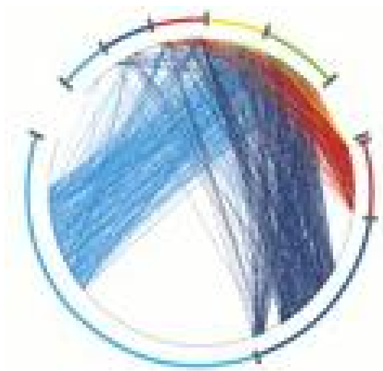
Models and Algorithms for Complex Networks

Graph Clustering and Network Communities



Clustering

- Given a set of objects V, and a notion of similarity (or distance) between them, partition the objects into disjoint sets $S_1, S_2, \ldots, S_k$, such that objects within the each set are similar, while objects across different sets are dissimilar

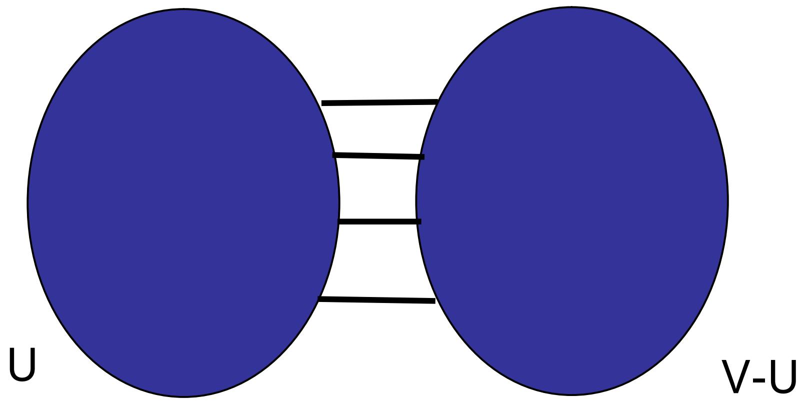
Graph Clustering

- Input: a graph G=(V,E)
  - edge (u,v) denotes similarity between u and v
  - weighted graphs: weight of edge captures the degree of similarity
- Clustering: Partition the nodes in the graph such that nodes within clusters are well interconnected (high edge weights), and nodes across clusters are sparsely interconnected (low edge weights)
  - most graph partitioning problems are NP hard

Measuring connectivity

- What does it mean that a set of nodes are well or sparsely interconnected?

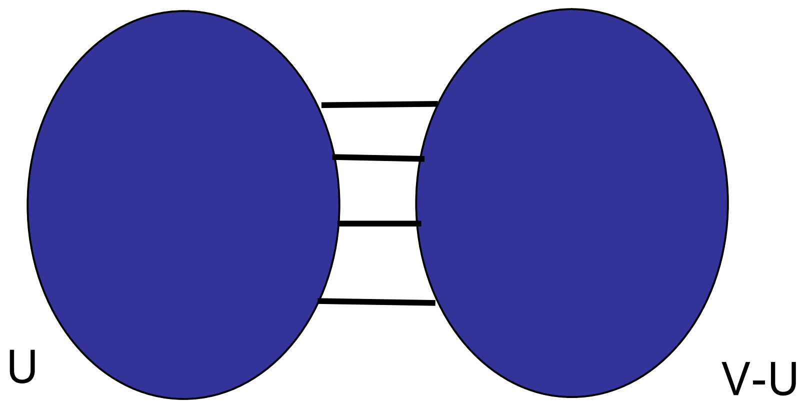
- min-cut: the min number of edges such that when removed cause the graph to become disconnected
  - small min-cut implies sparse connectivity
  - $\min_U E(U, V-U) = \sum_{i \in U} \sum_{j \in V-U} A[i, j]$

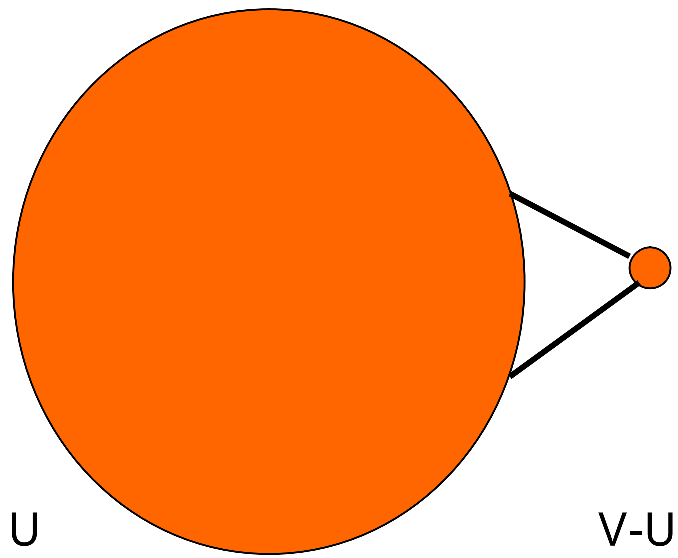


Measuring connectivity

- What does it mean that a set of nodes are well interconnected?

- min-cut: the min number of edges such that when removed cause the graph to become disconnected
  - not always a good idea!





