

1 Edge Betweenness Dendrogram

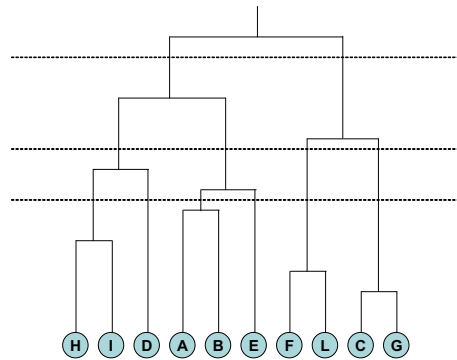


Figure 1: Example from Caldarelli 2007.

1.1 Existing Software

<http://www.sandia.gov/bahendr/chaco.html> (CHACO)

<http://glaros.dtc.umn.edu/gkhome/views/metis/index.html> (METIS)

<http://www.cs.utexas.edu/users/dml/Software/gracclus.html> (GRACCLUS)

2 Minimum Spanning Tree

A spanning tree for a connected, undirected graph $G = (V, E)$ is a subgraph of G that is an undirected tree and contains all the vertices of G .

In a weighted graph $G = (V, E, W)$, the weight of a subgraph is the sum of the weights of the edges in the subgraph.

A minimum spanning tree for a weighted graph is a spanning tree with the minimum weight.

2.1 Prim/Jarnik's Minimum Spanning Tree Algorithm

1. Select an arbitrary starting vertex (the root).
2. Branch out from the tree constructed so far by choosing an edge at each iteration, attaching the edge to the tree (the edge that has minimum weight among all edges that can be attached), and adding to the tree the vertex associated with the edge.
3. Vertices are divided into three disjoint categories: *tree vertices* in the tree constructed so far, *fringe vertices* not in the tree, but adjacent to some vertex in the tree, and *unseen vertices* including all others.

