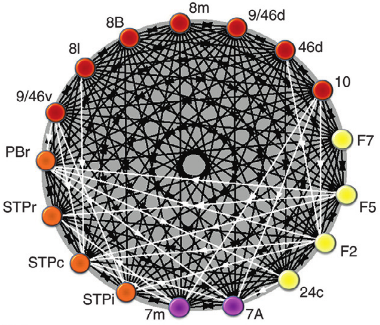
CS 4884: Components, Cliques, and Cores

T. M. Murali

February 20 and 23, 2020



Summary of Course Thus Far

- History of neuroscience
- Graphs (Definitions, basic concepts, Euler tours)
- Brain graphs (types of nodes and edges, experimental methods, Chapter 2)
- Brain connectivity matrices and node degrees (Chapters 3 and 4)
- Shortest paths (Chapter 7.1 and 7.2)
- Clustering coefficient and small world networks (Chapter 8.2)

Plan till Spring Break

- Clustering coefficient is a local measure of graph density.
- Small world measures capture global features of graphs.

Plan till Spring Break

- Clustering coefficient is a local measure of graph density.
- Small world measures capture global features of graphs.

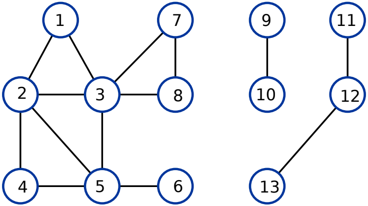
Are there intermediate notions of graph density?

- Subgraphs that represent backbones of network topology (components, shortest paths, cores, Chapter 6.1, 6.2, 7.1, February 20 and 25)
- Modularity (Chapter 9.1, February 25, 27, March 3, 5)

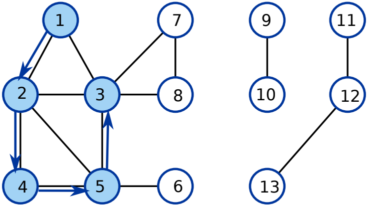
Plan after Spring Break

- Schedule meetings with project groups during class time in my office.
- Number of meetings per group will depend on number of groups.
- Poster preparation for VTURCS Symposium on April 28.

Paths and Connectivity

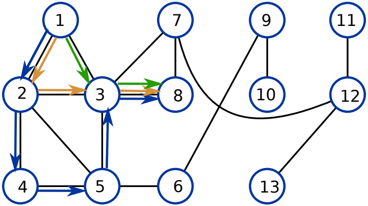


Paths and Connectivity



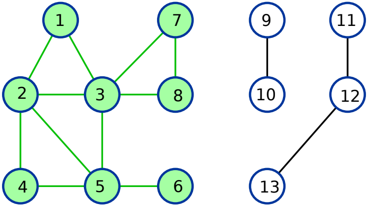
- A *v_1-v_k path* in an undirected graph $G = (V, E)$ is a sequence P of nodes $v_1, v_2, \ldots, v_{k-1}, v_k \in V$ such that every consecutive pair of nodes $v_i, v_{i+1}, 1 \leq i < k$ is connected by an edge in E.

Paths and Connectivity



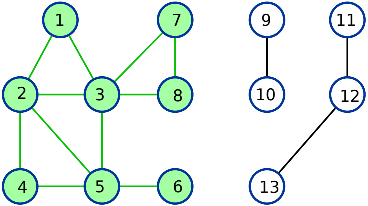
- A *v_1-v_k path* in an undirected graph $G = (V, E)$ is a sequence P of nodes $v_1, v_2, \ldots, v_{k-1}, v_k \in V$ such that every consecutive pair of nodes $v_i, v_{i+1}, 1 \leq i < k$ is connected by an edge in E.
- *Distance $\delta(u, v)$* between two nodes u and v is the minimum number of edges in any u-v path.

Paths and Connectivity



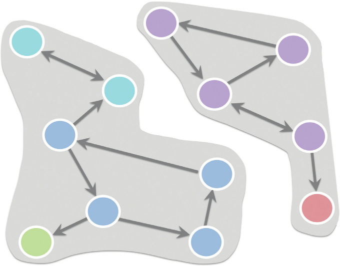
- A *v_1-v_k path* in an undirected graph $G = (V, E)$ is a sequence P of nodes $v_1, v_2, \ldots, v_{k-1}, v_k \in V$ such that every consecutive pair of nodes $v_i, v_{i+1}, 1 \leq i < k$ is connected by an edge in E.
- *Distance $\delta(u, v)$* between two nodes u and v is the minimum number of edges in any u-v path.
- A *connected component* of G is a subgraph $H = (V', E')$ of G such
  - for every pair of nodes u, v in V' there is a u-v path in H, i.e., that uses only the edges in E' and

Paths and Connectivity



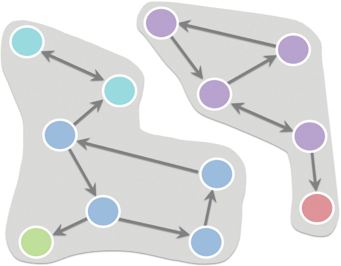
- A *v_1-v_k path* in an undirected graph $G = (V, E)$ is a sequence P of nodes $v_1, v_2, \ldots, v_{k-1}, v_k \in V$ such that every consecutive pair of nodes $v_i, v_{i+1}, 1 \leq i < k$ is connected by an edge in E.
- *Distance $\delta(u, v)$* between two nodes u and v is the minimum number of edges in any u-v path.
- A *connected component* of G is a subgraph $H = (V', E')$ of G such
  - for every pair of nodes u, v in V' there is a u-v path in H, i.e., that uses only the edges in E' and
  - H is *maximal*, i.e., for every node $x \in V - V'$, there is no path in G between x and any node in V'.

Connected Components in Directed Graphs



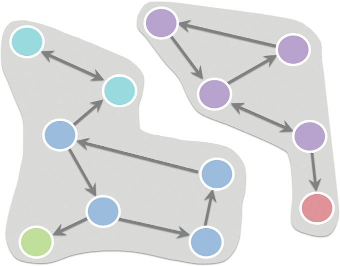
- In directed graphs, connectivity is not symmetric.

Connected Components in Directed Graphs



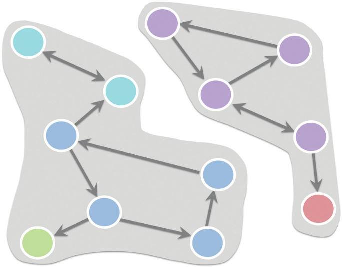
- In directed graphs, connectivity is not symmetric.
- A *weakly connected component* of a directed graph G is a connected component of the undirected graph G' obtained by replacing every edge in G by an undirected edge.

Connected Components in Directed Graphs



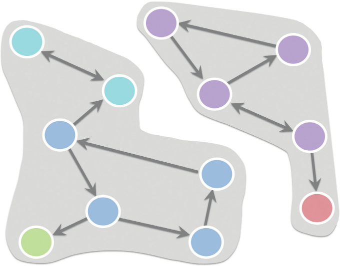
- In directed graphs, connectivity is not symmetric.
- A *weakly connected component* of a directed graph G is a connected component of the undirected graph G' obtained by replacing every edge in G by an undirected edge.
- We can compute all weakly connected components in linear time.

Connected Components in Directed Graphs



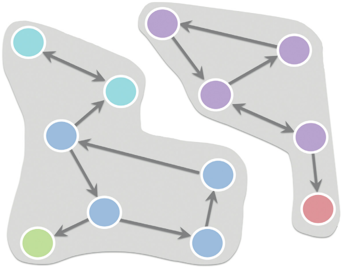
- In directed graphs, connectivity is not symmetric.
- A *strongly connected component* of a directed graph $G = (V, E)$ is a subgraph $H = (V', E')$ of G such

Connected Components in Directed Graphs



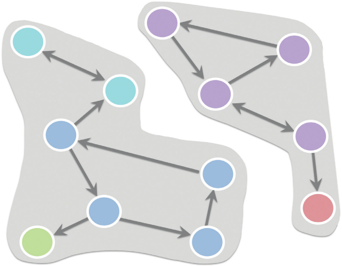
- In directed graphs, connectivity is not symmetric.
- A *strongly connected component* of a directed graph $G = (V, E)$ is a subgraph $H = (V', E')$ of G such
  - for every pair of nodes u, v in V' there is a u-to-v path and a v-to-u path in H, i.e., that use only the edges in E' and

