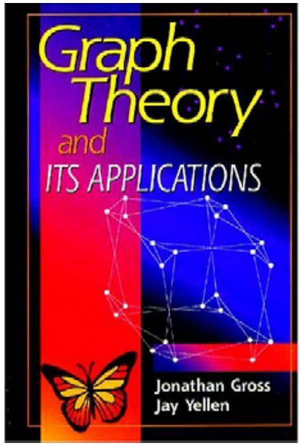
V6 Menger’s theorem

V6 closely follows chapter 5.3 in
on „Max-Min Duality and Menger‘s Theorems“



Second half of V6:
- Girven, Newman, PNAS 99, 7821 (2002)
- Radicchi *et al*. PNAS 101, 2658 (2004)

Borrowing from operations research terminology consider certain *primal-dual pairs* of of optimization problems that are intimately related.

Usually, one of these problems involves the maximization of some objective function, while the other is a minimization problem.

Separating set

A feasible solution to one of the problems provides a bound for the optimal value of the other problem (referred to as *weak duality*), and the optimal value of one problem is equal to the optimal value of the other (*strong duality*).

Definition: Let *u* and *v* be distinct vertices in a connected graph G. A vertex subset (or edge subset) *S* is *u*-*v* **separating** (or **separates** *u* and *v*), if the vertices *u* and *v* lie in different components of the deletion subgraph *G* – *S.*

→ a *u*-*v* separating vertex set is a vertex-cut, and
a *u*-*v* separating edge set is an edge-cut.

When the context is clear, the term *u*-*v* **separating set** will refer either to a *u*-*v* separating vertex set or to a *u*-*v* separating edge set.

Example

For the graph *G* in the Figure below, the vertex-cut {*x,w,z*} is a *u*-*v* separating set of vertices of minimum size, and the edge-cut {*a,b,c,d,e*} is a *u*-*v* separating set of edges of minimum size.

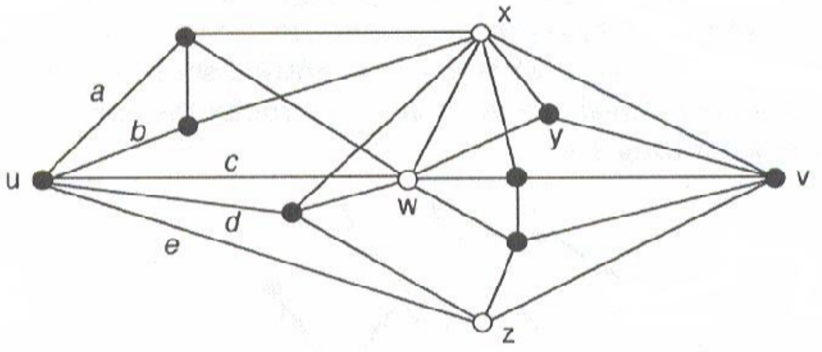


Figure 5.3.1 Vertex- and edge-cuts that are *u*-*v* separating sets.

Notice that a minimum-size *u*-*v* separating set of edges (vertices) need not be a minimum-size edge-cut (vertex-cut).

E.g., the set {*a,b,c,d,e*} is not a minimum-size edge-cut in *G,* because the set of edges incident on the 3-valent vertex *y* is an edge-cut of size 3.

A Primal-Dual Pair of Optimization Problems

The discussion in chapter 5.1 suggests two different interpretations of a graph‘s connectivity. One interpretation is the number of vertices or edges it takes to disconnect the graph, and the other is the number of alternative paths joining any two given vertices of the graph.

Corresponding to these two perspectives are the following two optimization problems for two non-adjacent vertices *u* and *v* of a connected graph *G*.

Maximization Problem: Determine the maximum number of internally disjoint *u*-*v* paths in graph *G*.

Minimization Problem: Determine the minimum number of vertices of graph *G* needed to separate the vertices *u* and *v.*

A Primal-Dual Pair of Optimization Problems

Proposition 5.3.1: (*Weak Duality*) Let u and v be any two non-adjacent vertices of a connected graph G. Let $\mathcal{P}_{uv}$ be a collection of internally disjoint u-v paths in G, and let S_{uv} be a u-v separating set of vertices in G.

Then $| \mathcal{P}_{uv}| \leq | S_{uv} |$.

Proof: Since S_{uv} is a u-v separating set, each u-v path in $\mathcal{P}_{uv}$ must include at least one vertex of S_{uv} . Since the paths in $\mathcal{P}_{uv}$ are internally disjoint, no two paths of them can include the same vertex.

Thus, the number of internally disjoint u-v paths in G is at most $| S_{uv} |$. □

Corollary 5.3.2. Let u and v be any two non-adjacent vertices of a connected graph G. Then the maximum number of internally disjoint u-v paths in G is less than or equal to the minimum size of a u-v separating set of vertices in G.

Menger‘s theorem will show that the two quantities are in fact equal.

A Primal-Dual Pair of Optimization Problems

The following corollary follows directly from Proposition 5.3.1.

Corollary 5.3.3: (*Certificate of Optimality*) Let *u* and *v* be any two non-adjacent vertices of a connected graph *G.*

Suppose that $\mathcal{P}_{\mathrm{uv}}$ is a collection of internally disjoint *u*-*v* paths in *G,* and that S_{uv} is a *u*-*v* separating set of vertices in G, such that $|\ \mathcal{P}_{\mathrm{uv}}| = |\ S_{uv}\ |$.

Then $\mathcal{P}_{\mathrm{uv}}$ is a maximum-size collection of internally disjoint *u*-*v* paths, and S_{uv} is a minimum-size *u*-*v* separating set.

