V5 Graph connectivity

V5 closely follows chapter 5.1 in
on „Vertex- and Edge-Connectivity“

V6 will cover part of chapter 5.3 on
„Max-Min Duality and Menger‘s Theorems“
and maybe chapter 5.4 on
„Block Decompositions“

Graph connectivity is related to analyzing biological networks for
- finding cliques
- edge betweenness
- modular decomposition
that will be covered in forthcoming lectures.

Second half of V5: finding cliques in sparse networks.
Motivation

Some connected graphs are „more connected“ than others.

E.g. some connected graphs can be disconnected by the removal of a single vertex or a single edge, whereas others remain connected unless more vertices or more edges are removed.

→ use **vertex-connectivity** and **edge-connectivity** to **measure** the **connectedness** of a graph.

Determining the number of edges (or vertices) that must be removed to disconnect a given connected graph applies directly to analyzing the **vulnerability** of existing networks.

**Definition**: A graph is **connected** if for every pair of vertices \( u \) and \( v \), there is a walk from \( u \) to \( v \).

**Definition**: A **component** of \( G \) is a maximal connected subgraph of \( G \).
Vertex- and Edge-Connectivity

**Definition:** A vertex-cut in a graph $G$ is a vertex-set $U$ such that $G – U$ has more components than $G$.

A cut-vertex (or cutpoint) is a vertex-cut consisting of a single vertex.

**Definition:** An edge-cut in a graph $G$ is a set of edges $D$ such that $G – D$ has more components than $G$.

A cut-edge (or bridge) is an edge-cut consisting of a single edge.

The vertex-connectivity of a connected graph $G$, denoted $\kappa_v(G)$, is the minimum number of vertices whose removal can either disconnect $G$ or reduce it to a 1-vertex graph.

$\rightarrow$ if $G$ has at least one pair of non-adjacent vertices, then $\kappa_v(G)$ is the size of a smallest vertex-cut.
Vertex- and Edge-Connectivity

**Definition**: A graph \( G \) is **\( k \)-connected** if \( G \) is connected and \( \kappa_v(G) \geq k \).

If \( G \) has non-adjacent vertices, then \( G \) is \( k \)-connected if every vertex-cut has at least \( k \) vertices.

**Definition**: The edge-connectivity of a connected graph \( G \), denoted \( \kappa_e(G) \), is the minimum number of edges whose removal can disconnect \( G \).

\[ \rightarrow \text{if } G \text{ is a connected graph,} \]

the edge-connectivity \( \kappa_e(G) \) is the size of a smallest edge-cut.

**Definition**: A graph \( G \) is **\( k \)-edge-connected** if \( G \) is connected and every edge-cut has at least \( k \) edges (i.e. \( \kappa_e(G) \geq k \)).
**Vertex- and Edge-Connectivity**

**Example:** In the graph below, the vertex set \( \{x, y\} \) is one of three different 2-element vertex-cuts. There is no cut-vertex. \( \rightarrow \kappa_v(G) = 2 \).

The edge set \( \{a, b, c\} \) is the unique 3-element edge-cut of graph \( G \), and there is no edge-cut with fewer than 3 edges. Therefore \( \kappa_e(G) = 3 \).

![Graph with vertex and edge connectivity](image)

**Figure 5.1.1** A graph \( G \) with \( \kappa_v(G) = 2 \) and \( \kappa_e(G) = 3 \).

**Application:** The connectivity measures \( \kappa_v \) and \( \kappa_e \) are used in a quantified model of **network survivability**, which is the capacity of a network to retain connections among its nodes after some edges or nodes are removed.
Vertex- and Edge-Connectivity

Since neither the vertex-connectivity nor the edge-connectivity of a graph is affected by the existence or absence of self-loops, we will assume in the following that all graphs are loopless.

Proposition 5.1.1 Let $G$ be a graph. Then the edge-connectivity $\kappa_e(G)$ is less than or equal to the minimum degree $\delta_{\text{min}}(G)$.