Graph expansion

- Normalize the cut by the size of the smallest component
- Cut ratio:

$$\alpha = \frac{E(U, V - U)}{\min\{|U|, |V - U|\}}$$

- Graph expansion:

$$\alpha(G) = \min_{U} \frac{E(U, V - U)}{\min\{|U|, |V - U|\}}$$

- We will now see how the graph expansion relates to the eigenvalue of the adjacency matrix A

Spectral analysis

- The Laplacian matrix $L = D – A$ where
  - A = the adjacency matrix
  - $D = \mathrm{diag}(d_1,d_2,\ldots,d_n)$
    - d_i = degree of node i

- Therefore
  - $L(i,i) = d_i$
  - $L(i,j) = -1$, if there is an edge (i,j)

Laplacian Matrix properties

- The matrix L is symmetric and positive semi-definite
  - all eigenvalues of L are positive
- The matrix L has 0 as an eigenvalue, and corresponding eigenvector $w_1 = (1,1,\dots,1)$
  - $\lambda_1 = 0$ is the smallest eigenvalue

The second smallest eigenvalue

- The second smallest eigenvalue (also known as Fielder value) λ_2 satisfies

$$\lambda_2 = \min_{x \perp w_1, \|x\|=1} x^T L x$$

- The vector that minimizes λ_2 is called the Fielder vector. It minimizes

$$\lambda_2 = \min_{x \neq 0} \frac{\sum_{(i,j)\in E} \left(x_i - x_j\right)^2}{\sum_i x_i^2} \quad \text{where} \quad \sum_i x_i = 0$$

Spectral ordering

- The values of x minimize

$$\min_{x \neq 0} \frac{\sum_{(i,j) \in E} (x_i - x_j)^2}{\sum_i x_i^2} \qquad \sum_i x_i = 0$$

- For weighted matrices

$$\min_{x \neq 0} \frac{\sum_{(i,j)} A[i,j](x_i - x_j)^2}{\sum_i x_i^2} \qquad \sum_i x_i = 0$$

- The ordering according to the x_i values will group similar (connected) nodes together

- Physical interpretation: The stable state of springs placed on the edges of the graph

Spectral partition

- Partition the nodes according to the ordering induced by the Fielder vector
- If $u = (u_1, u_2, \ldots, u_n)$ is the Fielder vector, then split nodes according to a value s
  - bisection: s is the median value in u
  - ratio cut: s is the value that minimizes α
  - sign: separate positive and negative values ($s=0$)
  - gap: separate according to the largest gap in the values of u
- This works well (provably for special cases)

Fielder Value

- The value λ_2 is a good approximation of the graph expansion

$$\frac{\alpha(G)^2}{2d} \leq \lambda_2 \leq 2\alpha(G)$$

d = maximum degree

$$\frac{\lambda_2}{2} \leq \alpha(G) \leq \sqrt{\lambda_2(2d - \lambda_2)}$$

- For the minimum ratio cut of the Fielder vector we have that

$$\frac{\alpha^2}{2d} \leq \lambda_2 \leq 2\alpha(G)$$

- If the max degree d is bounded we obtain a good approximation of the minimum expansion cut

Conductance

- The expansion does not capture the inter-cluster similarity well
  - The nodes with high degree are more important
- Graph Conductance

$$\varphi(G) = \min_{U} \frac{E(U, V - U)}{\min\{d(U), d(V - U)\}}$$

  - $d(U) = \sum_{i \in U} \sum_{j \in U} A[i, j]$ weighted degrees of nodes in U

Conductance and random walks

- Consider the normalized stochastic matrix $M = D^{-1}A$
- The conductance of the Markov Chain M is

$$\varphi(M) = \min_U \frac{\sum_{i \in U} \sum_{j \notin U} \pi(i) M[i,j]}{\min\{\pi(U), \pi(V-U)\}}$$

  - the probability that the random walk escapes set U

- The conductance of the graph is the same as that of the Markov Chain, $\varphi(A) = \varphi(M)$

- Conductance φ is related to the second eigenvalue of the matrix M

$$\frac{\varphi^2}{8} \le 1 - \mu_2 \le \varphi$$

Interpretation of conductance

- Low conductance means that there is some bottleneck in the graph
  - a subset of nodes not well connected with the rest of the graph.
- High conductance means that the graph is well connected

Clustering Conductance

- The conductance of a clustering is defined as the minimum conductance over all clusters in the clustering.
- Maximizing conductance of clustering seems like a natural choice
- ...but it does not handle well outliers

A clustering bi-criterion

- Maximize the conductance, but at the same time minimize the inter-cluster (between clusters) edges

- A clustering $C = \{C_1, C_2, \ldots, C_n\}$ is a (c,e)-clustering if
  - The conductance of each C_i is at least c
  - The total number of inter-cluster edges is at most a fraction e of the total edges

The clustering problem

- Problem 1: Given c, find a (c,e)-clustering that minimizes e

- Problem 2: Given e, find a (c,e)-clustering that maximizes c

- Both problems are NP-hard

A spectral algorithm

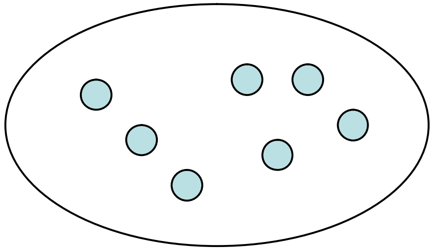
- Create matrix $M = D^{-1}A$
- Find the second largest eigenvector v
- Find the best ratio-cut (minimum conductance cut) with respect to v
- Recurse on the pieces induced by the cut.

- The algorithm has provable guarantees

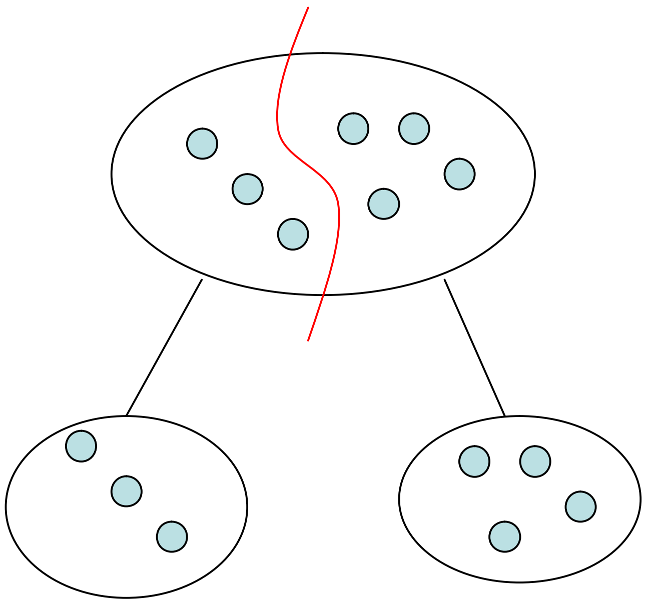
A divide and merge methodology

- Divide phase:
  - Recursively partition the input into two pieces until singletons are produced
  - output: a tree hierarchy
- Merge phase:
  - use dynamic programming to merge the leafs in order to produce a tree-respecting flat clustering