Vertex- and Edge-Connectivity

Example: In the graph G below, the vertex sequences $\langle u,x,y,t,v\rangle$, $\langle u,z,v\rangle$, and $\langle u,r,s,v\rangle$ represent a collection $\mathcal{P}$ of three internally disjoint u-v paths in G, and the set $S = \{y,s,z\}$ is a u-v separating set of size 3.

Therefore, by Corollary 5.3.3, $\mathcal{P}$ is a maximum-size collection of internally disjoint u-v paths, and S is a minimum-size u-v separating set.

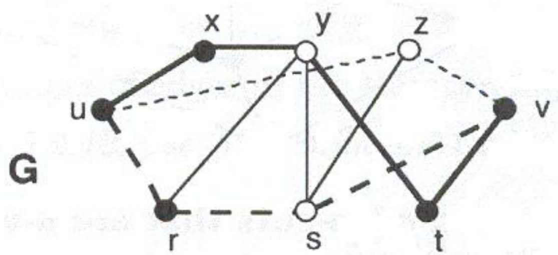


Figure 5.3.2

The next theorem proved by K. Menger in 1927 establishes a *strong duality* between the two optimization problems introduced earlier.

The proof given here is an example of a traditional style proof in graph theory. The theorem can also be proven e.g. based on the theory of network flows.

strict paths

Definition Let *W* be a set of vertices in a graph *G* and *x* another vertex not in *W.* A **strict *x-W* path** is a path joining *x* to a vertex in *W* and containing no other vertex of *W.* A **strict *W-x* path** is the reverse of a strict *x-W* path (i.e. its sequence of vertices and edges is in reverse order).

Example: Corresponding to the *u-v* separating set *W = {y,s,z}* in the graph below, the vertex sequences ⟨*u,x,y*⟩, ⟨*u,r,y*⟩, ⟨*u,r,s*⟩, and ⟨*u,z*⟩ represent the four strict *u-W* paths, and the three strict *W-v* paths are given by ⟨*z,v*⟩, ⟨*y,t,v*⟩, and ⟨*s,v*⟩.

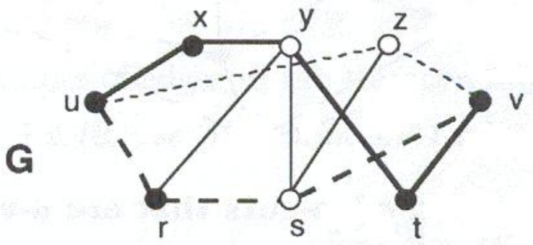


Figure 5.3.2

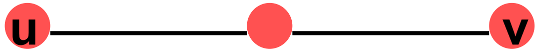
Menger's Theorem

Theorem 5.3.4 [Menger, 1927] Let *u* and *v* be distinct, non-adjacent vertices in a connected graph *G.*

Then the maximum number of internally disjoint *u*-*v* paths in *G* equals the minimum number of vertices needed to separate *u* and *v.*

Proof: The proof uses induction on the number of edges.

The smallest graph that satisfies the premises of the theorem is the path graph from *u* to *v* of length 2, and the theorem is trivially true for this graph.



Assume that the theorem is true for all connected graphs having fewer than *m* edges, e.g. for some $m \geq 3$.

Now suppose that *G* is a connected graph with *m* edges, and let *k* be the minimum number of vertices needed to separate the vertices *u* and *v.*

By Corollary 5.3.2, it suffices to show that there exist *k* internally disjoint *u*-*v* paths in *G.*

Since this is clearly true if $k = 1$ (since *G* is connected), assume $k \geq 2$.

Proof of Menger's Theorem

Assertion 5.3.4a If G contains a u-v path of length 2, then G contains k internally disjoint u-v paths.

Proof of 5.3.4a: Suppose that $\mathcal{P} = \langle u, e_1, x, e_2, v \rangle$ is a path in G of length 2.

Let W be a smallest u-v separating set for the vertex-deletion subgraph $G - x$. Since $W \cup \{x\}$ is a u-v separating set for G, the minimality of k implies that $|W| \geq k - 1$. By the induction hypothesis, there are at least $k - 1$ internally disjoint $u - v$ paths in $G - x$. Path $\mathcal{P}$ is internally disjoint from any of these, and, hence, there are k internally disjoint u-v paths in G. □

If there is a u-v separating set that contains a vertex adjacent to *both* vertices u and v, then Assertion 5.3.4a guarantees the existence of k internally disjoint u-v paths in G.

The argument for *distance* $(u,v) \geq 3$ is now broken into two cases, according to the kinds of u-v separating sets that exist in G.

Proof of Menger's Theorem

In Case 1, there exists a *u*-*v* separating set *W,* as depicted in the left side of the figure below, where neither *u* nor *v* is adjacent to every vertex of *W* .