Connected Components in Directed Graphs



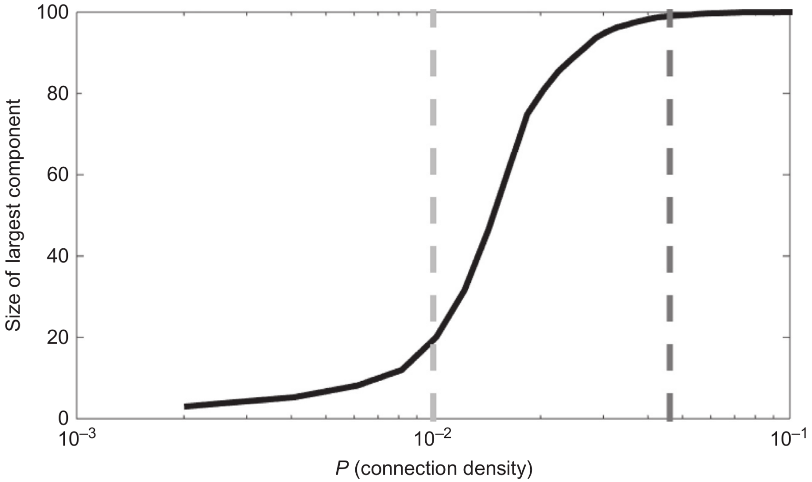
- In directed graphs, connectivity is not symmetric.
- A *strongly connected component* of a directed graph $G = (V, E)$ is a subgraph $H = (V', E')$ of G such
  - for every pair of nodes u, v in V' there is a u-to-v path and a v-to-u path in H, i.e., that use only the edges in E' and
  - H is *maximal*, i.e., for every node $x \in V - V'$, there is at least one node $y \in V'$ such that there is no path in G from x to y or from y to x.

Connected Components in Directed Graphs



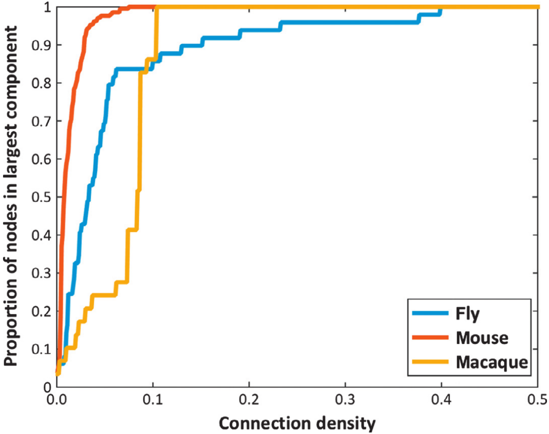
- In directed graphs, connectivity is not symmetric.
- A *strongly connected component* of a directed graph $G = (V, E)$ is a subgraph $H = (V', E')$ of G such
  - for every pair of nodes u, v in V' there is a u-to-v path and a v-to-u path in H, i.e., that use only the edges in E' and
  - H is *maximal*, i.e., for every node $x \in V - V'$, there is at least one node $y \in V'$ such that there is no path in G from x to y or from y to x.
- We can compute all strongly connected components in linear time using DFS with some tricks.

Largest Component in Brain Graphs



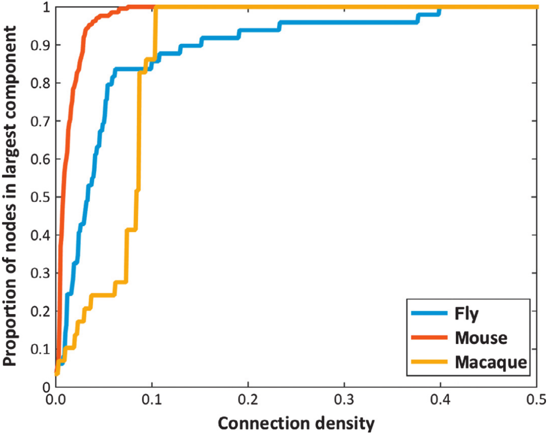
- Phase transition for appearance of large component in E-R graphs.

Largest Component in Brain Graphs



- Add edges in decreasing order of weight.
- Plot the size of the largest weakly connected component.

Largest Component in Brain Graphs



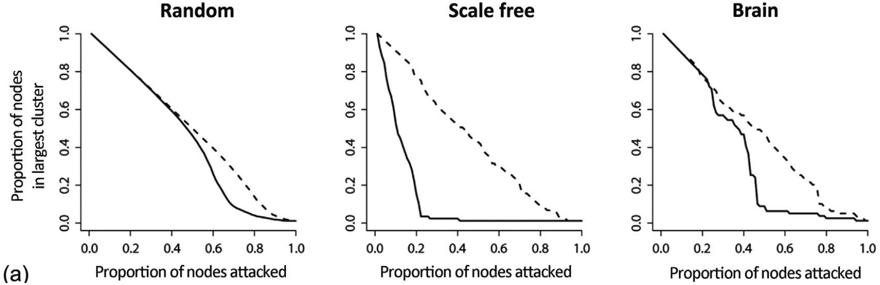
- Add edges in decreasing order of weight.
- Plot the size of the largest weakly connected component.
- Size of the largest component increases rapidly as a function of increasing network density.

Random and Targeted Attack on Brain Networks

- Remove nodes randomly.
- Targeted attack: Remove nodes in decreasing order of degree.

Random and Targeted Attack on Brain Networks

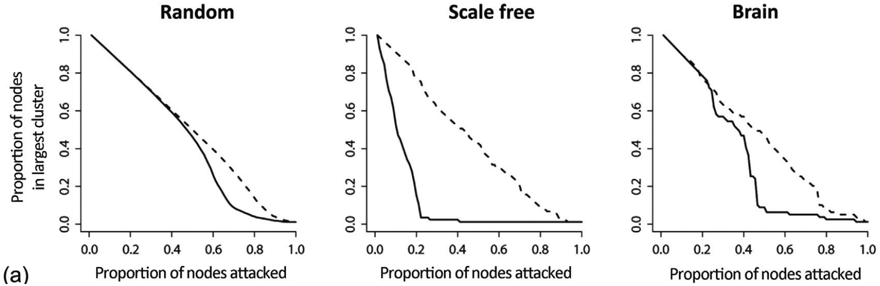
- Remove nodes randomly.
- Targeted attack: Remove nodes in decreasing order of degree.



$$\Pr\{\text{degree } = k\} \sim k^{-\gamma} \qquad \sim k^{-\gamma} e^{-k/k_c}$$

Random and Targeted Attack on Brain Networks

- Remove nodes randomly.
- Targeted attack: Remove nodes in decreasing order of degree.



$$\Pr\{\text{degree } = k\} \sim k^{-\gamma} \qquad \sim k^{-\gamma}e^{-k/k_c}$$

- Degree distribution of the brain is broad-scale: characterized by an exponentially-truncated power law.
- Concentration of links on hub nodes is weaker in a broad-scale network compared to a scale-free network.

Graph Measures Based on Shortest Paths

- *Characteristic path length* $l(G)$ is the average shortest path length between all pairs of nodes in G. $\delta(u, v)$ = shortest path length from u to v.

$$l(G) = \frac{1}{n(n-1)} \sum_{u,v \in V, u \neq v} \delta(u, v)$$

Graph Measures Based on Shortest Paths

- *Characteristic path length* $l(G)$ is the average shortest path length between all pairs of nodes in G. $\delta(u, v)$ = shortest path length from u to v.

$$l(G) = \frac{1}{n(n-1)} \sum_{u,v \in V, u \neq v} \delta(u, v)$$

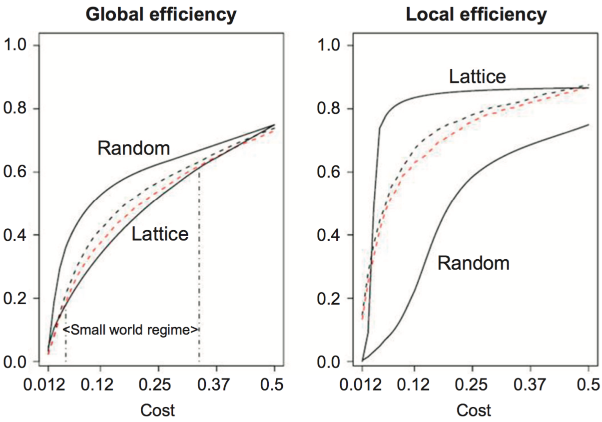
- *Global efficiency* $e_{\text{glob}}(G)$ is the average of the reciprocal of the shortest path length between all pairs of nodes in G.

$$e_{\text{glob}}(G) = \frac{1}{n(n-1)} \sum_{u,v \in V, u \neq v} \frac{1}{\delta(u, v)}$$

- *Local efficiency* $e_{\text{loc}}(v)$ of a node v is the average of the reciprocal of the shortest path length between all pairs of neighbours of v in G.

$$e_{\text{loc}}(v) = \frac{1}{d(v)(d(v)-1)} \sum_{\substack{u,v \in N(v) \\ u \neq v}} \frac{1}{\delta(u, v)}$$

Efficiency in Brain Networks



- Functional connectivity networks from fMRI data in young (black) and old (orange) human volunteers.
- x-axis is fraction of possible edges as threshold on edge weight varies.
- y-axis is global (left) and local (right) efficiency.
- Small world networks are both locally and globally efficient.