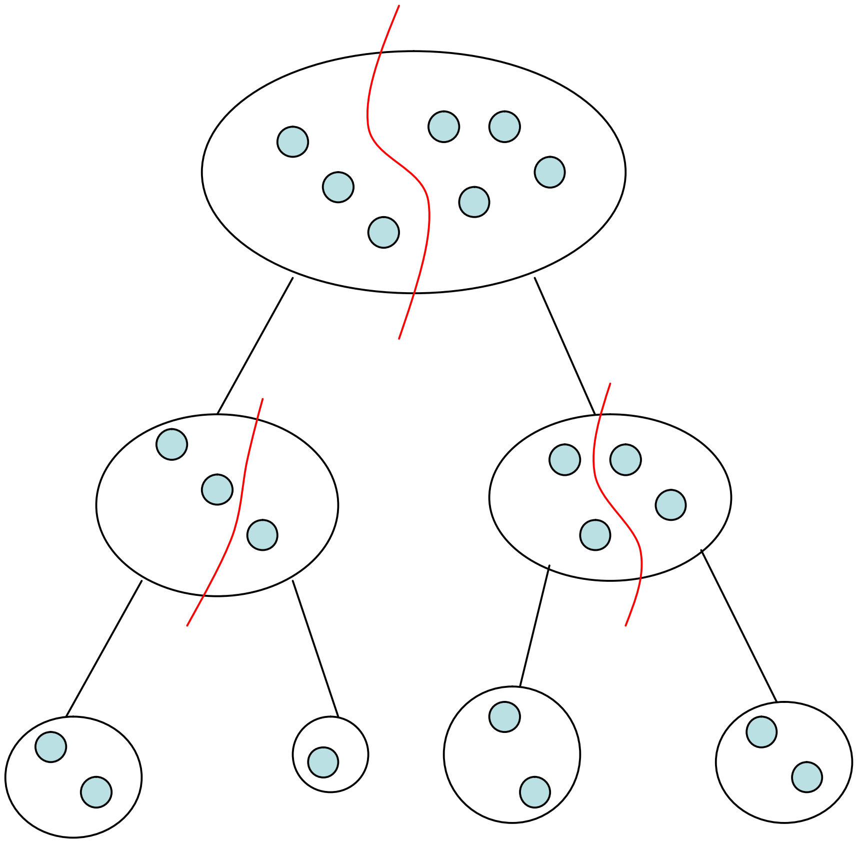
An example



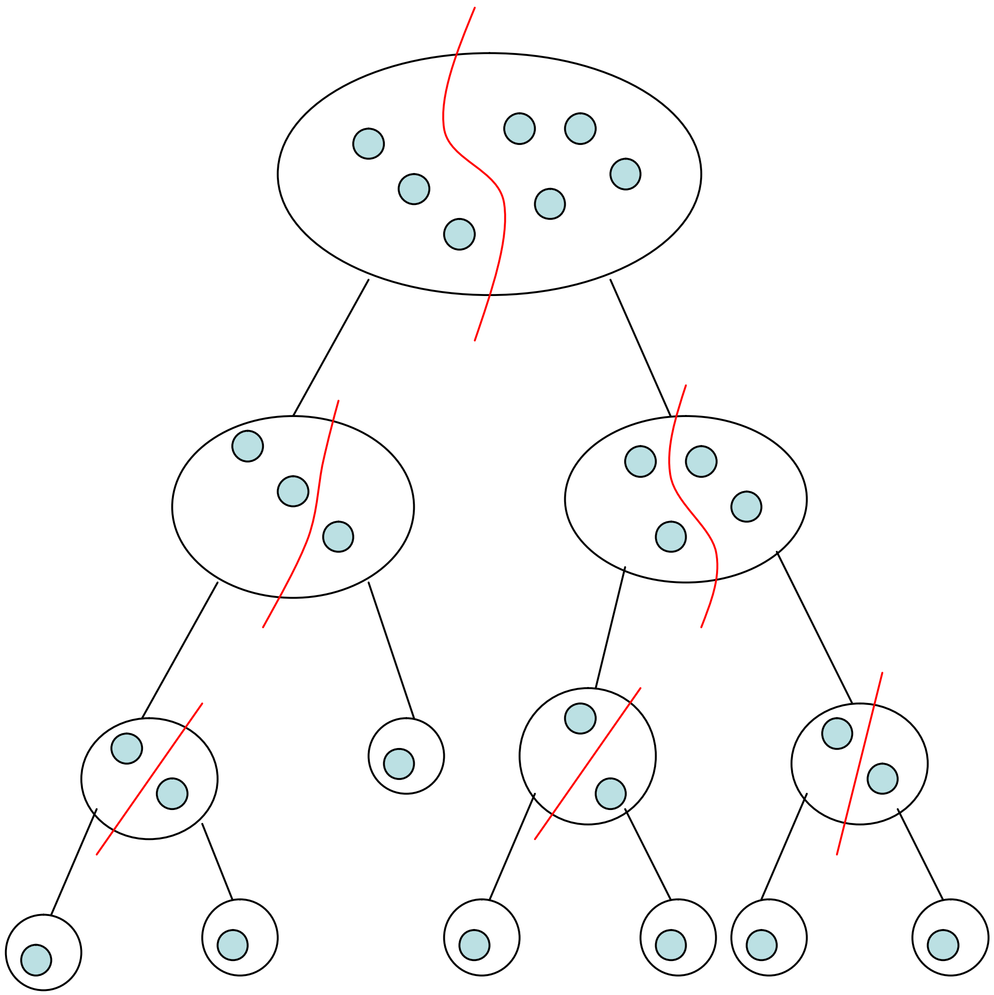
An example



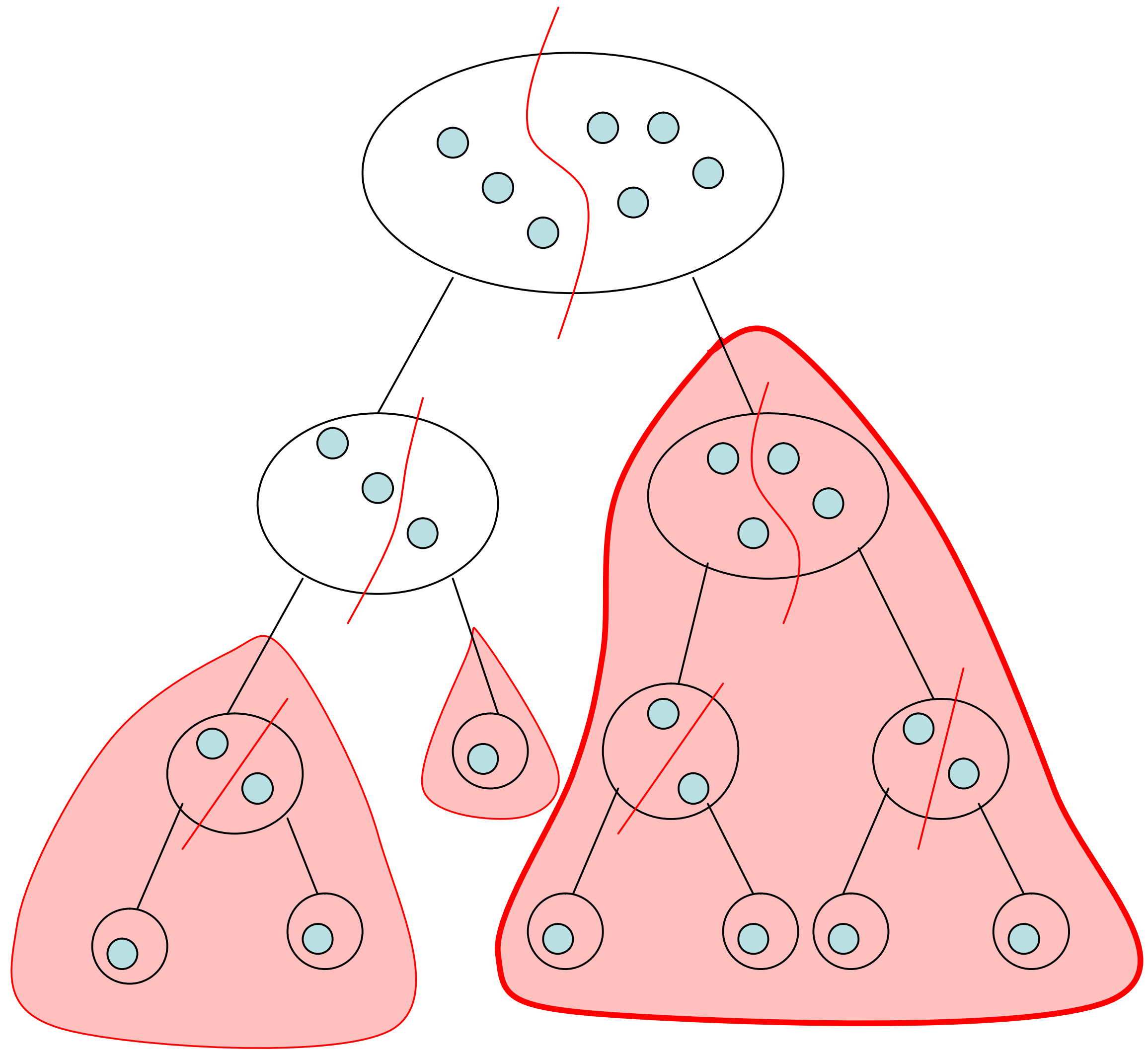
An example



An example



An example



Details

- The divide phase
  - use the spectral algorithm described before

- The merge phase
  - pick an optimization criterion
    - e.g. k-means

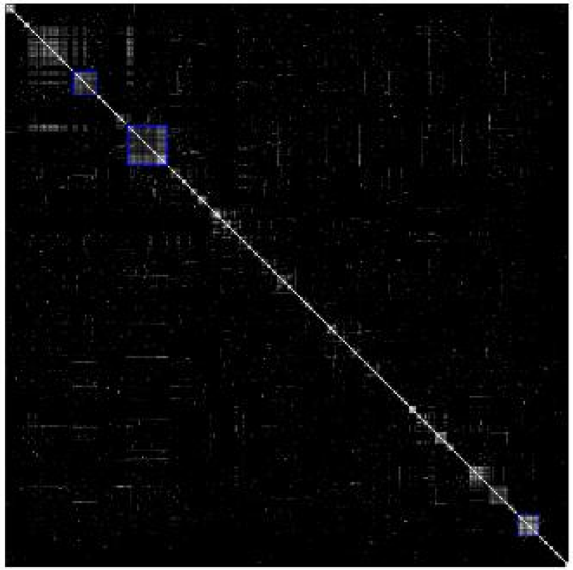
$$g(\{C_1, \ldots, C_k\}) = \sum_i \sum_{u \in C_i} d(u, p_i)^2.$$

  - perform dynamic programming

$$\mathsf{OPT}(C, i) = \begin{cases} C & \text{when } i = 1 \\ \operatorname{argmin}_{1 \le j < i} g(\mathsf{OPT}(C_l, j) \cup \mathsf{OPT}(C_r, i - j)) & \text{otherwise} \end{cases}$$

Applications to web search

- http://eigencluster.csail.mit.edu



(e) query:trees

Discovering communities

- Community: a set of nodes S, where the number of edges within the community is larger than the number of edges outside of the community.