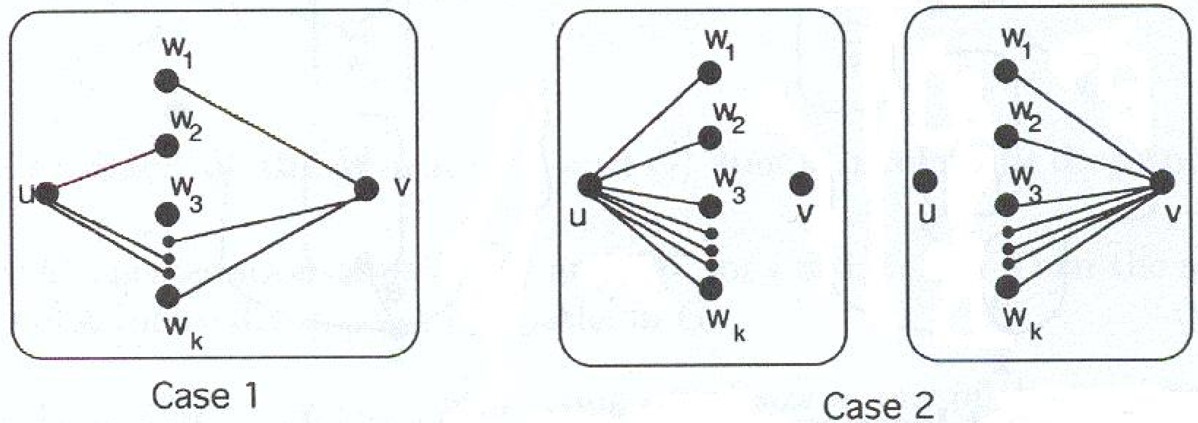


Figure 5.3.3 The two cases remaining in the proof of Menger's theorem.

In Case 2, no such separating set exists.

Thus, in every *u*-*v* separating set for Case 2, either every vertex is adjacent to *u* or every vertex is adjacent to *v,* as shown on the right side.

Proof of Menger's Theorem

Case 1: There exists a *u*-*v* separating set $W = \{w_1, w_2, \ldots, w_k\}$ of vertices in *G* of minimum size *k,* such that neither *u* nor *v* is adjacent to every vertex in *W.*

Let G_u be the subgraph induced on the union of the edge-sets of all strict *u*-*W* paths in *G,* and let G_v be the subgraph induced on the union of edge-sets of all strict *W*-*u* paths (see Fig. below).

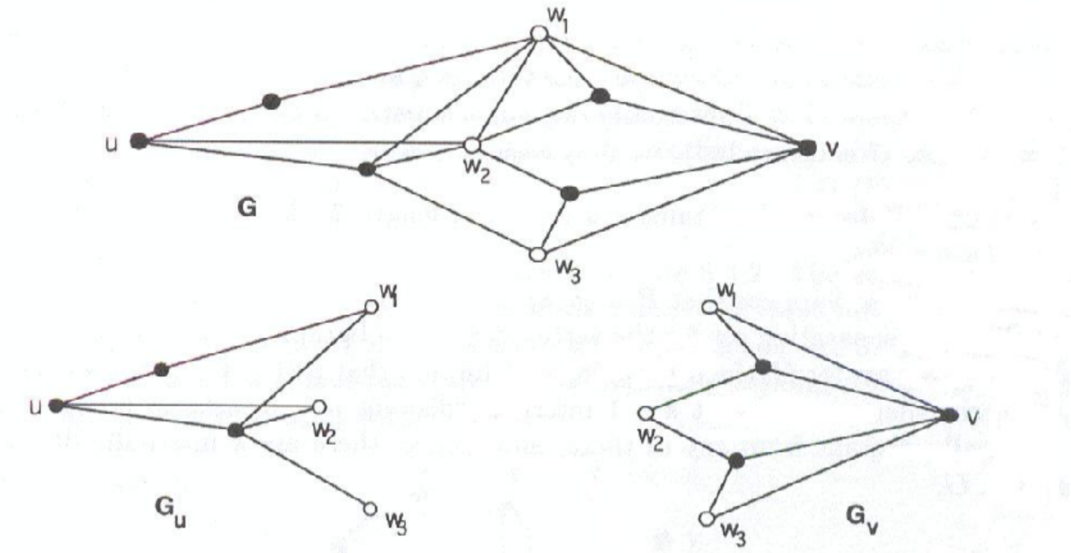


Figure 5.3.4 An example illustrating the subgraphs G_u and G_v.

Proof of Menger's Theorem

Assertion 5.3.4b: Both of the subgraphs G_u and G_v have more than k edges.

Poof of 5.3.4b: For each $w_i \in W$, there is a u-v path P_{wi} in G on which w_i is the only vertex of W (otherwise, $W - \{w_i\}$ would still be a u-v separating set, contradicting the minimality of W).

The u-w_i subpath of P_{wi} is a strict u-W path that ends at w_i.

Thus, the final edge of this strict u-W path is different for each w_i.

Hence, G_u has at least k edges.

The only way G_u could have exactly k edges would be if each of these strict u-W paths consisted of a single edge joining u and w_i, $i = 1, ..., k$.

But this is ruled out by the condition for Case 1. Therefore, G_u has more than k edges. A similar argument shows that G_v also has more than k edges. □

Proof of Menger's Theorem

Assertion 5.3.4c: The subgraphs G_u and G_v have no edges in common.

Poof of 5.3.4c: By way of contradiction, suppose that the subgraphs G_u and G_v have an edge *e* in common. By the definitions of G_u and G_v, edge *e* is an edge of both a strict *u*-*W* path and a strict *W*-*v* path.

A **strict *x*-*W* path** is a path joining *x* to a vertex in *W* and containing no other vertex of *W*.
A **strict *W*-*x* path** is the reverse of a strict *x*-*W* path (i.e. its sequence of vertices and edges is in reverse order).