Proof: Let $v$ be a vertex of graph $G$ with degree $k = \delta_{\text{min}}(G)$. Then, the deletion of the $k$ edges that are incident on vertex separates $v$ from the other vertices of $G$. \[\blacksquare\]

Definition: A collection of distinct non-empty subsets $\{S_1, S_2, \ldots, S_l\}$ of a set $A$ is a partition of $A$ if both of the following conditions are satisfied:

1. $S_i \cap S_j = \emptyset$, $\forall 1 \leq i < j \leq l$
2. $\bigcup_{i=1}^{l} S_i = A$
Partition Cuts and Minimal Edge-Cuts

Definition: Let $G$ be a graph, and let $X_1$ and $X_2$ form a partition of $V_G$. The set of all edges of $G$ having one endpoint in $X_1$ and the other endpoint in $X_2$ is called a partition-cut of $G$ and is denoted $\langle X_1, X_2 \rangle$.

Proposition 4.6.3: Let $\langle X_1, X_2 \rangle$ be a partition-cut of a connected graph $G$. If the subgraphs of $G$ induced by the vertex sets $X_1$ and $X_2$ are connected, then $\langle X_1, X_2 \rangle$ is a minimal edge-cut.

Proof: The partition-cut $\langle X_1, X_2 \rangle$ is an edge-cut of $G$, since $X_1$ and $X_2$ lie in different components of $G - \langle X_1, X_2 \rangle$. Is it minimal?

Let $S$ be a proper subset of $\langle X_1, X_2 \rangle$, and let edge $e \in \langle X_1, X_2 \rangle - S$. By definition of $\langle X_1, X_2 \rangle$, one endpoint of $e$ is in $X_1$ and the other endpoint is in $X_2$. Thus, if the subgraphs induced by the vertex sets $X_1$ and $X_2$ are connected, then $G - S$ is connected. Therefore, $S$ is not an edge-cut of $G$, which implies that $\langle X_1, X_2 \rangle$ is a minimal edge-cut. □
Partition Cuts and Minimal Edge-Cuts

Proposition 4.6.4. Let $S$ be a minimal edge-cut of a connected graph $G$, and let $X_1$ and $X_2$ be the vertex-sets of the two components of $G - S$. Then $S = \langle X_1, X_2 \rangle$.

Proof: Clearly, $S \subseteq \langle X_1, X_2 \rangle$, i.e. every edge $e \in S$ has one endpoint in $X_1$ and one in $X_2$. Otherwise, the two endpoints would either both belong to $X_1$ or to $X_2$. Then, $S$ would not be minimal because $S - e$ would also be an edge-cut of $G$. On the other hand, if $e \in \langle X_1, X_2 \rangle - S$, then its endpoints would lie in the same component of $G - S$, contradicting the definition of $X_1$ and $X_2$. □

Remark: This assumes that the removal of a minimal edge-cut from a connected graph creates exactly two components.
**Partition Cuts and Minimal Edge-Cuts**

**Proposition 4.6.5.** A partition-cut $\langle X_1, X_2 \rangle$ in a connected graph $G$ is a minimal edge-cut of $G$ or a union of edge-disjoint minimal edge-cuts.

Proof: Since $\langle X_1, X_2 \rangle$ is an edge-cut of $G$, it must contain a minimal edge-cut, say $S$. If $\langle X_1, X_2 \rangle \neq S$, then let $e \in \langle X_1, X_2 \rangle - S$, where the endpoints $v_1$ and $v_2$ of $e$ lie in $X_1$ and $X_2$, respectively. Since $S$ is a minimal edge-cut, the $X_1$-endpoints of $S$ are in one of the components of $G - S$, and the $X_2$-endpoints are in the other component. Furthermore, $v_1$ and $v_2$ are in the same component of $G - S$ (since $e \in G - S$).

Suppose, wlog, that $v_1$ and $v_2$ are in the same component as the $X_1$-endpoints of $S$. Then every path in $G$ from $v_1$ to $v_2$ must use at least one edge of $\langle X_1, X_2 \rangle - S$. Thus, $\langle X_1, X_2 \rangle - S$ is an edge-cut of $G$ and contains a minimal edge-cut $R$. Applying the same argument, $\langle X_1, X_2 \rangle - (S \cup R)$ either is empty or is an edge-cut of $G$. Eventually, the process ends with $\langle X_1, X_2 \rangle - (S_1 \cup S_2 \cup \ldots \cup S_r) = \emptyset$, where the $S_i$ are edge-disjoint minimal edge-cuts of $G$. □
Partition Cuts and Minimal Edge-Cuts

**Proposition 5.1.2.** A graph $G$ is $k$-edge-connected if and only if every partition-cut contains at least $k$ edges.

Proof: ($\Rightarrow$) Suppose, that graph $G$ is $k$-edge connected. Then every partition-cut of $G$ has at least $k$ edges, since a partition-cut is an edge-cut.