Min-cut Max-flow

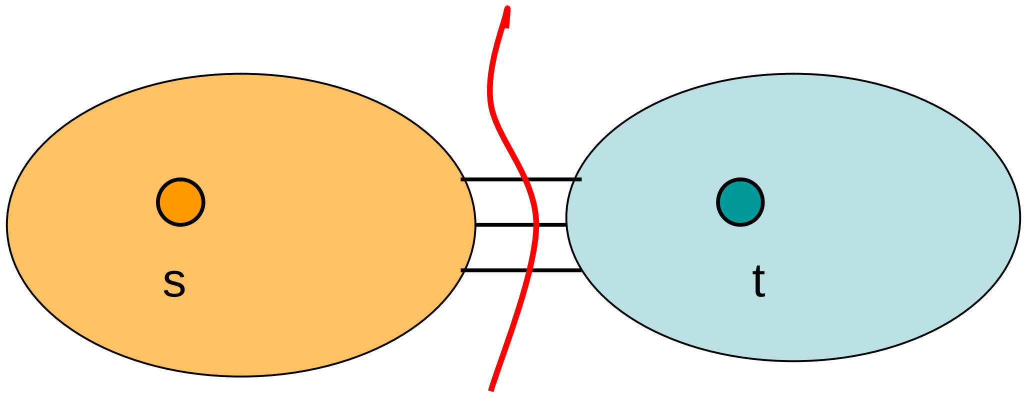
- Given a graph $G=(V,E)$, where each edge has some capacity $c(u,v)$, a source node s, and a destination node t, find the maximum amount of flow that can be sent from s to t, without violating the capacity constraints

- The max-flow is equal to the min-cut in the graph (weighted min-cut)

- Solvable in polynomial time

A seeded community

- The community of node s with respect to node t, is the set of nodes reachable from s in the min-cut that contains s
  - this set defines a community



Discovering Web communities

- Start with a set of seed nodes S
- Add a virtual source s
- Find neighbors a few links away
- Create a virtual sink t
- Find the community of s with respect to t

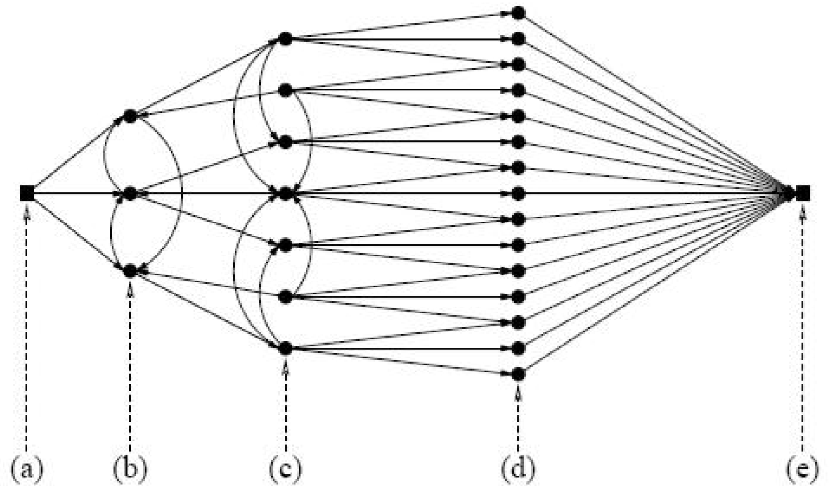


Figure 2: Focused community crawling and the graph induced: (a) The virtual source vertex; (b) vertices of seed web sites; (c) vertices of web sites one link away from any seed site; (d) references to sites not in (b) or (c); and (e) the virtual sink vertex.

A more structured approach

- Add a virtual source t in the graph, and connect all nodes to t, with edges of capacity α
- Let S be the community of node s with respect to t. For every partition P,Q of S we have

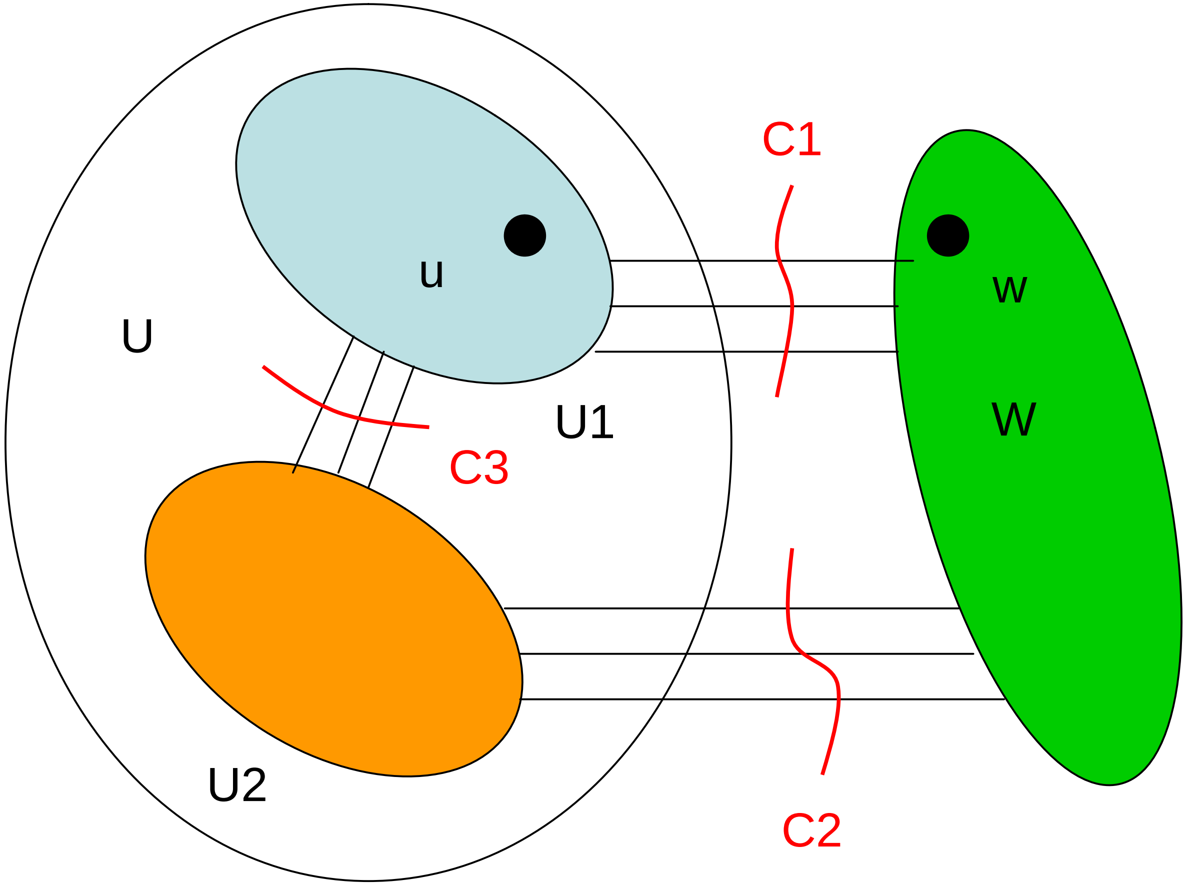
$$\frac{c(S, V-S)}{|V-S|} \leq \alpha \leq \frac{c(P, Q)}{\min\{|P|, |Q|\}}$$