Hence, at least one of the endpoints of *e*, say *x*, is not a vertex in the *u*-*v* separating set *W* (see Fig. below). This implies the existence of a *u*-*v* path in *G*-*W*, which contradicts the definition of *W*. □

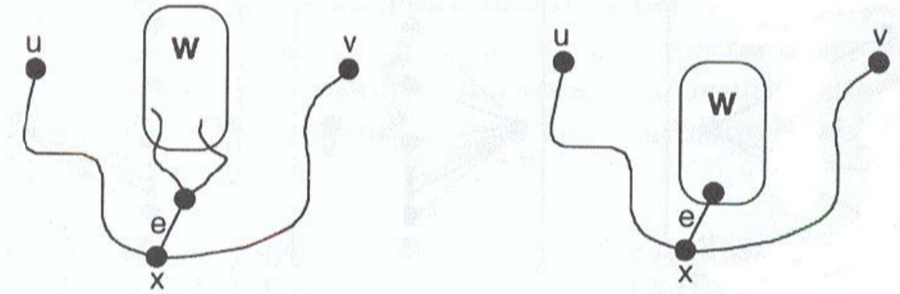


Figure 5.3.5 At least one of the endpoints of edge *e* lies outside *W*.

Proof of Menger's Theorem

We now define two auxiliary graphs G_u^* and G_v^*:

G_u^* is obtained from G by replacing the subgraph G_v with a new vertex v^* and drawing an edge from each vertex in W to v^*, and

G_v^* is obtained by replacing G_u with a new vertex u^* and drawing and edge from u^* to each vertex in W (see Fig. below).

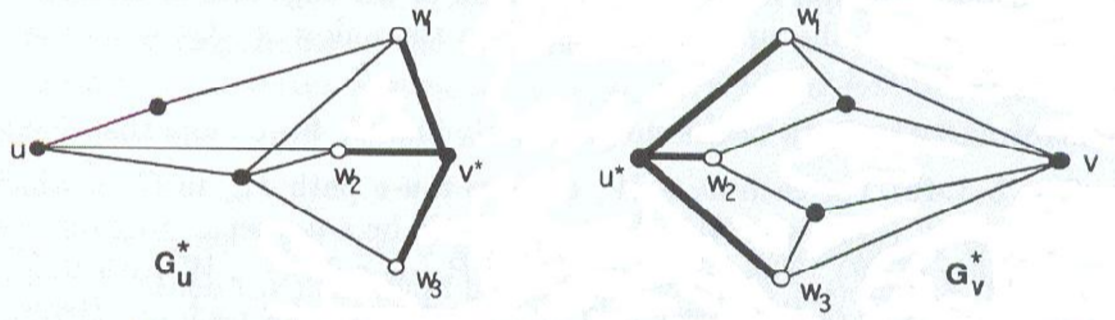


Figure 5.3.6 Illustration for the construction of graphs G_u^* and G_v^*.

Proof of Menger's Theorem

Assertion 5.3.4d: Both of the auxiliary graphs G_u^* and G_v^* have fewer edges than G.

Proof of 5.3.4d: The following chain of inequalities shows that graph G_u^* has fewer edges than G.

$$\left|E_G\right| \geq \left|E_{G_u \cup G_v}\right| \qquad \text{since } G_u \cup G_v \text{ is a subgraph of } G$$

$$\underset{5.3.4c}{=} \left|E_{G_u}\right| + \left|E_{G_v}\right|$$

$$\underset{5.3.4b}{>} \left|E_{G_u}\right| + k$$

$$= \left|E_{G_u^*}\right| \qquad \text{by the construction of } G_u^*$$

A similar argument shows that G_v^* also has fewer edges than G. □

Proof of Menger's Theorem

By the construction of graphs G_u^* and G_v^*, every u-v^* separating set in graph G_u^* and every u^*-v separating set in graph G_v^* is a u-v separating set in graph G. Hence, the set W is a smallest u-v^* separating set in G_u^* and a smallest u^*-v separating set in G_v^*.

Since G_u^* and G_v^* have fewer edges than G, the induction hypothesis implies the existence of two collections, $\mathcal{P}_u^*$ and $\mathcal{P}_v^*$ of k internally disjoint u-v^* paths in G_u^* and k internally disjoint u^*-v paths in G_v^*, respectively (see Fig.).

For each w_i, one of the paths in $\mathcal{P}_u^*$ consists of a u-w_i path P_i' in G plus the new edge from w_i to v^*, and one of the paths in $\mathcal{P}_v^*$ consists of the new edge from u^* to w_i followed by a w_i-v path P_i'' in G.

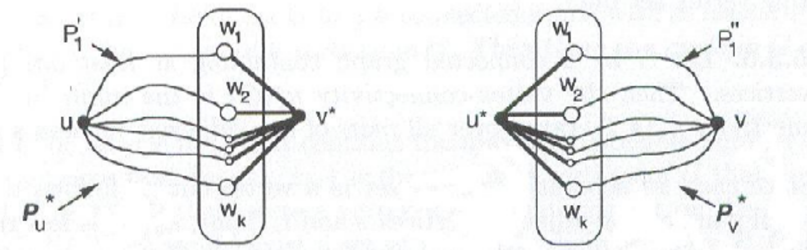


Figure 5.3.7 Each of the graphs G_u^* and G_v^* has k internally disjoint paths.

Let P_i be the concatenation of paths P_i' and P_i'', for $i = 1, ..., k$. Then the set $\{P_i\}$ is a collection of k internally disjoint u-v paths in G. □ (Case 1)

Proof of Menger's Theorem

Case 2: Suppose that for each *u*-*v* separating set of size *k*, one of the vertices *u* or *v* is adjacent to all the vertices in that separating set.