($\Leftarrow$) Suppose that every partition-cut contains at least $k$ edges. By proposition 4.6.4., every minimal edge-cut is a partition-cut. Thus, every edge-cut contains at least $k$ edges. $\square$
Proposition 5.1.3. Let $e$ be any edge of a $k$-connected graph $G$, for $k \geq 3$. Then the edge-deletion subgraph $G - e$ is $(k - 1)$-connected.

Proof: Let $W = \{w_1, w_2, \ldots, w_{k-2}\}$ be any set of $k - 2$ vertices in $G - e$, and let $x$ and $y$ be any two different vertices in $(G - e) - W$. It suffices to show the existence of an $x$-$y$ walk in $(G - e) - W$.

First, suppose that at least one of the endpoints of edge $e$ is contained in set $W$. Since the vertex-deletion subgraph $G - W$ is 2-connected, there is an $x$-$y$ path in $G - W$. This path cannot contain edge $e$. Hence, it is an $x$-$y$ path in the subgraph $(G - e) - W$.

Next suppose that neither endpoint of edge $e$ is in set $W$. Then there are two cases to consider.
Relationship between vertex- and edge-connectivity

Case 1: Vertices $x$ and $y$ are the endpoints of edge $e$.

Graph $G$ has at least $k + 1$ vertices (since $G$ is $k$-connected).

So there exists some vertex $z \in G – \{w_1, w_2, \ldots, w_{k-2}, x, y\}$.

Since graph $G$ is $k$-connected, there exists
an $x$-$z$ path $P_1$ in the vertex deletion subgraph $G – \{w_1, w_2, \ldots, w_{k-2}, y\}$ and
a $z$-$y$ path $P_2$ in the subgraph $G – \{w_1, w_2, \ldots, w_{k-2}, x\}$

![Diagram of graph with vertices and paths](image)

Figure 5.1.2 The existence of an $x$-$y$ walk in $(G - e) – \{w_1, w_2, \ldots, w_{k-2}\}$.

Neither of these paths contains edge $e$, and, therefore,
their concatenation is an $x$-$y$ walk in the subgraph $(G – e) – \{w_1, w_2, \ldots, w_{k-2}\}$
Relationship between vertex- and edge-connectivity

Case 2: At least one of the vertices $x$ and $y$, say $x$, is not an endpoint of edge $e$. Let $u$ be an endpoint of edge $e$ that is different from vertex $x$. Since graph $G$ is $k$-connected, the subgraph $G - \{w_1,w_2, \ldots, w_{k-2},u\}$ is connected.

Hence, there is an $x$-$y$ path $P$ in $G - \{w_1,w_2, \ldots, w_{k-2},u\}$.

It follows that $P$ is an $x$-$y$ path in $G - \{w_1,w_2, \ldots, w_{k-2}\}$ that does not contain vertex $u$ and, hence excludes edge $e$ (even if $P$ contains the other endpoint of $e$, which it could).

Therefore, $P$ is an $x$-$y$ path in $(G - e) - \{w_1,w_2, \ldots, w_{k-2}\}$. □
Relationship between vertex- and edge-connectivity

Corollary 5.1.4. Let $G$ be a $k$-connected graph, and let $D$ be any set of $m$ edges of $G$, for $m \leq k - 1$. Then the edge-deletion subgraph $G - D$ is $(k - m)$-connected.

Proof: this follows from the iterative application of proposition 5.1.3. □

Corollary 5.1.5. Let $G$ be a connected graph. Then $\kappa_e(G) \geq \kappa_v(G)$.