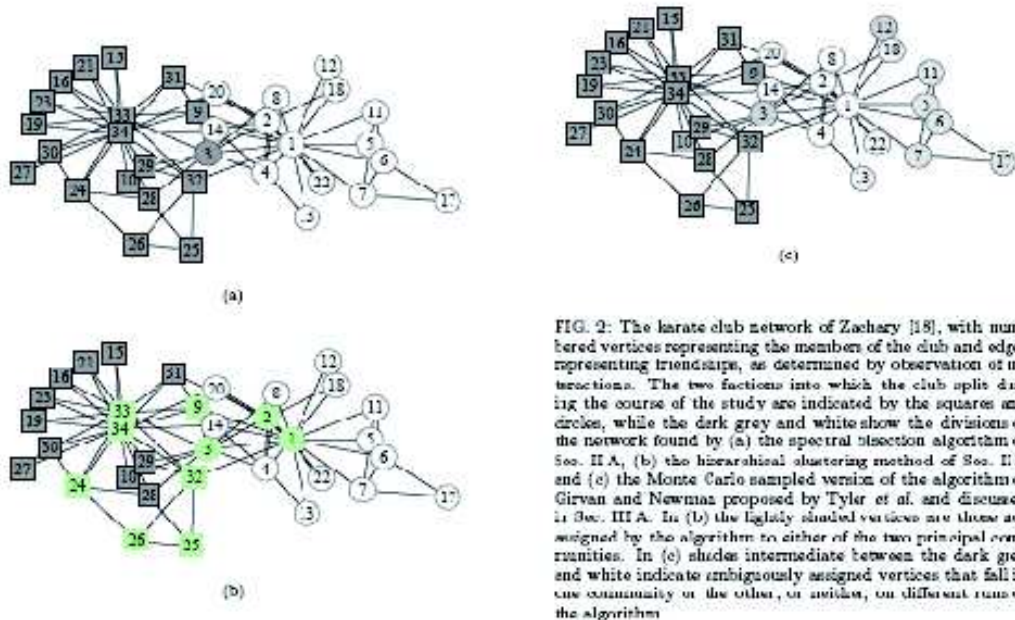


FIG. 2: The karate club network of Zachary [18], with numbered vertices representing the members of the club and edges representing friendships, as determined by observation of interactions. The two factions into which the club split during the course of the study are indicated by the squares and circles, while the dark grey and white show the divisions of the network found by (a) the spectral bisection algorithm of Sec. II A, (b) the hierarchical clustering method of Sec. II D and (c) the Monte Carlo sampled version of the algorithm of Girvan and Newman proposed by Tyler et al. and discussed in Sec. III A. In (b) the lightly shaded vertices are those not assigned by the algorithm to either of the two principal communities. In (c) shades intermediate between the dark grey and white indicate ambiguously assigned vertices that fall in one community or the other, or switch, on different runs of the algorithm.

Figure 2: Detecting community structure in networks, M. E. J. Newman, Eur. Phys. J. B 38, 321-330 (2004).

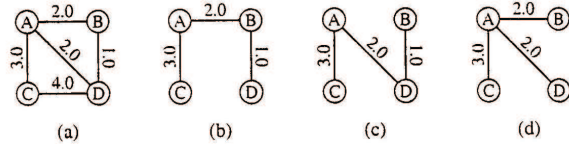


Figure 3: MST and mincut slides by Rada Mihalcea.

2.2 Prim’s Algorithm

Algorithm *PrimMinSpanningTree(G)*
 Initialize all nodes as unseen
 Select an arbitrary vertex *s* to start the tree
 Reclassify it as tree
 Reclassify all vertices adjacent to *s* as fringe
 While there are fringe vertices
 Select an edge of minimum weight between a tree vertex *t* and a fringe vertex *v*
 Reclassify *v* as tree; add edge *tv* to the tree
 Reclassify all unseen vertices adjacent to *v* as fringe

2.3 Properties of Minimum Spanning Trees

Definition: Minimum spanning tree property.
 Let *G* be a connected, weighted graph $G = (V, E, W)$, and let *T* be any spanning tree of *G*. Suppose that for every edge \widehat{uv} of *G* that is not in *T*:
 if \widehat{uv} is added to *T*, then it creates a cycle;
 \widehat{uv} is a maximum-weight edge on that cycle.

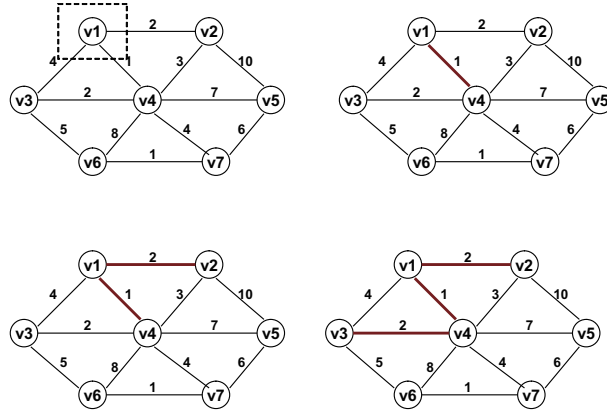


Figure 4: Prim's Algorithm.

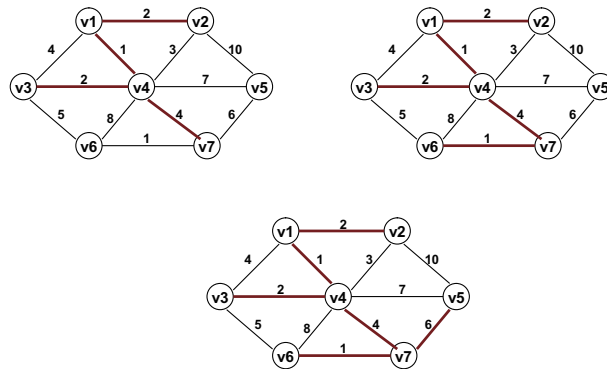


Figure 5: Prim's Algorithm.

The tree T is said to have the minimum spanning tree property. A graph can admit several minimum spanning trees.

Lemma: In a connected, weighted graph $G = (V, E, W)$, if T_1 and T_2 are two spanning trees that have the MST property, then they have the same total weight.

