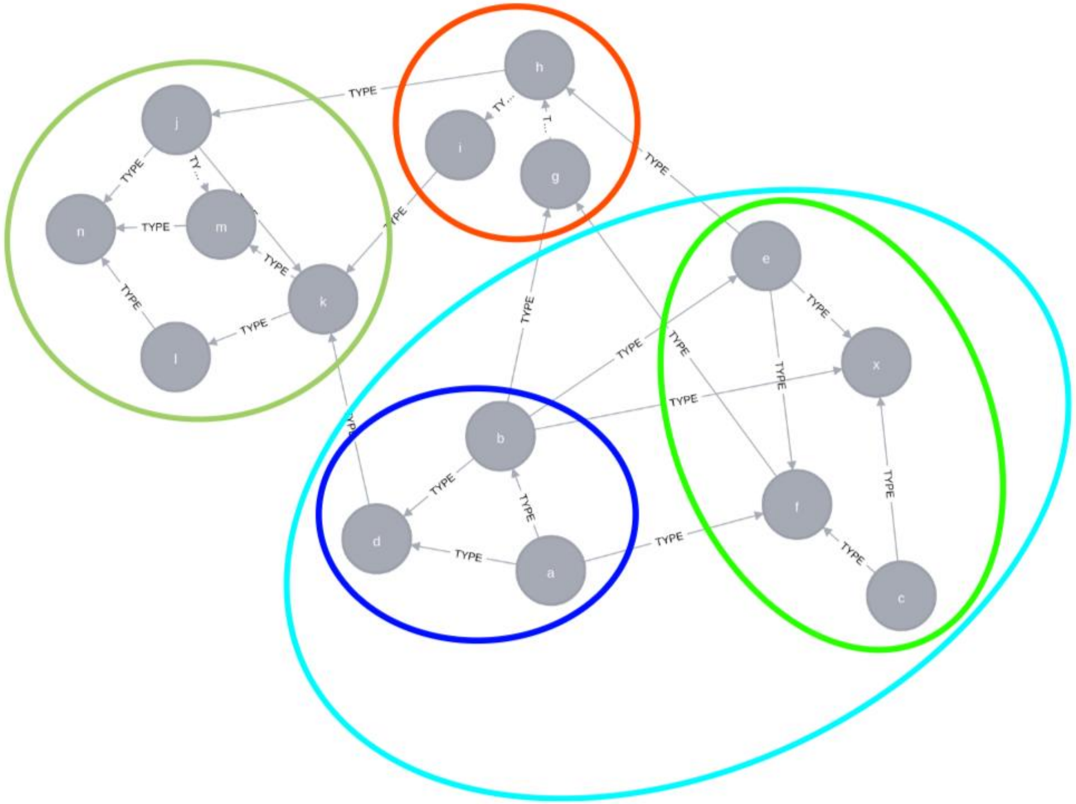
The Louvain Method (1/5)

- The **Louvain method for community detection** is a method to extract non-overlapping communities from large (undirected) networks.
  - Blondel, V. D., Guillaume, J. L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment, 2008(10), P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008
  - From the University of Louvain (the source of this method's name)

- The inspiration for this method of community detection is the optimization of modularity as the algorithm progresses.

The Louvain Method (2/5)

- In the Louvain method of community detection two phases are performed:
  1. First small communities are found by optimizing modularity locally on all nodes.
  2. Then each small community is grouped into one node and the first step is repeated.



- In this sense it is a hierarchical algorithm.

The Louvain Method (3/5)

- In the first phase (1.), two steps are repeated iteratively:
  a) First, each node in the network is assigned to its own community.
  b) Then for each node i, the change in modularity is calculated for removing i from its own community and moving it into the community of each neighbor j of i.

- The modularity of a community c can be calculated as:

We recall that:

$$Q = \frac{1}{2m}\sum_{vw}\left[A_{vw} - \frac{k_v k_w}{2m}\right]\delta(c_v, c_w)$$

$$Q_c = \frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2$$

- $\sum_{in}$: sum of edge weights between nodes within c (each edge is considered twice).
- $\sum_{tot}$: sum of all edge weights for nodes within c (also edges to other communities).