Let $P = \langle u, e_1, x_1, e_2, x_2, \ldots, v \rangle$ be a shortest *u*-*v* path in *G*.
By Assertion 5.3.4a, we can assume that *P* has length at least 3 and that vertex x_1 is not adjacent to vertex *v*.
By Proposition 5.1.3, the edge-deletion subgraph $G - e_2$ is connected.
Let *S* be a smallest *u*-*v* separating set in subgraph $G - e_2$ (see Fig.).

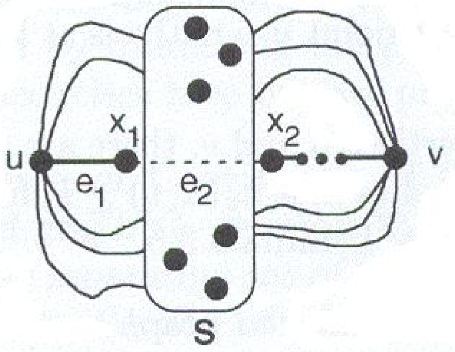


Figure 5.3.8 Completing Case 2 of Menger's theorem.

Proof of Menger's Theorem

Then S is a u-v separating set in the vertex-deletion subgraph $G - x_1$ (since $G - x_1$ is a subgraph of $G - e_2$). Thus, $S \cup \{x_1\}$ is a u-v separating set in G, which implies that $| S | \geq k - 1$, by the minimality of k. On the other hand, the minimality of $| S |$ in $G - e_2$ implies that $| S | \leq k$, since every u-v separating set in G is also a u-v separating set in $G - e_2$.

If $| S | = k$, then, by the induction hypothesis, there are k internally disjoint u-v paths in $G - e_2$ and, hence, in G.

If $| S | = k - 1$, then $x_i \notin S$, $i = 1,2$ (otherwise $S - \{x_i\}$ would be a u-v separating set in $G - e_2$, contradicting the minimality of k).

Thus, the sets $S \cup \{x_1\}$ and $S \cup \{x_2\}$ are both of size k and both u-v separating sets of G. The condition for Case 2 and the fact that vertex x_1 is not adjacent to v imply that every vertex in S is adjacent to vertex u.

Hence, no vertex in S is adjacent to v (lest there be a u-v path of length 2).

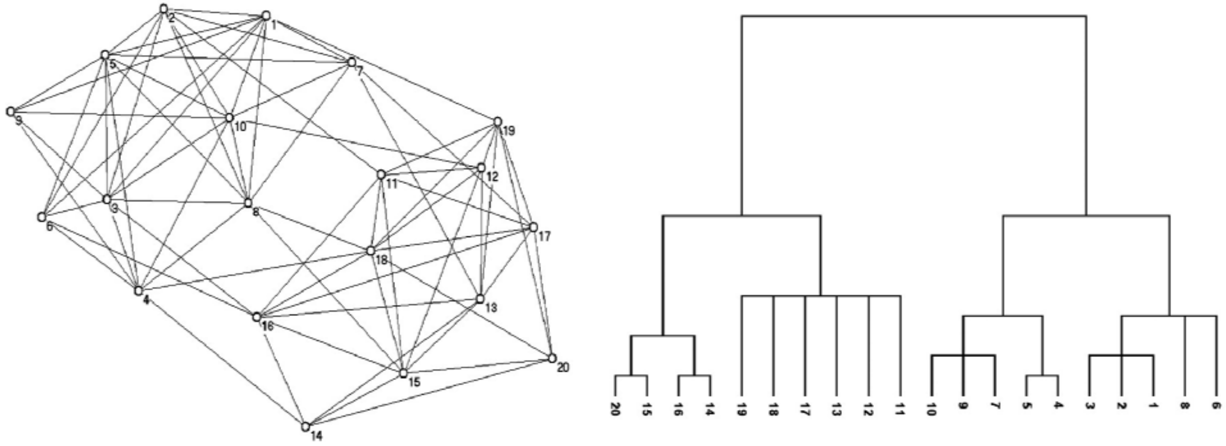
But then the condition of Case applied to $S \cup \{ x_2 \}$ implies that vertex x_2 is adjacent to vertex u, which contradicts the minimality of path P and completes the proof. □

Strategies to detect communities in networks

„Community“ stands for module, class, group, cluster, ...

Define **community** as a subset of nodes within the graph such that connections between the nodes are denser than connections with the rest of the network.

The detection of community structure is generally intended as a procedure for mapping the network onto a tree („dendogram“ in social sciences).



Leaves: nodes;

Branches join nodes or (at higher level) groups of nodes.

Radicchi et al. PNAS 101, 2658 (2004)

Agglomerative algorithms for mapping to tree

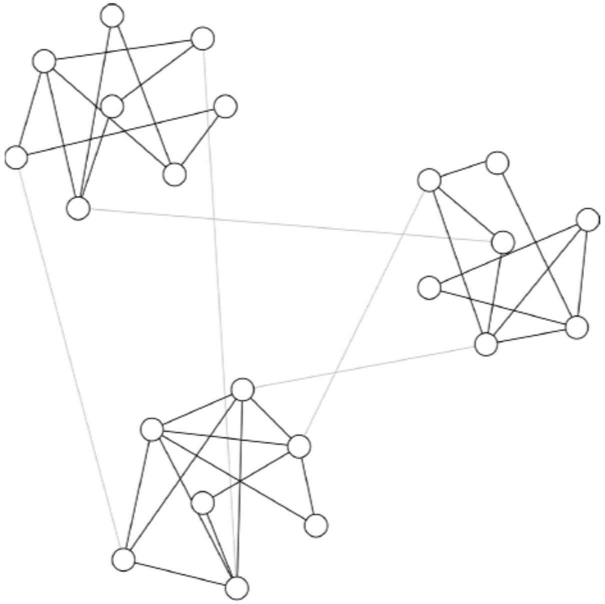
Traditional method to perform this mapping: **hierarchical clustering**.