The number of edges in a minimum spanning tree is $|V| - 1$.

3 Flow Networks

What if weights in a graph are maximum capacities of some flow of material?

Examples: Pipe network to transport fluid (e.g., water, oil) where edges denote pipes and vertices denote junctions of pipes. Data communication network where edges denote network connections of different capacity and vertices denote routers (do not produce or consume data just move it).

Concepts (informally): Source vertex s (where material is produced). Sink vertex t (where material is consumed). For all other vertices - what goes in must go out.

Goal: maximum rate of material flow from source to sink.

3.1 Formalization

Flow network - $G = (V, E)$: Directed, each edge has capacity $c(u, v) \geq 0$. Two special vertices: source s , and sink t . For any other vertex v , there is a path $s \rightarrow \dots \rightarrow v \rightarrow \dots \rightarrow t$.

Flow - $f : V \times V \rightarrow R$: Capacity constraint, $\forall u, v \in V, f(u, v) \leq c(u, v)$. Skew symmetry, $\forall u, v \in V, f(u, v) = -f(v, u)$. Flow conservation, $\forall u \in V - \{s, t\}, \sum_{v \in V} f(u, v) = f(u, V) = 0$ or $\sum_{v \in V} f(v, u) =$

$$f(V, u) = 0.$$

3.2 Cancellation of Flows

Do we want to have positive flows going in both directions between two vertices?
 No! such flows cancel (maybe partially) each other.

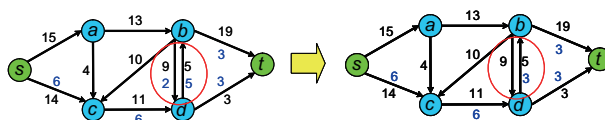


Figure 6: Cancellation of flows.

3.3 Maximum Flow

Want to maximize the total value of the flow f : $|f| = \sum_{v \in V} f(s, v) = f(s, V) = f(V, t)$.

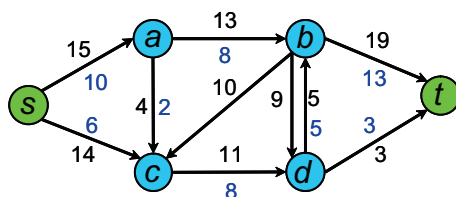
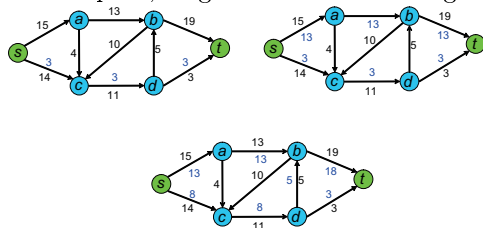


Figure 7: Find a flow of maximum value.

3.4 Augmenting Path

Idea for the algorithm: If we have some flow, and can find a path p from s to t (augmenting path), such that there is $a > 0$, and for each edge (u, v) in p we can add a units of flow: $f(u, v) + a \leq c(u, v)$; then mark down that path, to get a better flow. Augmenting path in this graph Figure 7?



3.5 Ford-Fulkerson Method

Ford – Fulkerson(G, s, t)

1. Initialize flow f to 0 everywhere
2. **While** there is an augmenting path p **do**
 augment flow f along p
3. Return f

How do we find augmenting path? How much additional flow can we send through that path?

3.5.1 Residual Network

How do we find augmenting path?

It is any path in residual network: Residual capacities: $c_f(u, v) = c(u, v) - f(u, v)$. Residual network: $G_f = (V, E_f)$, where $E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$

3.5.2 Residual Capacity of a Path

How much additional flow can we send through an augmenting path?

Residual capacity of a path p in G_f : $c_f(p) = \min\{c_f(u, v) : (u, v) \in p\}$.

Doing augmentation: $\forall (u, v) \in p$, add $c_f(p)$ to $f(u, v)$ (and subtract from $f(v, u)$).

Resulting flow is a valid flow with a larger value. What is the residual capacity of the path (s, a, b, t) in Figure 7?