The Louvain Method (4/5)

- The equation for step (b.) is:

$$\Delta Q = \left[\frac{\Sigma_{in} + 2k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

- Σ_{in}: sum of all the weights of the links inside the community i is moving into.
- Σ_{tot}: sum of all the weights of the links to nodes in the community i is moving into.
- k_i: weighted degree of i.
- $k_{i,in}$: sum of the weights of the links between i and other nodes in the community that i is moving into.
- m: sum of the weights of all links in the network.

The Louvain Method (5/5)

- Once this value is calculated for all communities i is connected to:
  - i is placed into the community that resulted in the greatest modularity increase.
  - If no increase is possible, i remains in its original community.
  - This process is applied repeatedly to all nodes until no modularity increase can occur.
  - Once this local maximum of modularity is hit, the first phase has ended.

- In the second phase of the algorithm (2.):
  - It groups all of the nodes in the same community and builds a new network where nodes are the communities from the previous phase.
  - Any links between nodes of the same community are now represented by self-loops on the new community node and links from multiple nodes in the same community to a node in a different community are represented by weighted edges between communities.
  - Once the new network is created, the second phase has ended and the first phase can be re-applied to the new network.

The Girvan-Newman Algorithm (1/3)

- The **Girvan–Newman algorithm** is a hierarchical method used to detect communities in complex systems.
  - Named after Michelle Girvan and Mark Newman.
  - Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. Proceedings of the national academy of sciences, 99(12), 7821-7826. https://doi.org/10.1073/pnas.122653799
  - Originally designed for non-overlapping communities in undirected graphs.

- The Girvan–Newman algorithm detects communities by progressively removing edges from the original network.
  - The connected components of the remaining network are the communities.

- Instead of trying to construct a measure that tells us which edges are the most central to communities, the Girvan–Newman algorithm focuses on edges that are most likely “between” communities.

The Girvan-Newman Algorithm (2/3)

1. **Calculate betweenness centrality**
   - Compute the betweenness centrality for all edges in the network.

2. **Edge removal**
   - Identify the edge with the highest betweenness centrality and remove it.
   - This process disrupts the most central connections in the network.

3. **Recalculate betweenness centrality**
   - Recalculate the betweenness centrality for all remaining edges.
   - The removal of an edge affects the centrality of other edges, so this step is necessary to update the centrality values.

The Girvan-Newman Algorithm (3/3)

4. **Repeat**
   - Repeat steps 2 and 3 until a certain criterion is met.
   - This criterion could be the desired number of communities, a specific modularity threshold, or the absence of edges.

5. **Community detection**
   - The resulting disconnected components (subgraphs) after edge removal are considered as communities.
   - The number of communities is determined based on the stopping criterion.