For every pair *i,j* of nodes in the network compute **weight** W_{ij} that measures how closely connected the vertices are.

Starting from the set of all nodes and no edges, **links** are **iteratively added** between pairs of nodes in order of decreasing weight.

In this way nodes are grouped into larger and larger communities, and the tree is built up to the root, which represents the whole network.

→ **„agglomerative“ algorithm**



Here: 3 communities of densely connected vertices (circles with solid lines) with a much lower density of connections (gray lines) between them.

Girven, Newman, PNAS 99, 7821 (2002)
Radicchi et al. PNAS 101, 2658 (2004)

Possible definitions of the weights

(1) number of **node-independent paths** between vertices
2 paths that connect the same pair of vertices are said to be node-independent if they share none of the same vertices other than their initial and final vertices.

(2) **edge-independent paths**.

Menger‘s theorem states that the number of node-independent (edge-independent) paths between 2 vertices *i* and *j* in a graph is equal to the minimum number of vertices (edges) that must be removed from the graph to disconnect *i* and *j* from one another
→ these numbers are a measure of the robustness of the network to deletion of nodes (edges).

Girven, Newman, PNAS 99, 7821 (2002)

Possible definitions of the weights (II)

(3) count **total number** of **paths** that run between them
(not just those that are node- or edge-independent).

Because the number of paths between any 2 vertices is either 0 or infinite, one typically **weighs paths** of length *l* by a factor α^l with small α so that the weighted count of number of paths converges.
Thus long paths contribute exponentially less weight than short paths.

These node- or edge-dependent path definitions for weights work okay for certain community structures, but show typical pathologies.

Girven, Newman, PNAS 99, 7821 (2002)

Problems

In particular, both counting of node- and edge-independent paths has a tendency to separate single peripheral vertices from the communities to which they should rightly belong.

If a vertex is, e.g., connected to the rest of a network by only a single edge then, to the extent that it belongs to any community, it should clearly be considered to belong to the community at the other end of that edge.

Unfortunately, both the numbers of independent paths and the weighted path counts for such vertices are small and hence single nodes often remain isolated from the network when the communities are constructed.

This and other pathologies, make the hierarchical clustering method, although useful, far from perfect.

Girven, Newman, PNAS 99, 7821 (2002)

New strategy: Use “betweenness” as definition of weights

Focus on those edges that are least central, that are „between“ communities.

Define **edge betweenness** of an edge as the number of shortest paths between pairs of vertices that run along it.
If there is more than one shortest path between a pair of vertices, each path is given equal weight such that the total weight of all of the paths is 1.

If a network contains communities or groups that are only loosely connected by a few intergroup edges, then all shortest paths between different communities must go along one of these few edges.
→ the edges connecting communities will have high edge betweenness.

By removing these edges we separate groups from one another and so reveal the underlying community structure of the graph.

Girven, Newman, PNAS 99, 7821 (2002)

GN Algorithm

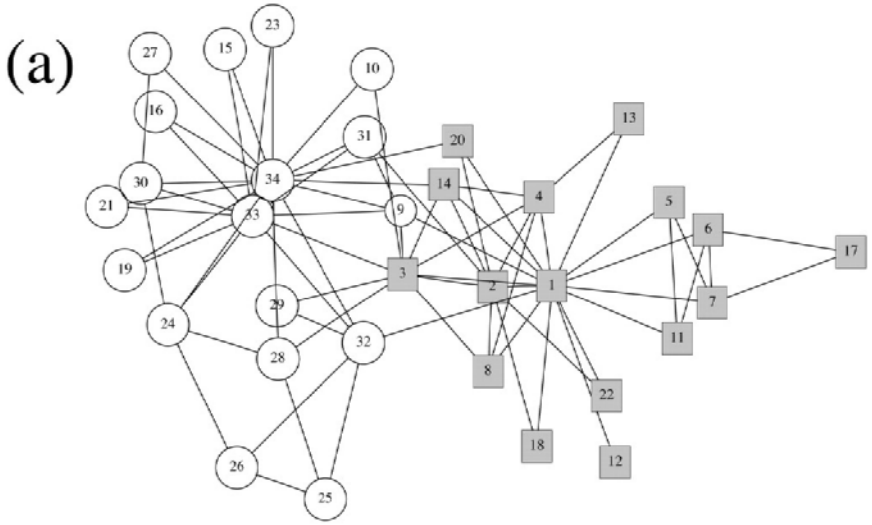
1. Calculate betweenness for all *m* edges in a graph of *n* vertices (can be done in $O(mn)$ time).
2. Remove the edge with the highest betweenness.
3. Recalculate betweenness for all edges affected by the removal.
4. Repeat from step 2 until no edges remain.