- Surprisingly, this simple algorithm gives guarantees for the expansion and the inter-community density

Min-Cut Trees

- Given a graph G=(V,E), the min-cut tree T for graph G is defined as a tree over the set of vertices V, where
  - the edges are weighted
  - the min-cut between nodes u and v is the smallest weight among the edges in the path from u to v.
  - removing this edge from T gives the same partition as removing the min-cut in G

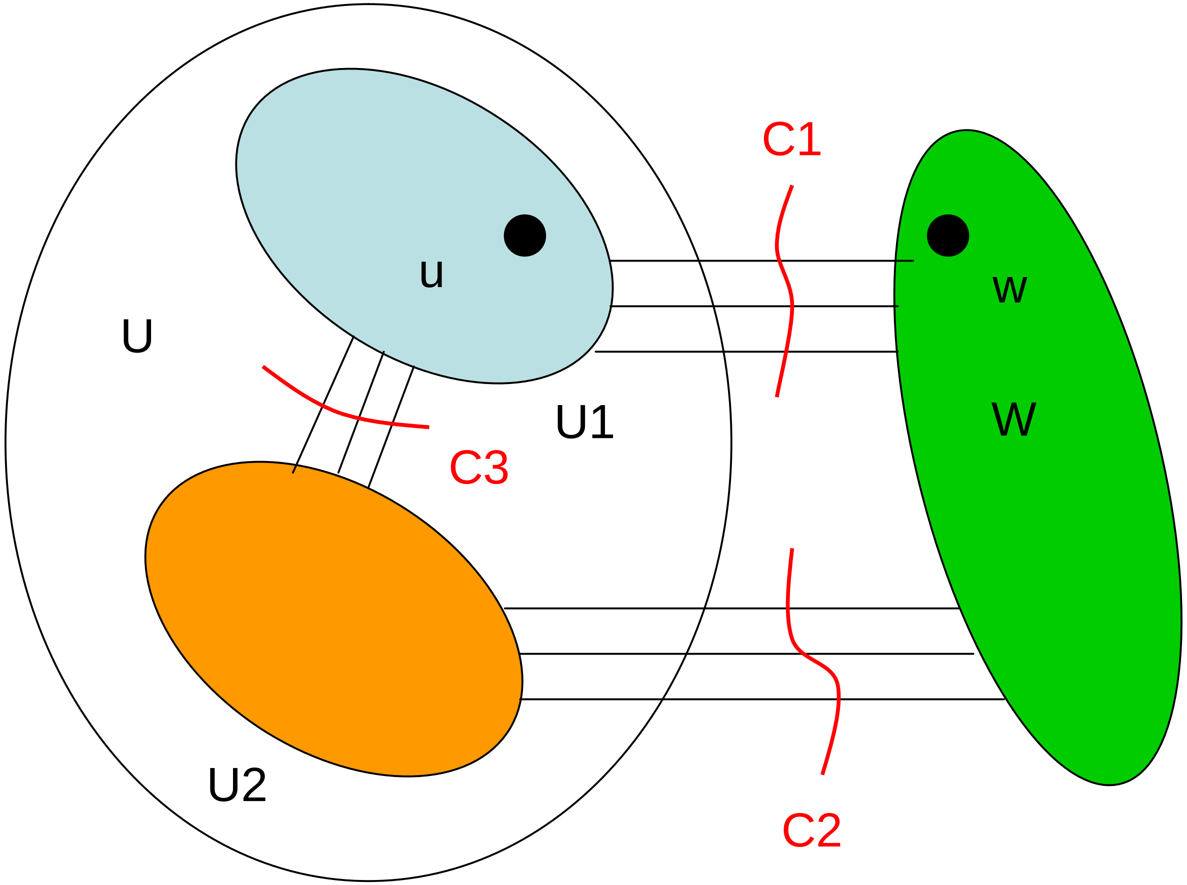
Lemma 1



c(W,U2) ≤ c(U1,U2)