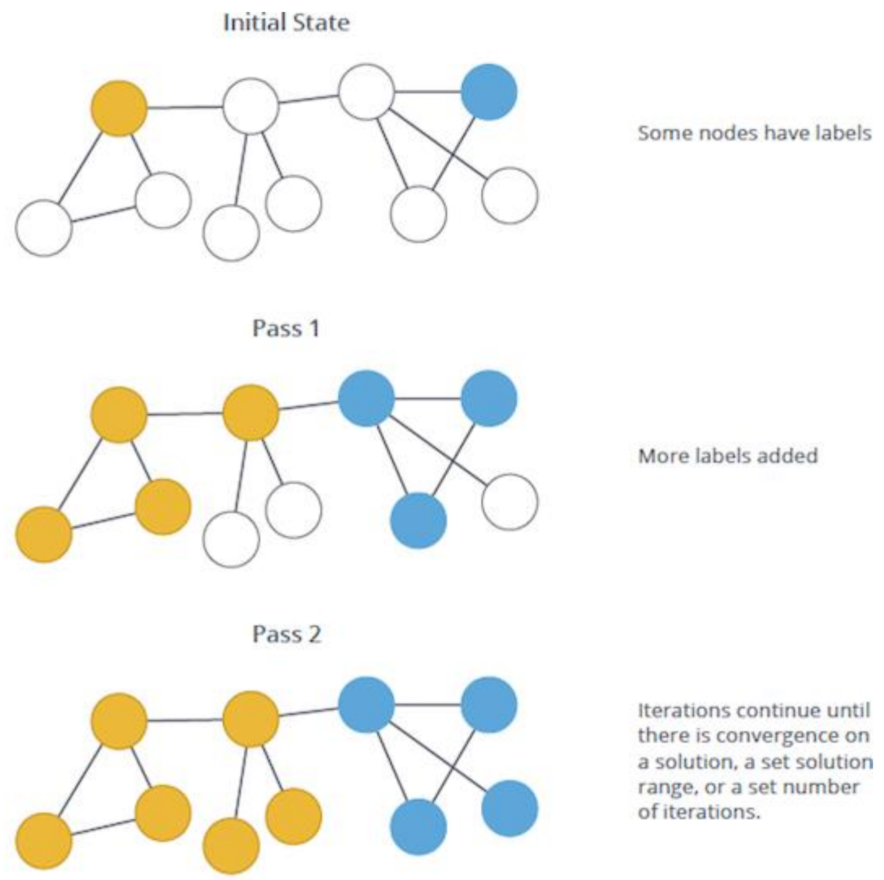
Label Propagation

- **Label propagation** is a semi-supervised machine learning algorithm that assigns labels to previously unlabeled data points.
  - At the start of the algorithm, a (generally small) subset of the data points have labels.
  - These labels are propagated to the unlabeled points throughout the course of the algorithm.
  - Zhu, X., & Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=8a6a114d699824b678325766be195b0e7b564705

The Label Propagation Algorithm (LPA) (1/2)

- **Within complex networks**, real networks tend to have community structure.
    - Label propagation is an algorithm for finding communities.
    - Raghavan, U. N., Albert, R., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. Physical review E, 76(3), 036106. https://doi.org/10.1103/PhysRevE.76.036106

- The original Label Propagation Algorithm (LPA) is designed for undirected graphs and primarily focuses on non-overlapping communities.



Label Propagation Algorithm

The Label Propagation Algorithm (LPA) (2/2)

1. **Initialization**
   - I start with a situation in which some nodes have labels, OR
   - *Assign a unique label to each node in the network*.

2. **Label Propagation**
   - Iteratively update the labels of nodes based on a given criterion for each node i:
     - Count the occurrences of each label among its neighbors.
     - Assign the label with the highest count to node i.
     - *In the case of assigning unique labels, another criterion can be selected*.
     - In the case of a tie, randomly select one of the tied labels.

3. **Convergence Check**
   - Check for convergence, i.e., whether the labels have stabilized or not.
     - If the labels have not stabilized, repeat the label propagation step.
     - If the labels have stabilized, proceed to the next step.

4. **Community Formation**
   - Form communities based on the final labels assigned to nodes.

Random Walk

- Let $G = (V, E, w)$ be a weighted undirected graph.

- A **random walk** on a graph is a process that begins at some vertex, and at each time step moves to another vertex.
  - When the graph is unweighted, the vertex the walk moves to is chosen uniformly at random among the neighbors of the present vertex.
  - When the graph is weighted, it moves to a neighbor with probability proportional to the weight of the corresponding edge.

- **Random walk-based algorithms**
  - Walktrap
    - Pons, P., & Latapy, M. (2005). Computing communities in large networks using random walks. In Computer and Information Sciences-ISCIS 2005: 20th International Symposium, Istanbul, Turkey, October 26-28, 2005. Proceedings 20 (pp. 284-293). Springer Berlin Heidelberg. https://doi.org/10.1007/11569596_31
  - Infomap

Random Walk-based Algorithms

- **Walktrap**
  - Pons, P., & Latapy, M. (2005). Computing communities in large networks using random walks. In Computer and Information Sciences-ISCIS 2005: 20th International Symposium, Istanbul, Turkey, October 26-28, 2005. Proceedings 20 (pp. 284-293). Springer Berlin Heidelberg. https://doi.org/10.1007/11569596_31
  - The original Walktrap algorithm, as proposed by Pascal Pons and Matthieu Latapy, is designed for undirected graphs and non-overlapping communities.

- Infomap
  - Rosvall, M., Axelsson, D., & Bergstrom, C. T. (2009). The map equation. The European Physical Journal Special Topics, 178(1), 13-23. https://doi.org/10.1140/epjst/e2010-01179-1

Walktrap

1. **Random walks**
   - The algorithm simulates random walks on the network, where a walker traverses the graph by moving to a neighboring node at each step.

2. **Similarity**
   - The algorithm measures the similarity between nodes based on the frequency with which random walks starting from those nodes intersect.
   - Nodes that are often visited together are considered similar.