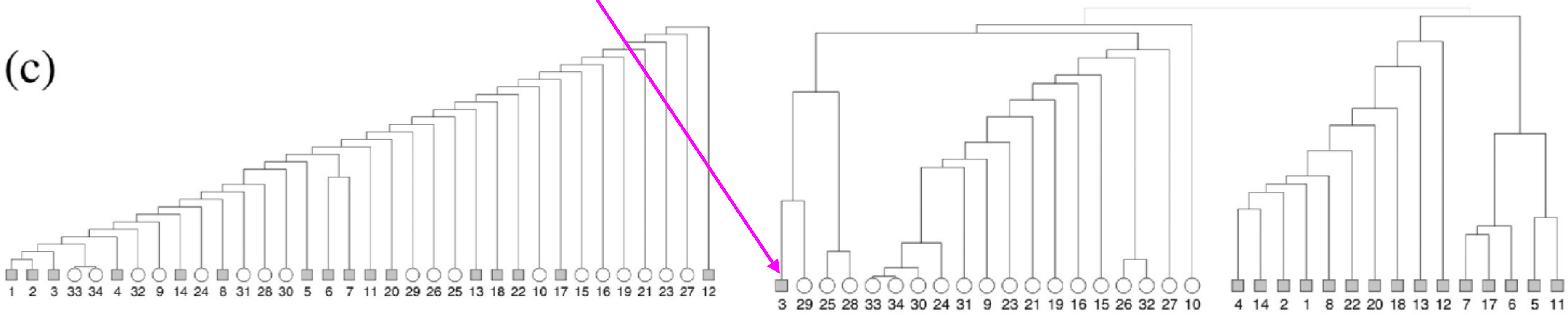
Because step 3 has to be done for all edges, the algorithm runs in worst-case time $O(m^2n)$.

Girven, Newman, PNAS 99, 7821 (2002)

Application of Girvan&Newman Algorithm

(*a*) The friendship network from Zachary's karate club study. The instructor and the administrator are represented by nodes 1 and 34.
Nodes associated with the club administrator's fraction are drawn as circles, those associated with the instructor's faction are drawn as squares.
(*b*) Hierarchical tree showing the complete community structure for the network calculated by using the Girven-Newman algorithm. The initial split of the network into two groups is in agreement with the actual factions observed by Zachary, except for the misclassified node 3.
(*c*) Hierarchical tree calculated by using edge-independent path counts, which fails to extract the known community structure of the network.





Girven, Newman, PNAS 99, 7821 (2002)

Divisive algorithms for mapping to tree

Reverse order of tree construction compared to agglomerative algorithms:

start with the whole graph and iteratively cut the edges
→ divide network progressively into smaller and smaller disconnected subnetworks identified as the communities.

Crucial point: how to select the edges to be cut.
One possible way is given by the Girven & Newman algorithm (GN)

Main problem of the GN algorithm: it requires the repeated evaluation of a global property, the betweenness, for each edge whose value depends on the properties of the whole system.
→ becomes computationally very expensive for networks with e.g. ≥ 10000 nodes.

Radicchi et al. PNAS 101, 2658 (2004)

Faster algorithm

Introduce divisive algorithm that only requires the consideration of local quantities.

Need: quantity that can single out edges connecting nodes belonging to different communities.

Consider **edge-clustering coefficient**:
number of triangles to which a given edge belongs divided by the number of triangles that might potentially include it, given the degrees of the adjacent nodes.

For the edge-connecting node *i* to node *j*, the edge-clustering coefficient is

$$C_{i,j}^{(3)} = \frac{z_{i,j}^{(3)} + 1}{\min[(k_i - 1), (k_j - 1)]}$$

where $z_{i,j}^{(3)}$ is the number of triangles built on that edge and
$\min[(k_i - 1), (k_j - 1)]$ is the maximal possible number of them.
1 is added to $z_{i,j}^{(3)}$ to remove degeneracy for $z_{i,j}^{(3)} = 0$.

Radicchi et al. PNAS 101, 2658 (2004)

Faster algorithm

Edges connecting nodes in different communities are included in few or no triangles and tend to have small values of $C_{i,j}^{(3)}$.

On the other hand, many triangles exist within clusters.

By considering higher order cycles one can define coefficients of order g

$$C_{i,j}^{(g)} = \frac{z_{i,j}^{(g)} + 1}{s_{i,j}^{(g)}}$$

where $z_{i,j}^{(g)}$ is the number of cyclic structures of order *g* the edge *(i,j)* belongs to, and $s_{i,j}^{(g)}$ is the number of possible cyclic structures of order *g* that can be built given the degrees of the nodes.

Define, for every *g*, a dectection algorithm that works exactly as the GN method with the difference that, at every step, the removed edges are those with the smallest value of $C_{i,j}^{(g)}$.

By considering increasing values of *g*, one can smoothly interpolate between a local and a nonlocal algorithm.