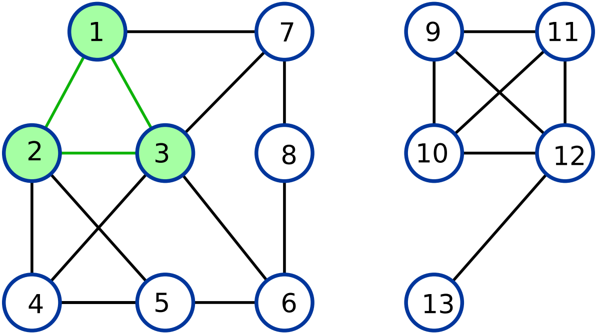
Motivation for Modules

- Connected components offer a fairly coarse description of the core of a network.
- Most brain networks contain one large component that spans most nodes.
- Therefore, analysis of connected components does not identify sets of nodes/edges that act as critical backbones or information-processing cores.

Core-Periphery Architecture

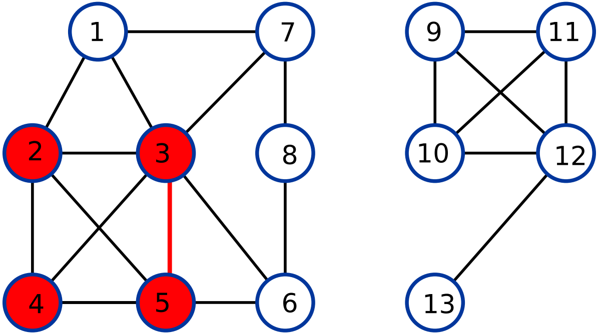
- Core nodes should occupy a topologically central position in the network.
- Nodes in the core should be highly interconnected with each other.
- Peripheral nodes should be moderately connected to core nodes, but sparsely interconnected with each other.

Defining Modules



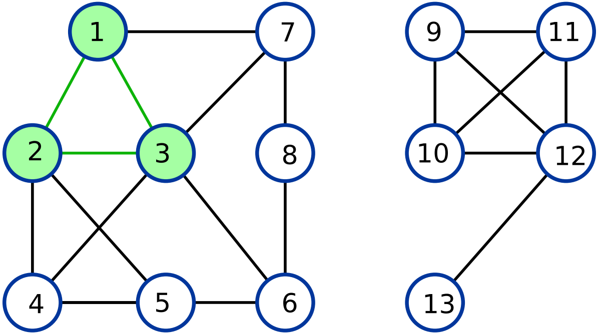
- How do we define a module in an undirected graph?
- In an undirected graph $G = (V, E)$, a subset of nodes $C \subseteq V$ is a *clique* or *complete subgraph* if for every pair of nodes $u, v \in C$, (u, v) is an edge in E.

Defining Modules



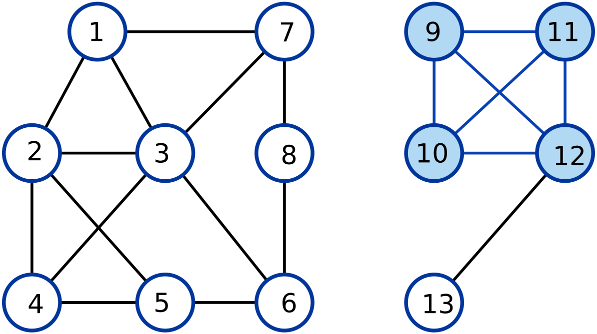
- How do we define a module in an undirected graph?
- In an undirected graph $G = (V, E)$, a subset of nodes $C \subseteq V$ is a *clique* or *complete subgraph* if for every pair of nodes $u, v \in C$, (u, v) is an edge in E.

Defining Modules



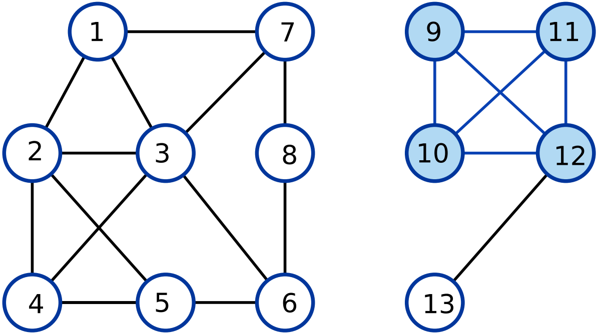
- How do we define a module in an undirected graph?
- In an undirected graph $G = (V, E)$, a subset of nodes $C \subseteq V$ is a *clique* or *complete subgraph* if for every pair of nodes $u, v \in C$, (u, v) is an edge in E.
  - A clique C is *maximal* if no node outside C can be added to it, i.e., for every node $x \in V - C$, x is not connected to at least one node in C.

Defining Modules



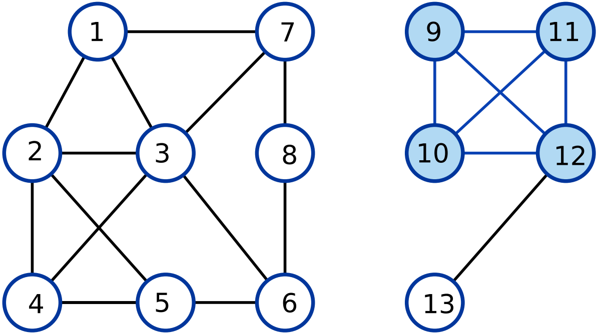
- How do we define a module in an undirected graph?
- In an undirected graph $G = (V, E)$, a subset of nodes $C \subseteq V$ is a *clique* or *complete subgraph* if for every pair of nodes $u, v \in C$, (u, v) is an edge in E.
  - A clique C is *maximal* if no node outside C can be added to it, i.e., for every node $x \in V - C$, x is not connected to at least one node in C.
  - A clique C is *maximum* if there is no clique C' in G with more nodes than C.

Computing a Maximum Clique



Maximum Clique
Given an undirected, unweighted graph $G(V, E)$, compute the largest clique in G.

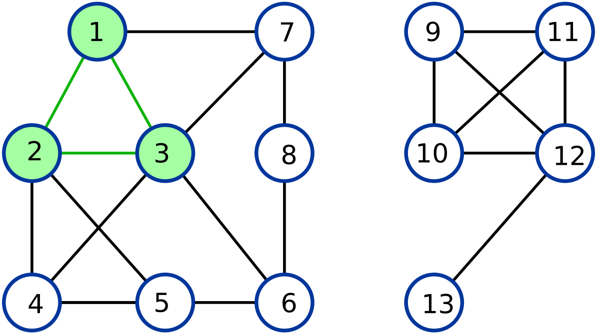
Computing a Maximum Clique



MAXIMUM CLIQUE
Given an undirected, unweighted graph $G(V, E)$,
compute the largest clique in G.

- Computing a maximum clique is NP-hard.
- Any algorithm that can provably compute the maximum clique is likely to have a running time that is exponential in the size of the graph.

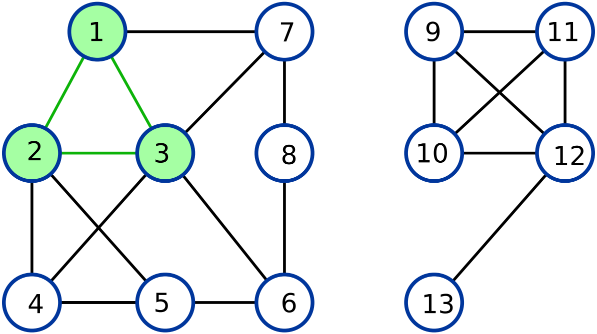
Computing a Maximal Clique



Maximal Clique

Given an undirected, unweighted graph $G(V, E)$,
compute a maximal clique in G.

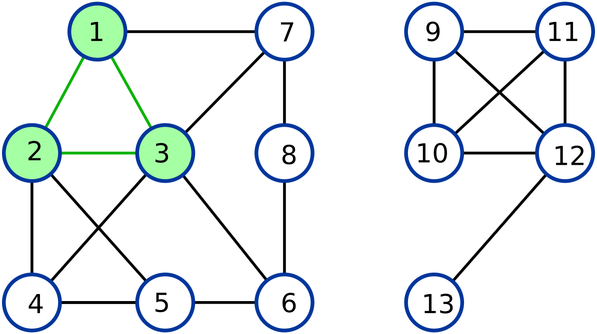
Computing a Maximal Clique



Maximal Clique
Given an undirected, unweighted graph $G(V, E)$,
compute a maximal clique in G.