3. **Community detection**
   - The similarity information is used to group nodes into communities.
   - Nodes with high similarity are more likely to belong to the same community.

4. **Agglomerative approach**
   - The algorithm uses an agglomerative approach, starting with each node in its own community and progressively merging communities based on the similarity of their nodes.

Graph Partitioning

- **Graph partitioning** is the process of dividing a graph into smaller subgraphs by partitioning its set of nodes into mutually exclusive groups.

- Edges of the original graph that cross between the groups will produce edges in the partitioned graph.

- The goal of graph partitioning is to simplify graph analysis by reducing the size of the graph while preserving its essential properties.

Graph Partitioning-based Algorithms

- Spectral clustering
- **The Kernighan-Lin algorithm**
- **FluidC**
- **METIS**

The Kernighan-Lin Algorithm (1/2)

- The **Kernighan–Lin algorithm** is a heuristic algorithm for finding partitions of graphs.
  - Kernighan, B. W., & Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. The Bell system technical journal, 49(2), 291-307. https://doi.org/10.1002/j.1538-7305.1970.tb01770.x

- **Problem**. Divide a weighted graph with $2n$ nodes into two parts, each of size n, to minimize the cut size (or sum of the weights) crossing the two parts.
  - Given a graph, a "cut" refers to a partition of the nodes into two (or more) disjoint subsets.
  - The "cut size" is the number of edges that have one endpoint in one subset and the other endpoint in the other subset.
  - In other words, it's the number of edges that cross the partition boundary.

The Kernighan-Lin Algorithm (2/2)

1. **Initialization**
   - Start with an initial partition of the graph into two subsets.
   - This initial partition could be random or based on some heuristic.

2. **Iterative refinement and convergence**
   - The partition is iteratively refined by swapping nodes between subsets to improve cut size.
   - For each iteration:
     a) Compute initial gain: calculate the gain for moving each node from its current subset to the other subset. The gain is the reduction in cut size if the node is moved.
     b) Node pair selection: select a pair of nodes (one from each subset) based on their initial gains. This is typically done by choosing the pair with the highest combined gain.
     c) Swap nodes: swap the selected pair of nodes between the subsets.
     d) Update gains: recalculate gains for the affected nodes. The gains for nodes that were moved are updated, as well as gains for their neighbors.
     e) Repeat: Repeat the process until no more improvement can be achieved.
   - The algorithm converges when no more swaps can be made that result in a reduction in the cut size.

FluidC (1/4)

- **FluidC** is an algorithm inspired from LPA.
  - https://www.arxiv-vanity.com/papers/1703.09307/

- The algorithm tries to mimic the behaviour of several fluids (i.e., communities) expanding and pushing one another in a shared, closed environment (i.e., graph), until an equilibrium state found.
  - A fluid community will conquer parts of the environment which have a favorable topology (i.e., which are strongly connected with its vertices) while losing some parts to other fluid communities.

- One of the most relevant features of FluidC is that it can find any number of communities in a graph (e.g., one may specify the number of communities FluidC must find) simply by initializing that number of fluids.

FluidC (2/4)