3.5.3 Ford-Fulkerson Method with Details

Ford – Fulkerson(G, s, t)

1. **for** each edge $(u, v) \in G.E$ **do**
2. $f(u, v) = f(v, u) = 0$
3. **while** \exists path p from s to t in residual network G_f **do**
4. $c_f = \min\{c_f(u, v) : (u, v) \in p\}$
5. **for** each edge (u, v) in p **do**
6. $f(u, v) = f(u, v) + c_f$
7. $f(v, u) = -f(u, v)$
8. **return** f

The algorithms based on this method differ in how they choose p in step 3.

3.5.4 Ford-Fulkerson Method: Example

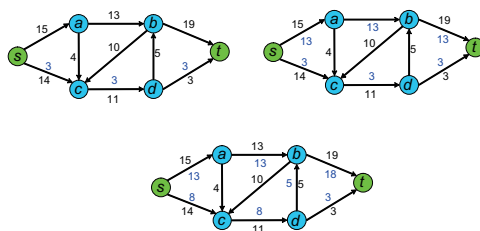


Figure 8: Ford-fulkerson method: example.

4 Cuts

An $s - t$ cut is a partition (S, T) such that $s \in S, t \in T$.

The capacity of an $s - t$ cut (S, T) is:

$$\sum_{e \text{ out of } S} u(e) = \sum_{(v,w) \in E, v \in S, w \in T} u(v, w)$$

Min $s - t$ cut: find an $s - t$ cut of minimum capacity.

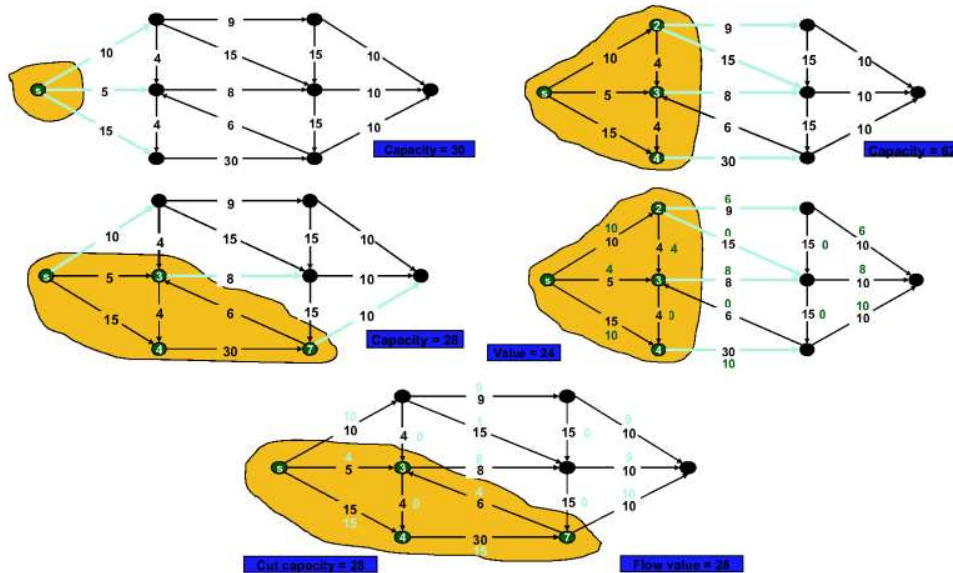


Figure 9: $s - t$ cut I-V.

4.1 Flows and Cuts

Lemma 1. Let f be a flow, and let (S, T) be a cut. Then, the net flow sent across the cut is equal to the amount reaching t .

$$\sum_{e \text{ out of } S} f(e) - \sum_{e \text{ in to } S} f(e) = \sum_{e \text{ out of } S} f(e) = |f|$$

4.2 Max Flow and Min Cut

Corollary. Let f be a flow, and let (S, T) be a cut. If $|f| = \text{cap}(S, T)$, then f is a max flow and (S, T) is a min cut.

Max-flow/min-cut Theorem (Ford-Fulkerson): In any network, the value of the max flow is equal to the value of the min cut.