Proof. Let $k = \kappa_v(G)$, and let $S$ be any set of $k - 1$ edges in graph $G$. Since $G$ is $k$-connected, the graph $G - S$ is 1-connected, by corollary 5.1.4. Thus, the edge subset $S$ is not an edge-cut of graph $G$, which implies that $\kappa_e(G) \geq k$. □

Corollary 5.1.6. Let $G$ be a connected graph. Then $\kappa_v(G) \leq \kappa_e(G) \leq \delta_{\min}(G)$.

This is a combination of Proposition 5.1.1 and Corollary 5.1.5. □
Internally Disjoint Paths and Vertex-Connectivity: Whitney’s Theorem

A communications network is said to be *fault-tolerant* if it has at least two alternative paths between each pair of vertices. This notion characterizes 2-connected graphs. A more general result for *k*-connected graphs follows later.

**Terminology:** A vertex of a path $P$ is an **internal vertex** of $P$ if it is neither the initial nor the final vertex of that path.

**Definition:** Let $u$ and $v$ be two vertices in a graph $G$. A collection of $u$-$v$ paths in $G$ is said to be **internally disjoint** if no two paths in the collection have an internal vertex in common.
Theorem 5.1.7 [Whitney, 1932] Let \( G \) be a connected graph with \( n \geq 3 \) vertices. Then \( G \) is 2-connected if and only if for each pair of vertices in \( G \), there are two internally disjoint paths between them.

Proof: (\( \Leftarrow \)) Suppose that graph \( G \) is not 2-connected. Then let \( v \) be a cut-vertex of \( G \). Since \( G - v \) is not connected, there must be two vertices such that there is no \( x-y \) path in \( G - v \). It follows that \( v \) is an internal vertex of every \( x-y \) path in \( G \).

(\( \Rightarrow \)) Suppose that graph \( G \) is 2-connected, and let \( x \) and \( y \) be any two vertices in \( G \). We use induction on the distance \( d(x,y) \) to prove that there are at least two vertex-disjoint \( x-y \) paths in \( G \).

If there is an edge \( e \) joining vertices \( x \) and \( y \), (i.e., \( d(x,y) = 1 \)), then the edge-deletion subgraph \( G - e \) is connected, by Corollary 5.1.4. Thus, there is an \( x-y \) path \( P \) in \( G - e \).

It follows that path \( P \) and edge \( e \) are two internally disjoint \( x-y \) paths in \( G \).
Internally Disjoint Paths and Vertex-Connectivity: Whitney’s Theorem

Next, assume for some $k \geq 2$ that the assertion holds for every pair of vertices whose distance apart is less than $k$. Let $x$ and $y$ be vertices such that distance $d(x,y) = k$, and consider an $x$-$y$ path of length $k$.

Let $w$ be the vertex that immediately precedes vertex $y$ on this path, and let $e$ be the edge between vertices $w$ and $y$.

Since $d(x,w) < k$, the induction hypothesis implies that there are two internally disjoint $x$-$w$ paths in $G$, say $P$ and $Q$.

Also, since $G$ is 2-connected, there exists an $x$-$y$ path $R$ in $G$ that avoids vertex $w$.

Path $Q$ either contains vertex $y$ (right) or it does not (left)
Internally Disjoint Paths and Vertex-Connectivity: Whitney’s Theorem

Let \( z \) be the last vertex on path \( R \) that precedes vertex \( y \) and is also on one of the paths \( P \) or \( Q \) (\( z \) might be vertex \( x \)). Assume wlog that \( z \) is on path \( P \). Then \( G \) has two internally disjoint \( x-y \) paths. One of these paths is the concatenation of the subgraph of \( P \) from \( x \) to \( z \) with the subpath of \( R \) from \( z \) to \( y \).

If vertex \( y \) is not on path \( Q \), then a second \( x-y \) path, internally disjoint from the first one, is the concatenation of path \( Q \) with the edge \( e \) joining vertex \( w \) to vertex \( y \).

If \( y \) is on path \( Q \), then the subpath of \( Q \) from \( x \) to \( y \) can be used as the second path.

\( \square \)

**Corollary 5.1.8.** Let \( G \) be a graph with at least three vertices. Then \( G \) is 2-connected if and only if any two vertices of \( G \) lie on a common cycle.