C2 ≤ C3

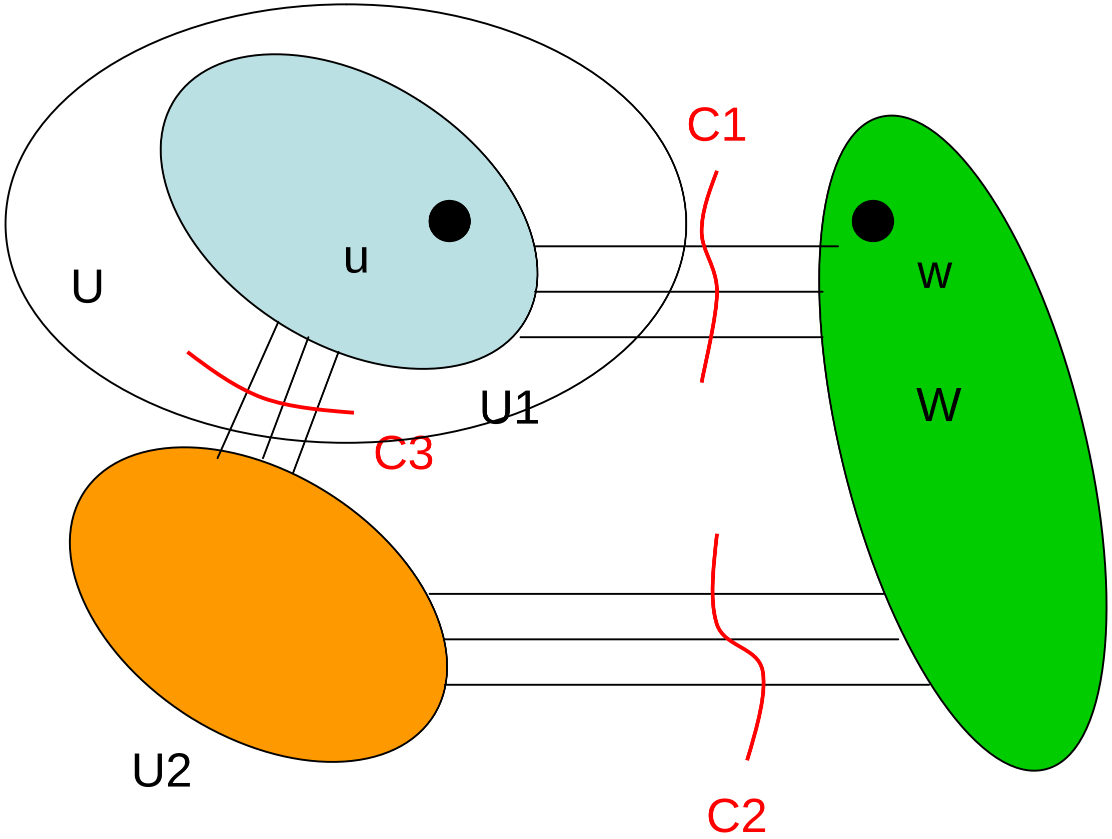
Lemma 1



c(W,U) =C1+C2

if C2 > C3 then
C1 + C3 < C1 +C2

Lemma 1



c(W,U) =C1+C2

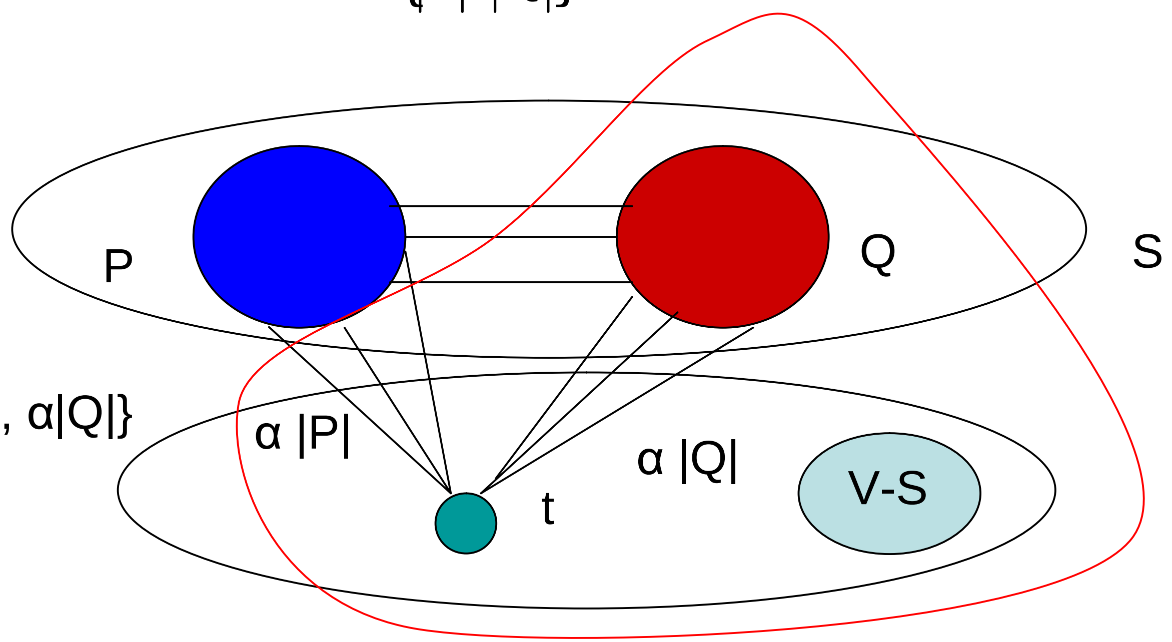
if C2 > C3 then
C1 + C3 < C1 +C2

this would be a
better cut: contradiction!

Lemma 2

- Let S be the community of the node s with respect to the artificial sink t. For any partition P,Q of S we have

$$\alpha \leq \frac{c(P,Q)}{\min\{|P|,|Q|\}}$$



if c(P,Q) < min{α |P|, α|Q|} then we would split differently

Lemma 3

- Let S be the community of node s with respect to t. Then we have

$$\frac{c(S, V-S)}{|V-S|} \leq \alpha$$

- Follows from Lemma 1:
  - $W = S$
  - $U_2 = V\text{-}S$
  - $U_1 = \{t\}$

Algorithm for finding communities

- § Add a virtual sink t to the graph G and connect all nodes with capacity α à graph G’
- § Create the min-cut tree T’ of graph G’
- § Remove t from T’
- § Return the disconnected components as clusters

Effect of α

- When α is too small, the algorithm returns a single cluster (the easy thing to do is to remove the sink t)
- When α is too large, the algorithm returns singletons (the tree is a star with t in the middle)
- In between is the interesting area.
- We can explore for the right value of α
- We can run the algorithm hierarchically
  - start with small α and increase it gradually
  - the clusters returned are nested