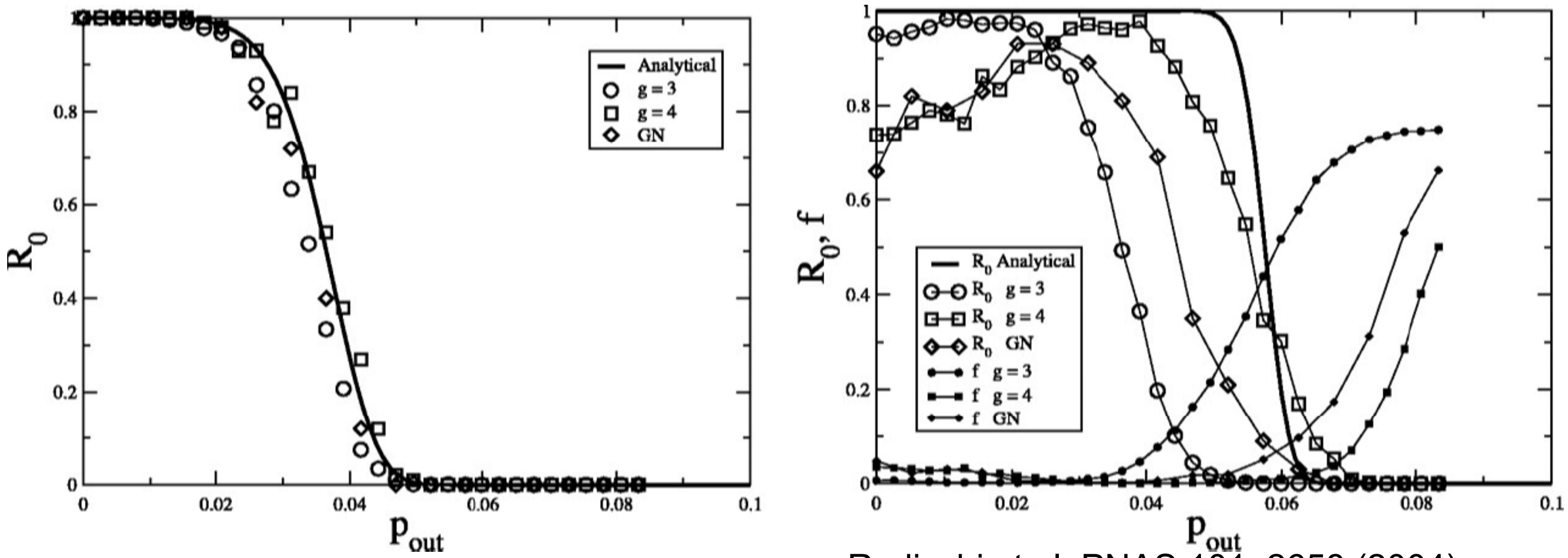
Radicchi et al. PNAS 101, 2658 (2004)

Comparison with GN method

Test of the efficiency of the different algorithms in the analysis of the artificial graph with four communities. Here *N* = 128 and p_{in} is changed with p_{out} to keep the average degree equal to 16.

(*Left*) Strong definition: fraction of successes for the different algorithms compared with the analytical probability that four communities are actually defined.

(*Right*) Weak definition: in addition to the same quantities plotted in *Left*, here we report, for every algorithm, the fraction *f* of nodes not correctly classified.



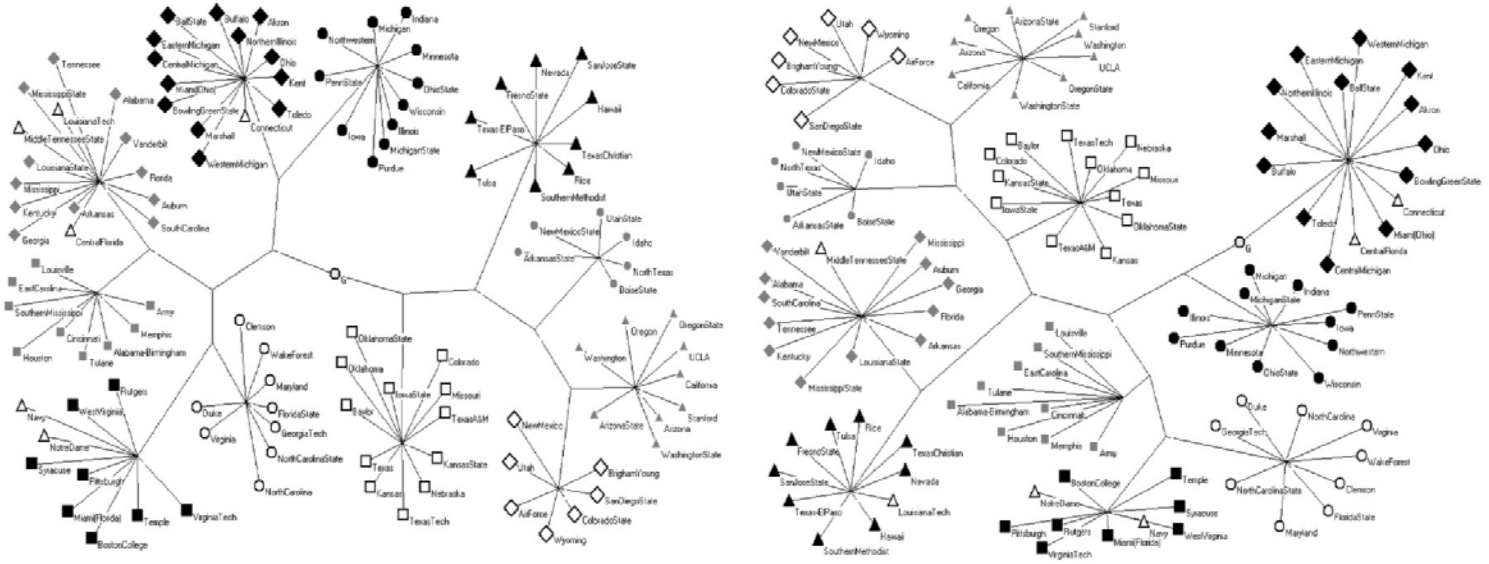
Radicchi et al. PNAS 101, 2658 (2004)

Comparison with GN algorithm

Plot of the dendrograms for the network of college football teams, obtained by using the GN algorithm (*Left*) and the Radicchi algorithm with *g* = 4 (*Right*). Different symbols denote teams belonging to different conferences.

Note: to reduce the amount of travelling in US college football, the teams play a large number of games against teams in the same conference and only a small number of games against teams from other conferences. Here, the algorithm tries to discover which team belongs to which conference from the names of the teams in all games that were played.

In both cases, the observed communities perfectly correspond to the conferences, with the exception of the six members of the „Independent conference“, which are misclassified.



Radicchi et al. PNAS 101, 2658 (2004)