Proof: this follows from 5.1.7., since two vertices \( x \) and \( y \) lie on a common cycle if and only if there are two internally disjoint \( x-y \) paths. \( \square \)
Characterization of 2-connected graphs

Theorem 5.1.9. Let $G$ be a connected graph with at least 3 vertices. Then the following statements are equivalent:

1. The graph $G$ is 2-connected.
2. For any 2 vertices of $G$, there is a cycle containing both.
3. For any vertex and any edge of $G$, there is a cycle containing both.
4. For any two edges of $G$, there is a cycle containing both.
5. For any two vertices and one edge of $G$, there is a path containing all three.
6. For any three distinct vertices of $G$, there is a path containing all three.
7. For any three distinct vertices of $G$, there is a path containing any two of them which does not contain the third.

End of Chapter 5.1! Pooh.
Mesoscale properties of networks
- identify cliques and highly connected clusters

Most relevant processes in biological networks correspond to the mesoscale (5-25 genes or proteins) not to the entire network.

However, it is computationally enormously expensive to study mesoscale properties of biological networks.

e.g. a network of 1000 nodes contains $1 \times 10^{23}$ possible 10-node sets.

Spirin & Mirny analyzed combined network of protein interactions with data from CELLZOME, MIPS, BIND: 6500 interactions.
Identify connected subgraphs

The network of protein interactions is typically presented as an undirected graph with proteins as nodes and protein interactions as undirected edges.

**Aim:** identify highly connected subgraphs (clusters) that have more interactions within themselves and fewer with the rest of the graph.

A fully connected subgraph, or clique, that is not a part of any other clique is an example of such a cluster. The „maximum clique problem“ – finding the largest clique in a given graph is known be NP-hard.

In general, clusters need not to be fully connected.

Measure density of connections by

\[ Q = \frac{2m}{n(n-1)} \]

where \( n \) is the number of proteins in the cluster and \( m \) is the number of interactions between them.

Spirin, Mirny, PNAS 100, 12123 (2003)
(method I) Identify all fully connected subgraphs (cliques)

The general problem - finding all cliques of a graph - is very hard. Because the protein interaction graph is so far very sparse (the number of interactions (edges) is similar to the number of proteins (nodes), this can be done quickly.

To find cliques of size $n$ one needs to enumerate only the cliques of size $n-1$.

The search for cliques starts with $n = 4$, pick all (known) pairs of edges (6500 × 6500 protein interactions) successively. For every pair $A-B$ and $C-D$ check whether there are edges between $A$ and $C$, $A$ and $D$, $B$ and $C$, and $B$ and $D$. If these edges are present, $ABCD$ is a clique.

For every clique identified, $ABCD$, pick all known proteins successively. For every picked protein $E$, if all of the interactions $E-A$, $E-B$, $E-C$, and $E-D$ are known, then $ABCDE$ is a clique with size 5.

Continue for $n = 6, 7, ...$ The largest clique found in the protein-interaction network has size 14. Spirin, Mirny, PNAS 100, 12123 (2003)
(I) Identify all fully connected subgraphs (cliques)

These results include, however, many redundant cliques. For example, the clique with size 14 contains 14 cliques with size 13.

To find all nonredundant subgraphs, mark all proteins comprising the clique of size 14, and out of all subgraphs of size 13 pick those that have at least one protein other than marked.

After all redundant cliques of size 13 are removed, proceed to remove redundant twelves etc.

In total, only 41 nonredundant cliques with sizes 4 - 14 were found.
(method II) Superparamagnetic Clustering (SPC)

SPC uses an analogy to the physical properties of an inhomogenous ferromagnetic model to find tightly connected clusters on a large graph.

Every node on the graph is assigned a Potts spin variable $S_i = 1, 2, ..., q$. The value of this spin variable $S_i$ performs thermal fluctuations, which are determined by the temperature $T$ and the spin values on the neighboring nodes.

Energetically, 2 nodes connected by an edge are favored to have the same spin value. Therefore, the spin at each node tends to align itself with the majority of its neighbors.

When such a Potts spin system reaches equilibrium for a given temperature $T$, high correlation between fluctuating $S_i$ and $S_j$ at nodes $i$ and $j$ would indicate that nodes $i$ and $j$ belong to the same cluster.