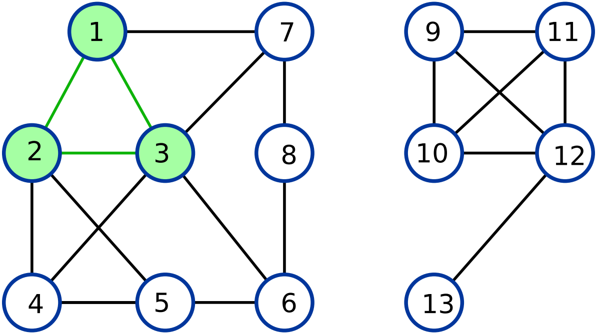
1. Select an arbitrary node v and add it to S (the clique we will output).
2. If there is a node u in $V - S$ that is connected to every node in S, add u to S.
3. Repeat the previous step until no such node u is found.

Running Time to Compute a Maximal Clique



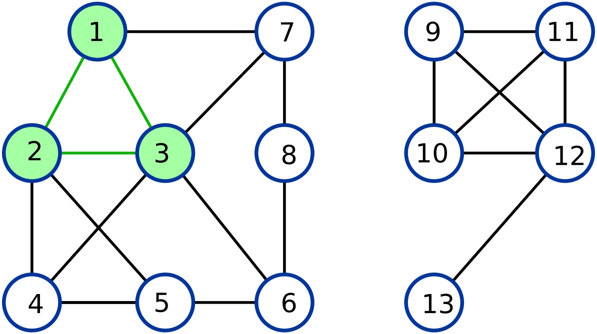
1. Select an arbitrary node v and add it to S (the clique we will output).
2. If there is a node u in $V - S$ that is connected to every node in S, add u to S.
3. Repeat the previous step until no such node u is found.

Running Time to Compute a Maximal Clique



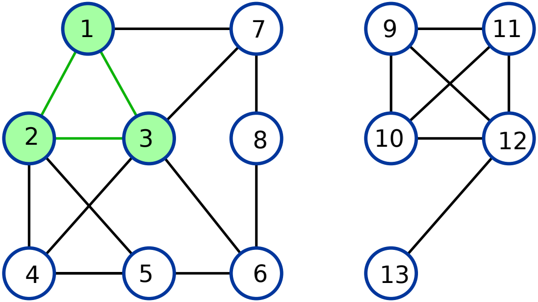
1. Select an arbitrary node v and add it to S (the clique we will output).
2. If there is a node u in $V - S$ that is connected to every node in S, add u to S. $O(n|S|)$ checks for edge existence.
3. Repeat the previous step until no such node u is found.

Running Time to Compute a Maximal Clique



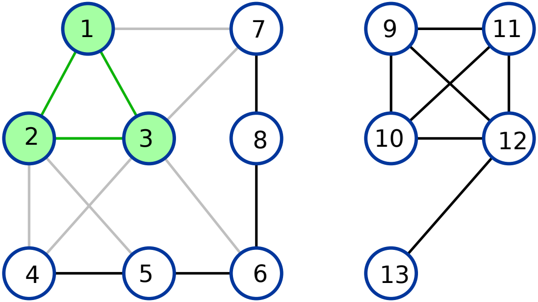
1. Select an arbitrary node v and add it to S (the clique we will output).
2. If there is a node u in $V - S$ that is connected to every node in S, add u to S. $O(n|S|)$ checks for edge existence.
3. Repeat the previous step until no such node u is found. $O(n|S|^2)$ checks for edge existence.

Clique Decomposition



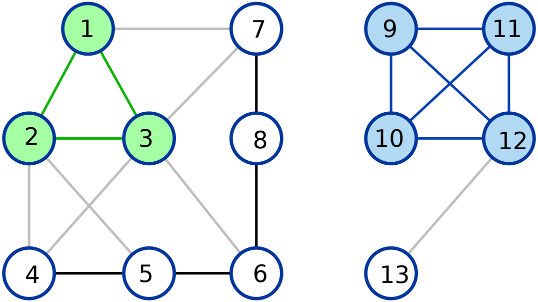
- What do we do after computing a maximal clique?

Clique Decomposition



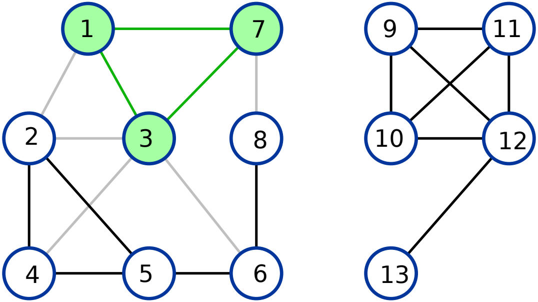
- What do we do after computing a maximal clique?
- Delete nodes in that clique from the graph and repeat.

Clique Decomposition



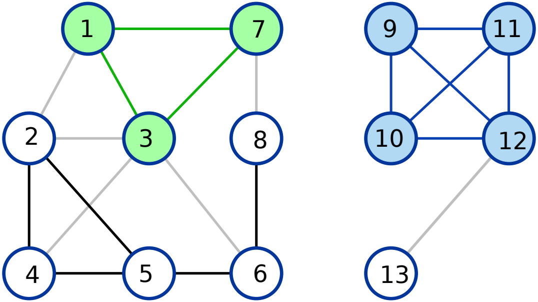
- What do we do after computing a maximal clique?
- Delete nodes in that clique from the graph and repeat.
- The resulting set of cliques forms a *clique decomposition* of G.

Clique Decomposition



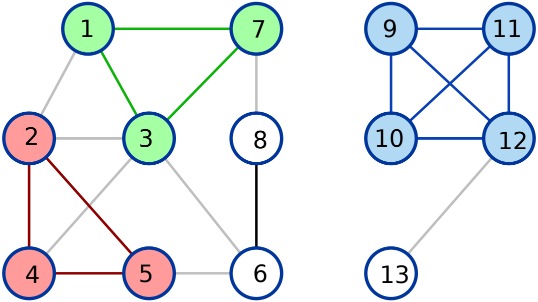
- What do we do after computing a maximal clique?
- Delete nodes in that clique from the graph and repeat.
- The resulting set of cliques forms a *clique decomposition* of G.
- Sequence of cliques found depends on order of processing nodes.

Clique Decomposition



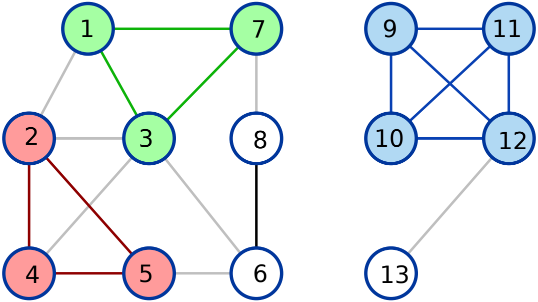
- What do we do after computing a maximal clique?
- Delete nodes in that clique from the graph and repeat.
- The resulting set of cliques forms a *clique decomposition* of G.
- Sequence of cliques found depends on order of processing nodes.

Clique Decomposition



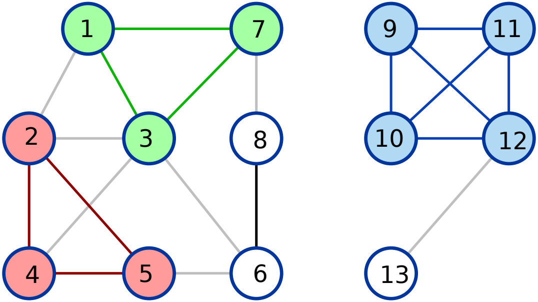
- What do we do after computing a maximal clique?
- Delete nodes in that clique from the graph and repeat.
- The resulting set of cliques forms a *clique decomposition* of G.
- Sequence of cliques found depends on order of processing nodes.

Clique Decomposition



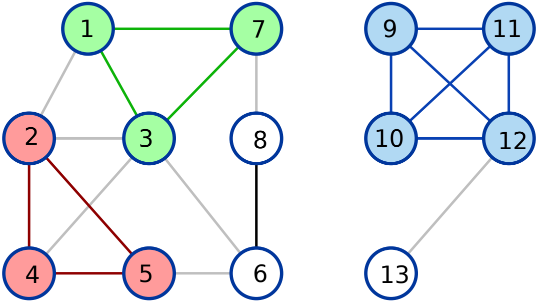
- What do we do after computing a maximal clique?
- Delete nodes in that clique from the graph and repeat.
- The resulting set of cliques forms a *clique decomposition* of G.
- Sequence of cliques found depends on order of processing nodes.
- There is no notion of correctness here since we defined what we compute (the clique decomposition) based on an algorithm we specified.