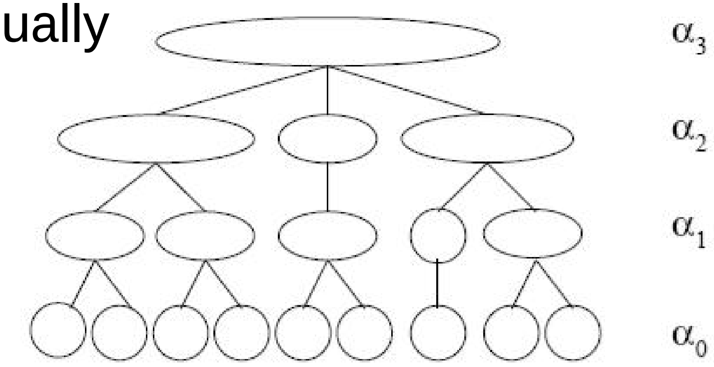


Figure 4: Hierarchical tree of clusters.

Some experiments

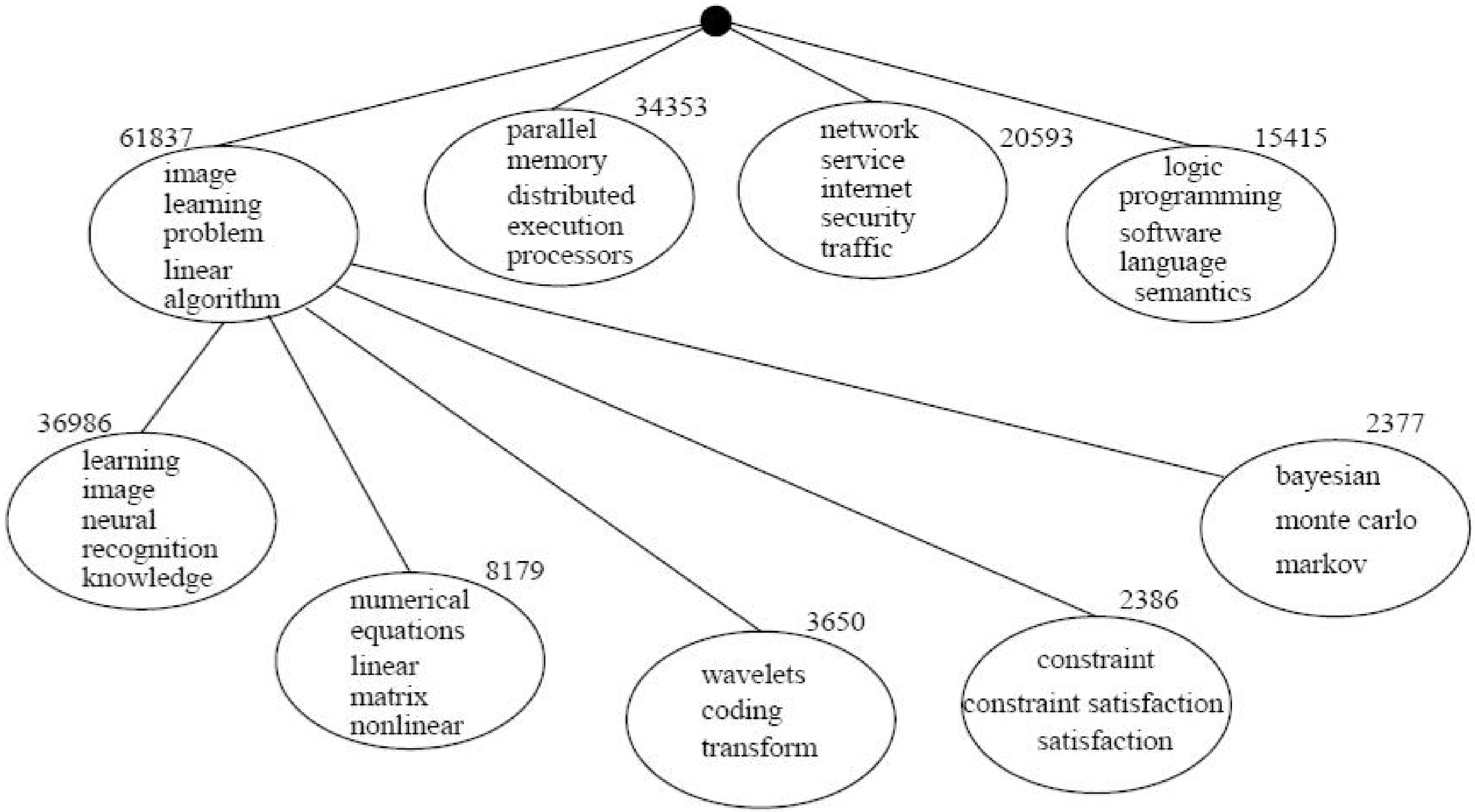


Figure 5: Top level clusters of CiteSeer. The sizes of each cluster are shown, as well as the top features for each cluster.

References

- J. Kleinberg. Lecture notes on spectral clustering
- Daniel A. Spielman and Shang-Hua Teng. Spectral Partitioning Works: Planar graphs and finite element meshes. Proceedings of the 37th Annual IEEE Conference on Foundations of Computer Science, 1996. and UC Berkeley Technical Report number UCB CSD-96-898.
- Ravi Kannan, Santos Vempala, Adrian Vetta, On clusterings: good, bad and spectral. Journal of the ACM (JACM) 51(3), 497--515, 2004.
- D. Cheng, R. Kannan, S. Vempala, G. Wang, A divide and merge methodology for clustering, PODS 2004.
- Gary Flake, Steve Lawrence, C. Lee Giles, Efficient identification of Web Communities, SIGKDD 2000
- G.W. Flake, K. Tsioutsiouliklis, R.E. Tarjan, Graph Clustering Techniques based on Minimum Cut Trees, Internet Mathematics, Volume 1, Issue 4, 2004