Spirin, Mirny, PNAS 100, 12123 (2003)
(II) Superparamagnetic Clustering (SPC)

The protein-interaction network is represented by a graph where every pair of interacting proteins is an edge of length 1.

The simulations are run for temperatures ranging from 0 to 1 in units of the coupling strength.

The network splits two monomers at temperatures between 0.7 and 0.8, whereas larger clusters only exist for temperatures between 0.1 and 0.7.

Clusters are recorded at all values temperature. The overlapping clusters are then merged and redundant ones are removed.

Spirin, Mirny, PNAS 100, 12123 (2003)
(method III) Monte Carlo Simulation

Use MC to find a tight subgraph of a predetermined number of nodes $M$.

At time $t = 0$, a random set of $M$ nodes is selected.
For each pair of nodes $i,j$ from this set, the shortest path $L_{ij}$ between $i$ and $j$ on the graph is calculated.
Denote the sum of all shortest paths $L_{ij}$ from this set as $L_0$.
At every time step one of $M$ nodes is picked at random, and one node is picked at random out of all its neighbors.

The new sum of all shortest paths, $L_1$, is calculated if the original node were to be replaced by this neighbor.
If $L_1 < L_0$, accept replacement with probability 1.
If $L_1 > L_0$, accept replacement with probability $\exp \left(-\frac{L_1 - L_0}{T}\right)$ where $T$ is the effective temperature.

Spirin, Mirny, PNAS 100, 12123 (2003)
(III) Monte Carlo Simulation

Every tenth time step an attempt is made to replace one of the nodes from the current set with a node that has no edges to the current set to avoid getting caught in an isolated disconnected subgraph.

This process is repeated
(i) until the original set converges to a complete subgraph, or
(ii) for a predetermined number of steps,
after which the tightest subgraph (the subgraph corresponding to the smallest $L_0$) is recorded.

The recorded clusters are merged and redundant clusters are removed.

Spirin, Mirny, PNAS 100, 12123 (2003)
Optimal temperature in MC simulation

For every cluster size there is an optimal temperature that gives the fastest convergence to the tightest subgraph.

Time to find a clique with size 7 in MC steps per site as a function of temperature $T$. The region with optimal temperature is shown in *Inset*.

The required time increases sharply as the temperature goes to 0, but has a relatively wide plateau in the region $3 < T < 7$.

Simulations suggest that the choice of temperature $T \approx M$ would be safe for any cluster size $M$.

Spirin, Mirny, PNAS 100, 12123 (2003)
Comparison of SPC and Monte Carlo methods

Comparison of clusters found with SPC (blue) and MC simulation (red). Reasonable overlap (ca. one third of all clusters are found by both methods) – but both methods seem complementary.

Spirin, Mirny, PNAS 100, 12123 (2003)
Comparison of SPC and Monte Carlo methods

The SPC method is best at detecting high-Q value clusters with relatively few links with the outside world. An example is the TRAPP complex, a fully connected clique of size 10 with just 7 links with outside proteins. This cluster was perfectly detected by SPC, whereas the MC simulation was able to find smaller pieces of this cluster separately rather than the whole cluster.

By contrast, MC simulations are better suited for finding very „outgoing“ cliques. The Lsm complex, a clique of size 11, includes 3 proteins with more interactions outside the complex than inside. This complex was easily found by MC, but was not detected as a stand-alone cluster by SPC.

Spirin, Mirny, PNAS 100, 12123 (2003)
Merging Overlapping Clusters

A simple statistical test shows that nodes which have only one link to a cluster are statistically insignificant. Clean such statistically insignificant members first.

Then merge overlapping clusters:
For every cluster $A_i$ find all clusters $A_k$ that overlap with this cluster by at least one protein.
For every such found cluster calculate Q value of a possible merged cluster $A_i \cup A_k$. Record cluster $A_{best}(i)$ which gives the highest Q value if merged with $A_i$.

After the best match is found for every cluster, every cluster $A_i$ is replaced by a merged cluster $A_i \cup A_{best}(i)$ unless $A_i \cup A_{best}(i)$ is below a certain threshold value for $Q_C$.
This process continues until there are no more overlapping clusters or until merging any of the remaining clusters will make a cluster with Q value lower than $Q_C$.