Clique Decomposition



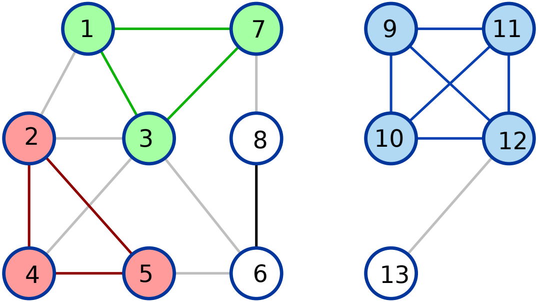
- What do we do after computing a maximal clique?
- Delete nodes in that clique from the graph and repeat.
- The resulting set of cliques forms a *clique decomposition* of G.
- Sequence of cliques found depends on order of processing nodes.
- There is no notion of correctness here since we defined what we compute (the clique decomposition) based on an algorithm we specified.
- Will every edge in G be in some clique in the decomposition? Can a node be in multiple cliques?

Clique Decomposition



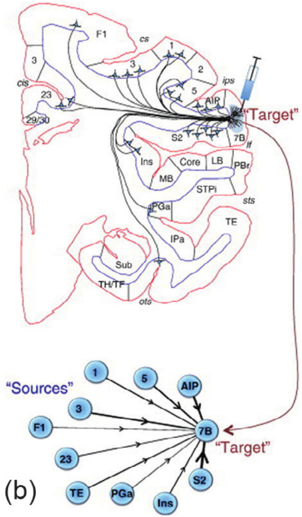
- What do we do after computing a maximal clique?
- Delete nodes in that clique from the graph and repeat.
- The resulting set of cliques forms a *clique decomposition* of G.
- Sequence of cliques found depends on order of processing nodes.
- There is no notion of correctness here since we defined what we compute (the clique decomposition) based on an algorithm we specified.
- Will every edge in G be in some clique in the decomposition? Can a node be in multiple cliques? No, to both questions.

Clique Decomposition



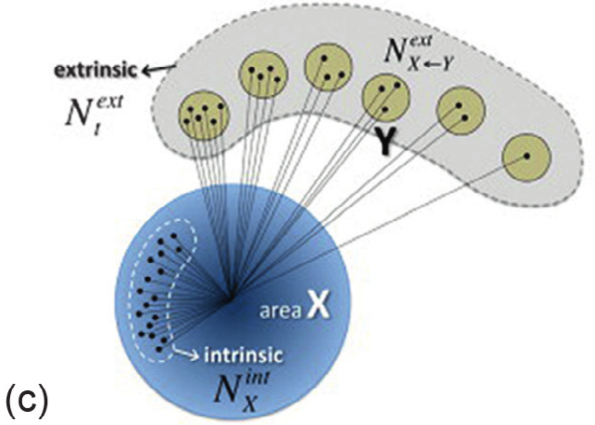
- What do we do after computing a maximal clique?
- Delete nodes in that clique from the graph and repeat.
- The resulting set of cliques forms a *clique decomposition* of G.
- Sequence of cliques found depends on order of processing nodes.
- There is no notion of correctness here since we defined what we compute (the clique decomposition) based on an algorithm we specified.
- Will every edge in G be in some clique in the decomposition? Can a node be in multiple cliques? No, to both questions.
- Modification: After finding a clique, delete only the edges in it.

Structural Connectivity at the Mesoscale



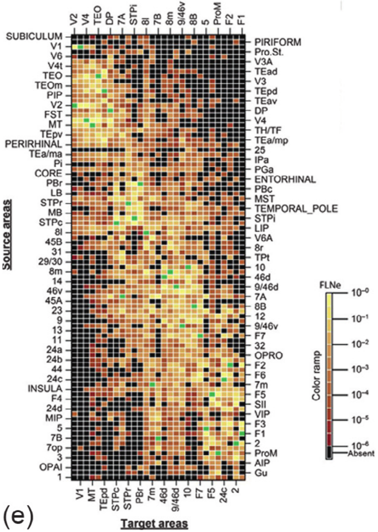
Use retrograde tract tracing. Determine edges coming into node representing area of injection from "labelled" nodes representing neurons that the tracer reaches.

Structural Connectivity at the Mesoscale



Injection is at X: $w(Y, X) = \frac{\text{number of neurons labelled in } Y}{\text{total number of labelled neurons}}$

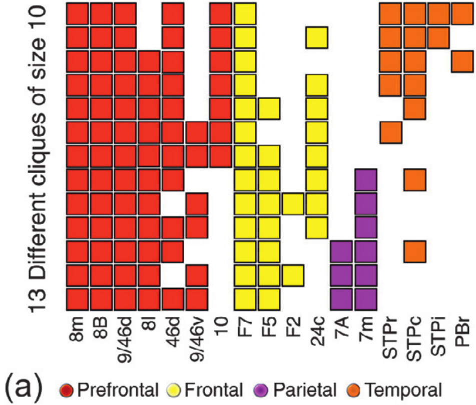
Structural Connectivity at the Mesoscale



Example of connectivity matrix.

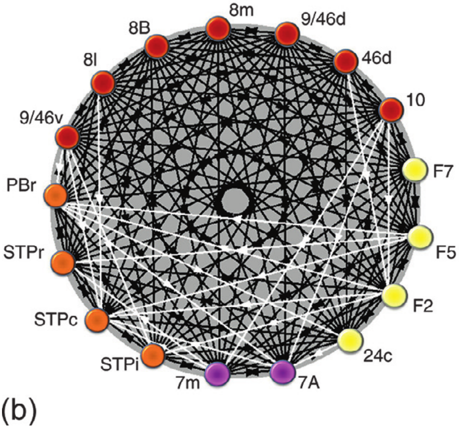
Edge weights range over six orders of magnitude.

Cliques in Macaque Cerebral Cortex Connectome



- 29-node directed graph representing connectome of the cerebral cortex of the macaque; only considering nodes with tracer injection points.
- Computed all 13 maximum cliques, each of which had 10 nodes.

Cliques in Macaque Cerebral Cortex Connectome

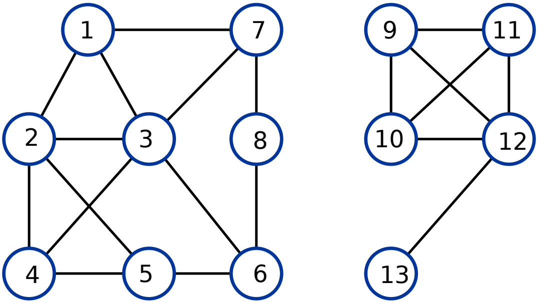


- Union of cliques formed a dense subgraph among 17 nodes.
  - Network among these 17 nodes had density of 0.92.
  - Network between 17 nodes and remaining 12 nodes had density of 0.54.
  - Network among 12 nodes had density of 0.49.
  - *Evidence of core-periphery structure.*

Motivation for k-Cores

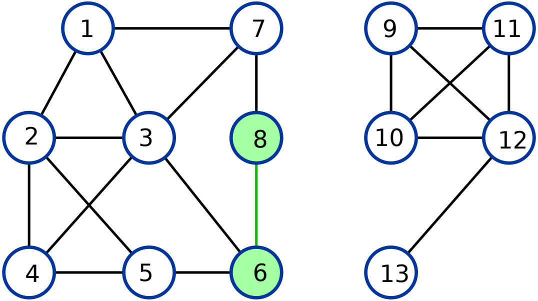
- Definition of a clique is very restrictive.
- Most real networks have small maximal cliques.
- Not very informative of connectome structure.

k-Cores



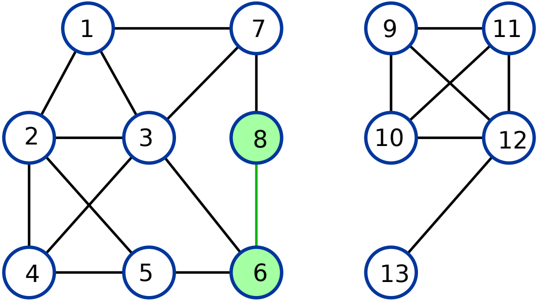
- In an undirected graph $G = (V, E)$, a subset of nodes $C \subseteq V$ is a *k-core* if every node $u \in C$ is connected in G to at least k nodes in C.

k-Cores



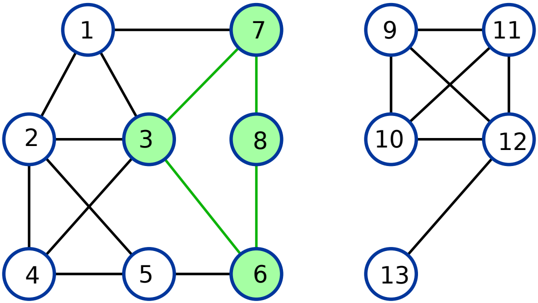
- In an undirected graph $G = (V, E)$, a subset of nodes $C \subseteq V$ is a *k-core* if every node $u \in C$ is connected in G to at least k nodes in C.
- What is largest the 1-core of G?