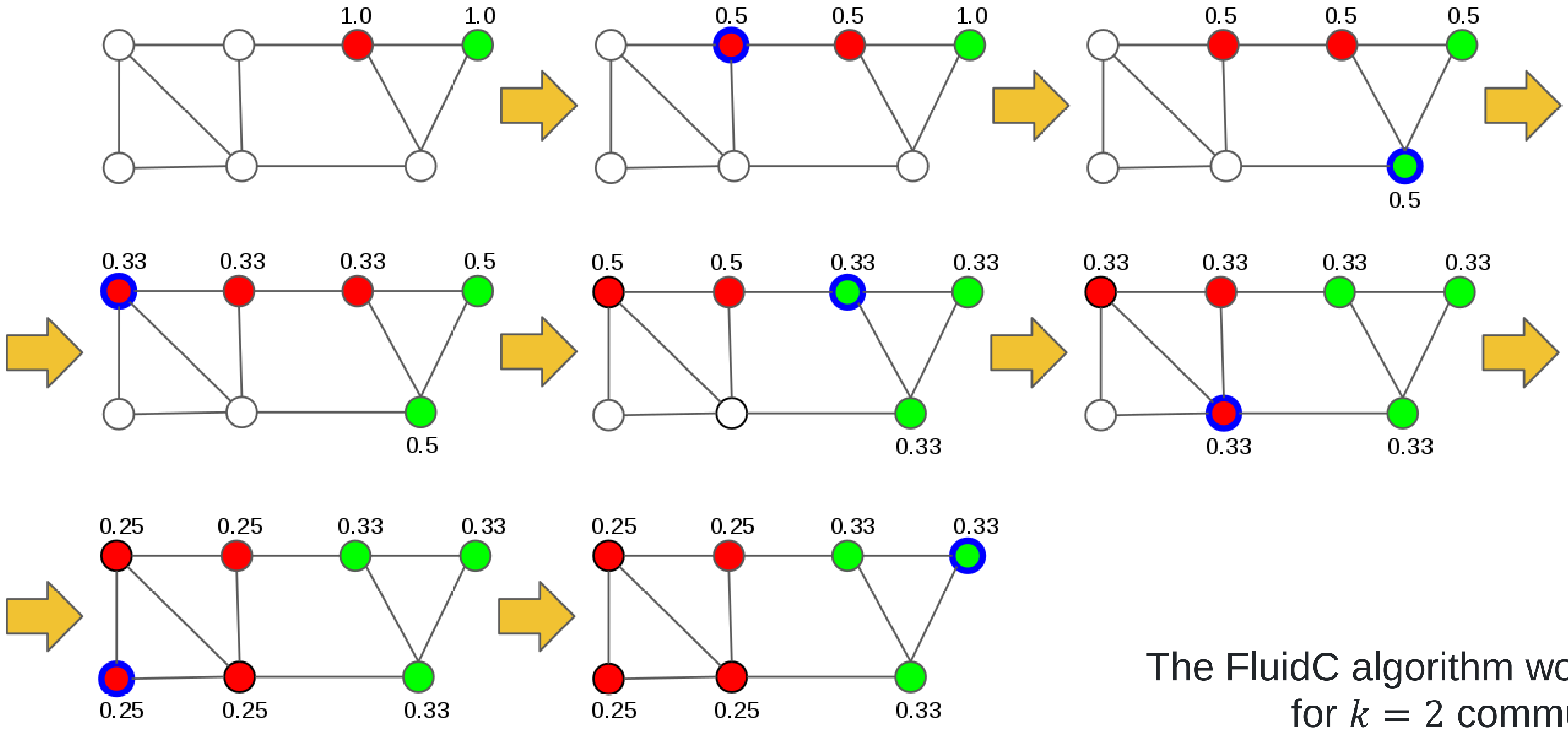
- Consider a graph $G = (V, E)$ formed by a set of vertices V and a set of edges E.
- FluidC initializes k fluid communities on k different vertices of V, communities that will begin expanding throughout the graph.
- At all times, each fluid community λ has a total density of 1.0.
- When a fluid community is compacted into a single vertex (e.g., at initialization), such vertex holds the full community density (i.e., 1.0), which is also the maximum density a single vertex may ever have.
  - As a community spans through multiple vertices, its density becomes evenly distributed among the vertices that compose it.

FluidC (3/4)

- The **FluidC workflow** follows the propagation approach introduced by LPA.

- **On each step**, FluidC iterates over all vertices in random order, updating the community each vertex belongs to using an update rule.
  - Simply put, the update rule sums the densities of a vertex neighbours community-wise, including itself, and returns the community with maximum density.
    - If the maximum density is shared by two or more communities but the previous community of the vertex is not among those, a random one is chosen.
    - If the previous community of the vertex is among the set of communities with maximum density, the vertex keeps its previous community.

FluidC (4/4)



The FluidC algorithm workflow
for $k = 2$ communities

METIS

- The **METIS algorithm** allows to obtain two very balanced communities.
  - Karypis, G., & Kumar, V. (1995). Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0. University of Minnesota. https://www.researchgate.net/publication/246815679_METIS_--_Unstructured_Graph_Partitioning_and_Sparse_Matrix_Ordering_System_Version_20
  - In METIS, it is possible to define the number of required communities.
  - The algorithm is very efficient with respect to the required computational time.

METIS

- METIS is aimed at partitioning undirected graphs, according to the topological characteristics of the network.

- Partitioning is based on a so-called multilevel graph bisection.
  - It implies a progressive reduction of the graph, with a subsequent "regrowth" to its original size.