Spirin, Mirny, PNAS 100, 12123 (2003)
Statistical significance of complexes and modules

Number of complete cliques ($Q = 1$) as a function of clique size enumerated in the network of protein interactions (red) and in randomly rewired graphs (blue, averaged >1,000 graphs where number of interactions for each protein is preserved).

*Inset* shows the same plot in log-normal scale. Note the dramatic enrichment in the number of cliques in the protein-interaction graph compared with the random graphs. Most of these cliques are parts of bigger complexes and modules.

Spirin, Mirny, PNAS 100, 12123 (2003)
Statistical significance of complexes and modules

Distribution of Q of clusters found by the MC search method.
Red bars: original network of protein interactions.
Blue curves: randomly rewired graphs.

Clusters in the protein network have many more interactions than their counterparts in the random graphs.

Spirin, Mirny, PNAS 100, 12123 (2003)
Architecture of protein network

Fragment of the protein network. Nodes and interactions in discovered clusters are shown in bold. Nodes are colored by functional categories in MIPS: red, transcription regulation; blue, cell-cycle/cell-fate control; green, RNA processing; and yellow, protein transport. Complexes shown are the SAGA/TFIID complex (red), the anaphase-promoting complex (blue), and the TRAPP complex (yellow).

Spirin, Mirny, PNAS 100, 12123 (2003)
Examples of discovered **functional modules**.

(A) A module involved in cell-cycle regulation. This module consists of cyclins (CLB1-4 and CLN2) and cyclin-dependent kinases (CKS1 and CDC28) and a nuclear import protein (NIP29). Although they have many interactions, these proteins are not present in the cell at the same time.

(B) Pheromone signal transduction pathway in the network of protein–protein interactions. This module includes several MAPK (mitogen-activated protein kinase) and MAPKK (mitogen-activated protein kinase kinase) kinases, as well as other proteins involved in signal transduction. These proteins do not form a single complex; rather, they interact in a specific order.

Spirin, Mirny, PNAS 100, 12123 (2003)
Architecture of protein network

Comparison of discovered complexes and modules with complexes derived experimentally (BIND and Cellzome) and complexes catalogued in MIPS. Discovered complexes are sorted by the overlap with the best-matching experimental complex. The overlap is defined as the number of common proteins divided by the number of proteins in the best-matching experimental complex.

The first 31 complexes match exactly, and another 11 have overlap above 65%. Inset shows the overlap as a function of the size of the discovered complex. Note that discovered complexes of all sizes match very well with known experimental complexes. Discovered complexes that do not match with experimental ones constitute our predictions.

Spirin, Mirny, PNAS 100, 12123 (2003)
Robustness of clusters found

Model effect of false positives in experimental data: randomly reconnect, remove or add 10-50% of interactions in network.

Cluster recovery probability as a function of the fraction of altered links. Black curves correspond to the case when a fraction of links are rewired. Red, removed; green, added. Circles represent the probability to recover 75% of the original cluster; triangles represent the probability to recover 50%.

Noise in the form of removal or additions if links has less deteriorating effect than random rewiring. About 75% of clusters can still be found when 10% of links are rewired.

Spirin, Mirny, PNAS 100, 12123 (2003)
Summary

Here: analysis of meso-scale properties demonstrated the presence of highly connected clusters of proteins in a network of protein interactions. Strong support for suggested modular architecture of biological networks.

Distinguish 2 types of clusters: protein complexes and dynamic functional modules. Both complexes and modules have more interactions among their members than with the rest of the network.

Dynamic modules are elusive to experimental purification because they are not assembled as a complex at any single point in time. Computational analysis allows detection of such modules by integrating pairwise molecular interactions that occur at different times and places. However, computational analysis alone, does not allow to distinguish between complexes and modules or between transient and simultaneous interactions.
Summary

Most of the discovered complexes and modules come from traditional studies, rather than from large-scale experiments. This suggests that although large-scale proteomic studies provide a wealth of protein interaction data, the scarcity of the data (and its contamination with false positives) makes such studies less valuable for identification of functional modules.