k-Cores



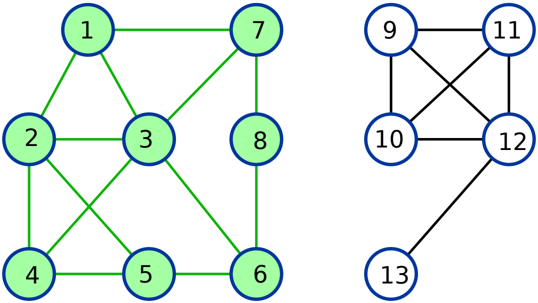
- In an undirected graph $G = (V, E)$, a subset of nodes $C \subseteq V$ is a *k-core* if every node $u \in C$ is connected in G to at least k nodes in C.
- What is largest the 1-core of G? G itself (without any nodes of degree zero).

k-Cores



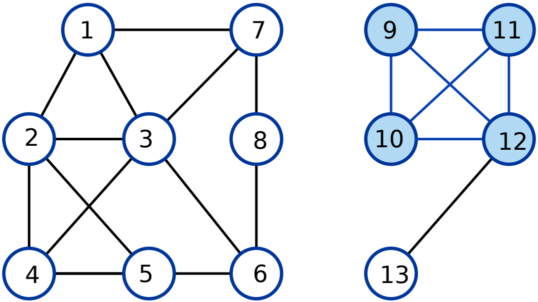
- In an undirected graph $G = (V, E)$, a subset of nodes $C \subseteq V$ is a *k-core* if every node $u \in C$ is connected in G to at least k nodes in C.
- What is largest the 1-core of G? G itself (without any nodes of degree zero).

k-Cores



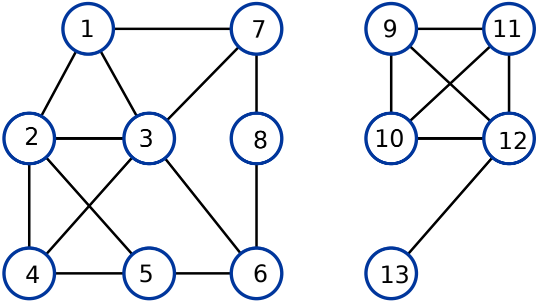
- In an undirected graph $G = (V, E)$, a subset of nodes $C \subseteq V$ is a *k-core* if every node $u \in C$ is connected in G to at least k nodes in C.
- What is largest the 1-core of G? G itself (without any nodes of degree zero).

k-Cores



- In an undirected graph $G = (V, E)$, a subset of nodes $C \subseteq V$ is a *k-core* if every node $u \in C$ is connected in G to at least k nodes in C.
- What is largest the 1-core of G? G itself (without any nodes of degree zero).
- Does this graph have a 4-core?

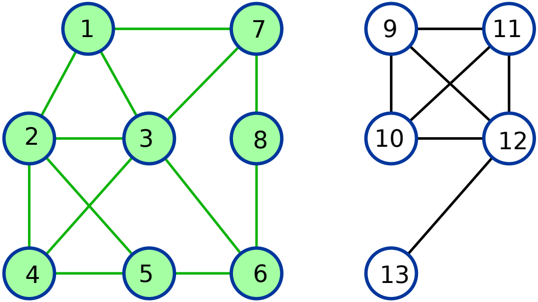
Problems related to k-cores



k-CORE EXISTENCE

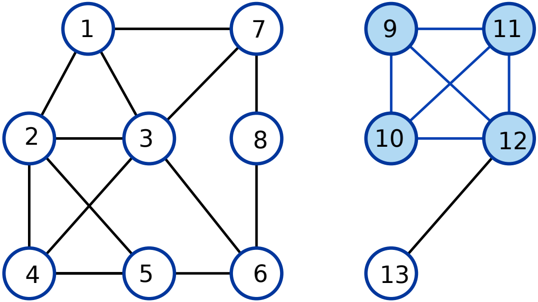
Given an undirected, unweighted graph $G(V, E)$ and an integer k, compute the k-core with the largest number of nodes in G, if it exists.

Problems related to k-cores



k-core Existence
Given an undirected, unweighted graph $G(V, E)$ and an integer k, compute the k-core with the largest number of nodes in G, if it exists.

Problems related to k-cores



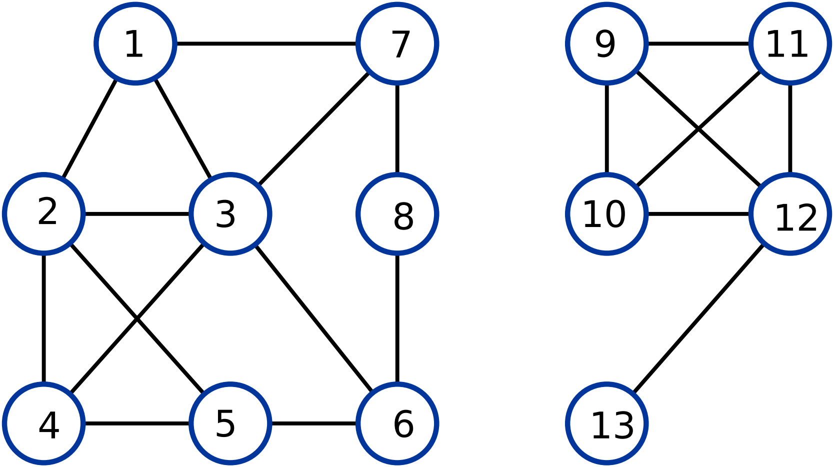
k-core Existence

Given an undirected, unweighted graph $G(V, E)$ and an integer k, compute the k-core with the largest number of nodes in G, if it exists.

Largest k-core

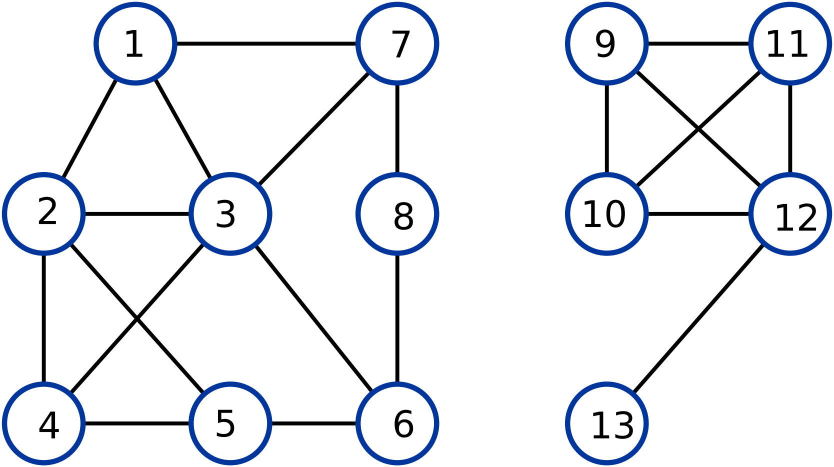
Given an undirected, unweighted graph $G(V, E)$, compute the largest value of k for which G contains a k-core.

Algorithm for k-Core Existence



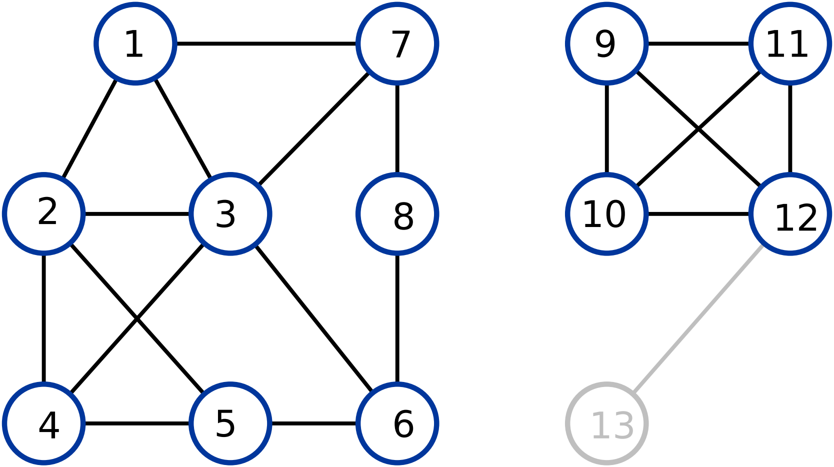
- Repeatedly delete all nodes of degree $< k$ until

Algorithm for k-Core Existence



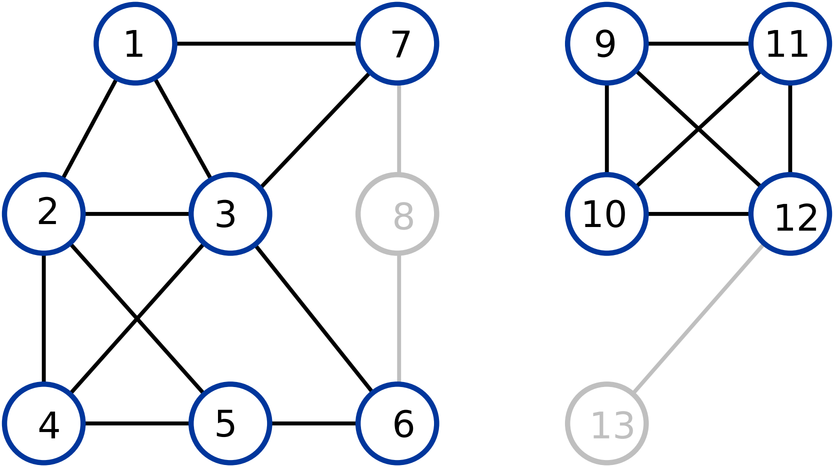
- Repeatedly delete all nodes of degree $< k$ until every remaining node has degree $\geq k$.
- Resulting graph is the largest k-core.

Algorithm for k-Core Existence



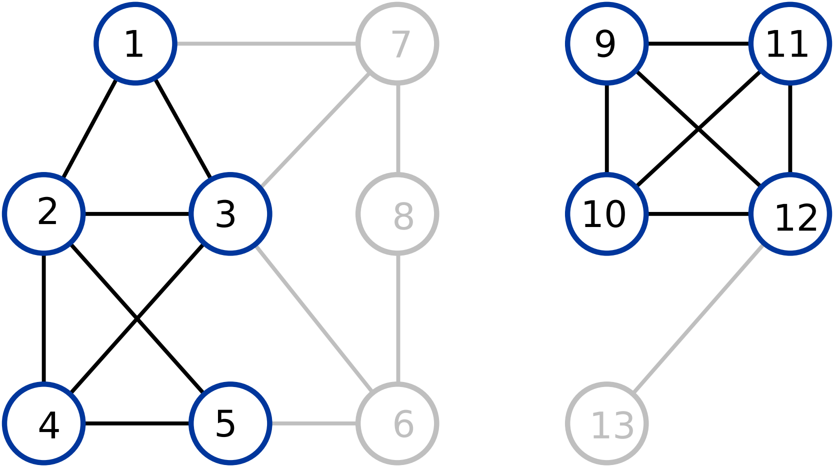
- Repeatedly delete all nodes of degree $< k$ until every remaining node has degree $\geq k$.
- Resulting graph is the largest k-core.

Algorithm for k-Core Existence



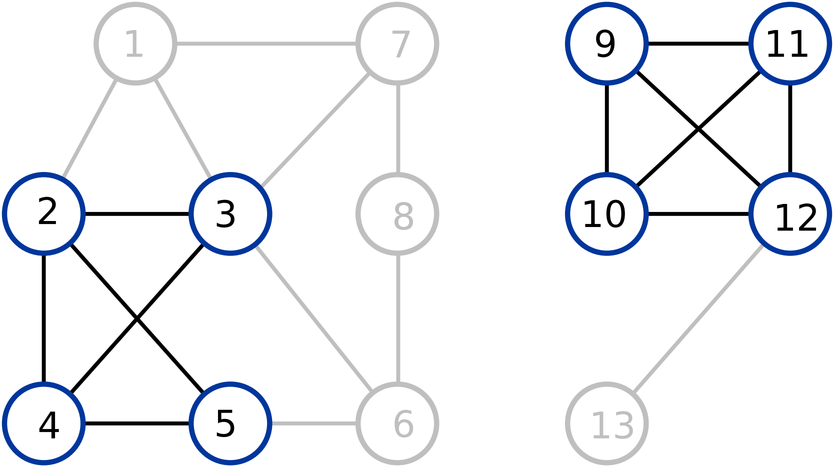
- Repeatedly delete all nodes of degree $< k$ until every remaining node has degree $\geq k$.
- Resulting graph is the largest k-core.

Algorithm for k-Core Existence



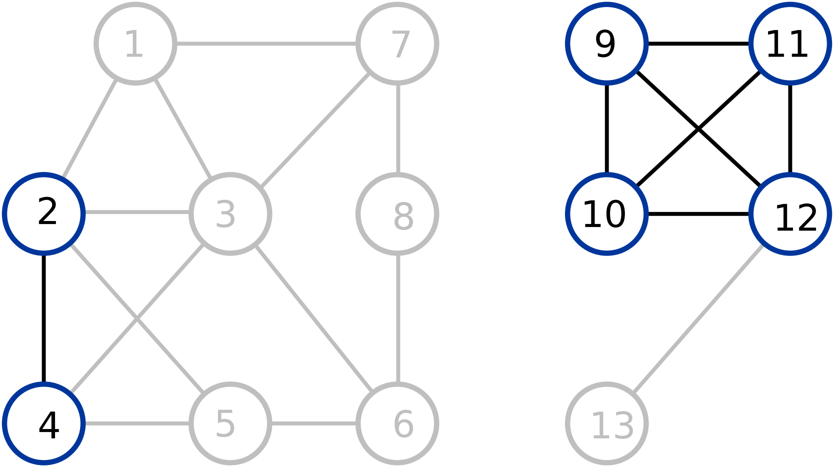
- Repeatedly delete all nodes of degree $< k$ until every remaining node has degree $\geq k$.
- Resulting graph is the largest k-core.

Algorithm for k-Core Existence



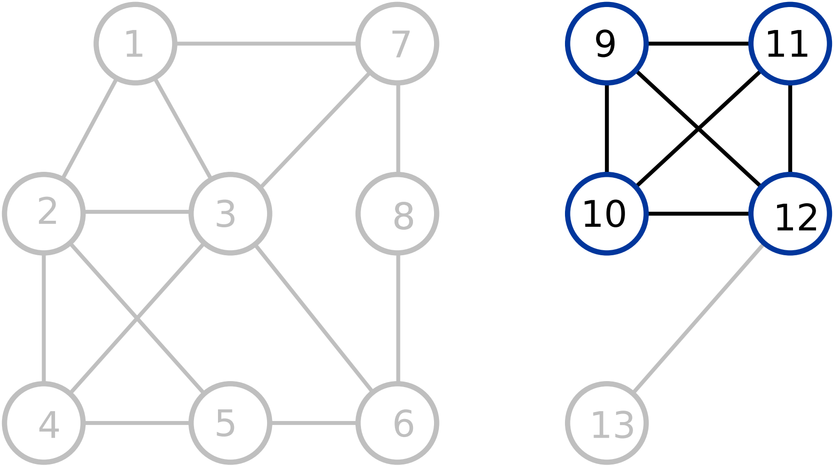
- Repeatedly delete all nodes of degree $< k$ until every remaining node has degree $\geq k$.
- Resulting graph is the largest k-core.

Algorithm for k-Core Existence



- Repeatedly delete all nodes of degree $< k$ until every remaining node has degree $\geq k$.
- Resulting graph is the largest k-core.

Algorithm for k-Core Existence



- Repeatedly delete all nodes of degree $< k$ until every remaining node has degree $\geq k$.
- Resulting graph is the largest k-core.

Correctness of k-Core Existence Algorithm

- Repeatedly delete all nodes of degree $< k$ until every remaining node has degree $\geq k$.
- Why should the resulting graph H be a k-core?
- Why should the resulting graph H be the k-core with the largest number of nodes?

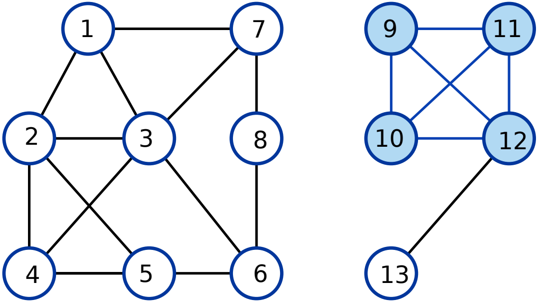
Correctness of k-Core Existence Algorithm

- Repeatedly delete all nodes of degree $< k$ until every remaining node has degree $\geq k$.
- Why should the resulting graph H be a k-core?
- Why should the resulting graph H be the k-core with the largest number of nodes?
- Proof by contradiction.
  - Suppose there is a k-core H' with more nodes than H.
  - Then $H \cup H'$ is also a k-core.
  - Moreover, no node in H' will be deleted by the algorithm.

Correctness of k-Core Existence Algorithm

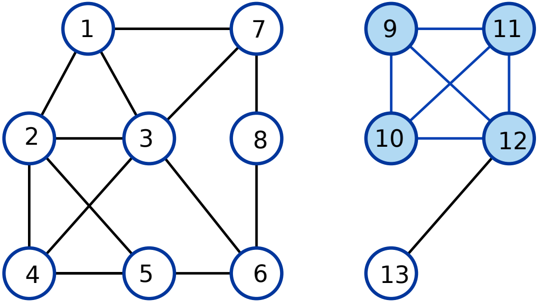
- Repeatedly delete all nodes of degree $< k$ until every remaining node has degree $\geq k$.
- Why should the resulting graph H be a k-core?
- Why should the resulting graph H be the k-core with the largest number of nodes?
- Proof by contradiction.
  - Suppose there is a k-core H' with more nodes than H.
  - Then $H \cup H'$ is also a k-core.
  - Moreover, no node in H' will be deleted by the algorithm.
- How do we implement k-core algorithm efficiently?

Cores vs. Cliques



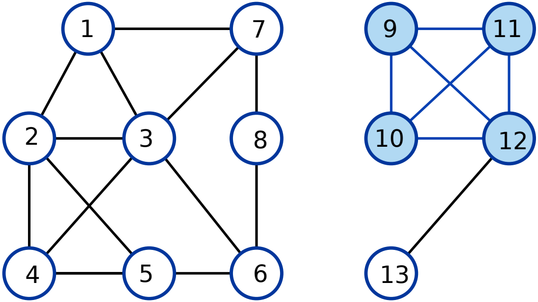
- A clique with k nodes is a $(k-1)$-core.
- Can we use the k-core algorithm to find maximum cliques?

Cores vs. Cliques



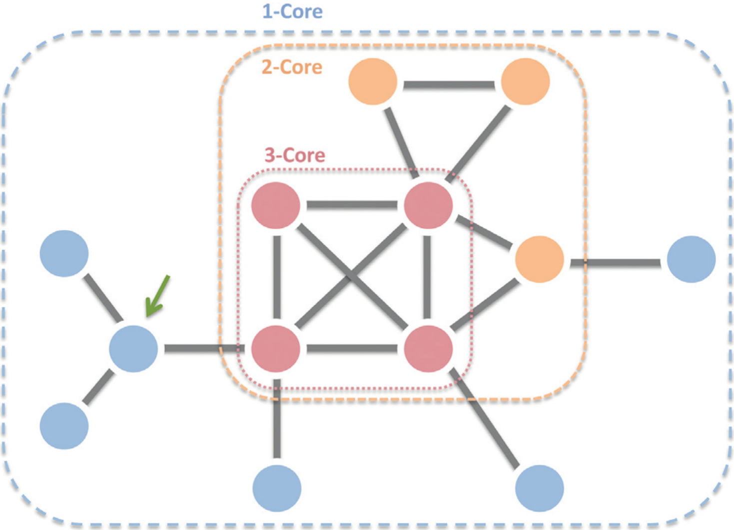
- A clique with k nodes is a $(k-1)$-core.
- Can we use the k-core algorithm to find maximum cliques?
- Idea: Compute the largest value of k for which a k-core H exists. If H is a clique, it must be the largest clique (of size $k+1$) in the graph.

Cores vs. Cliques



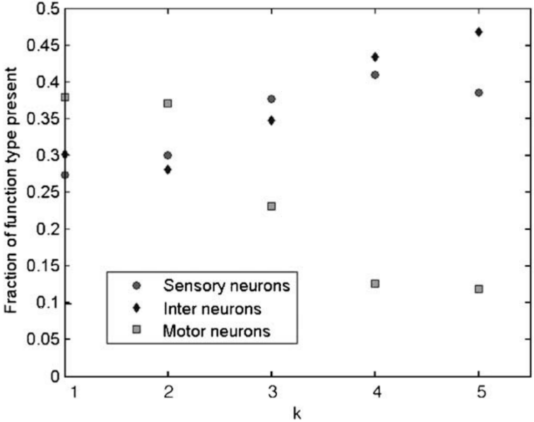
- A clique with k nodes is a $(k-1)$-core.
- Can we use the k-core algorithm to find maximum cliques?
- Idea: Compute the largest value of k for which a k-core H exists. If H is a clique, it must be the largest clique (of size $k+1$) in the graph.
- Flaw is that H may not be a clique, in general. The largest clique may be disjoint from H or be a subgraph of H.
- Moreover, the maximum clique may have l nodes while there may be a k-core where $k > l - 1$, e.g., $k = 3$ and $l = 3$. Create such an example.

k-Core Decomposition



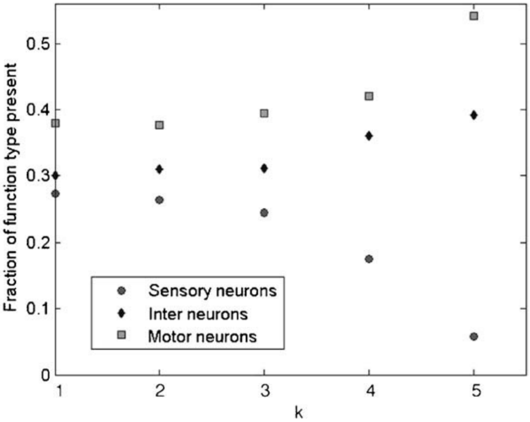
- Label each node by the k-core to which it belongs.

k-Core Decomposition of C. Elegans Connectome



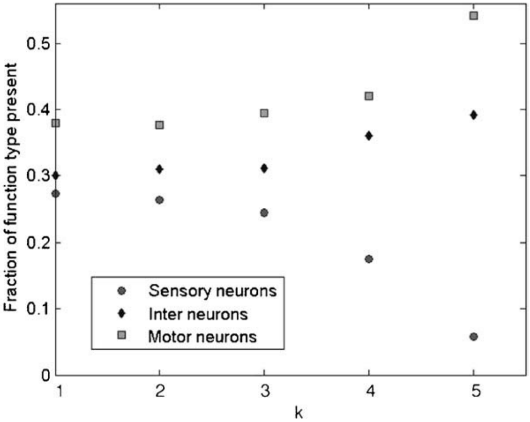
- Based on out-degree: Sensory neurons comprise the innermost cores.

k-Core Decomposition of C. Elegans Connectome



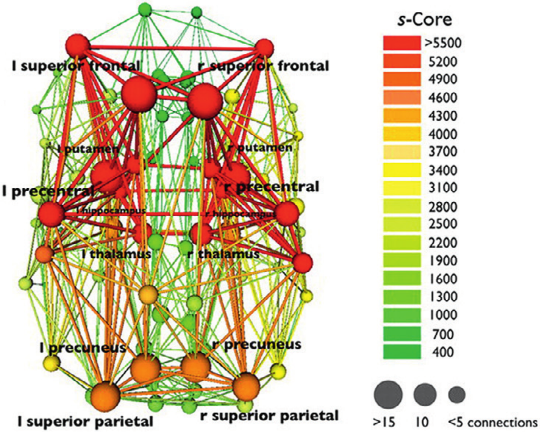
- Based on out-degree: Sensory neurons comprise the innermost cores.
- Based on in-degree: Motor neurons comprise the innermost cores.

k-Core Decomposition of C. Elegans Connectome



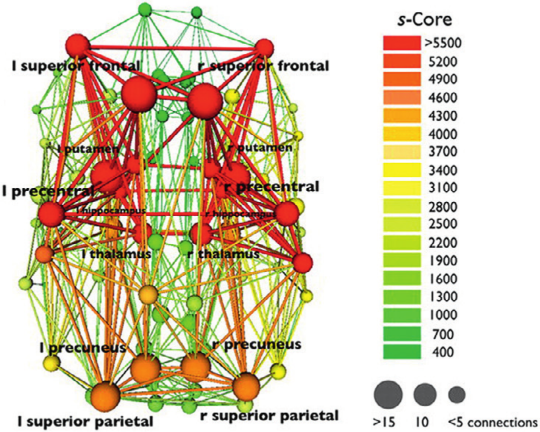
- Based on out-degree: Sensory neurons comprise the innermost cores.
- Based on in-degree: Motor neurons comprise the innermost cores.
- Neurons in lateral ganglion present innermost in-cores and out-cores $\Rightarrow$ they represent hubs that link sensory and motor function.

s-Core Decomposition of Human Connectome



- Structural connectivity from diffusion tensor imaging.
- Connectome is the average of 21 individuals.
- Extend k-core algorithm to weighted networks.

s-Core Decomposition of Human Connectome



- Structural connectivity from diffusion tensor imaging.
- Connectome is the average of 21 individuals.
- Extend k-core algorithm to weighted networks.

- Thalamus relays sensory information and acts as a center for pain perception.
- Precuneus is involved in self-referential processing, imagery and memory, and its deactivation is associated with anaesthetic-induced loss of consciousness.
- Putamen is interconnected with many other structures and influences many types of motor behaviors.