Clustering

Rose Catherine K.
Roll no: 07305010

M. Tech. Project Stage I

under the guidance of

Prof. S. Sudarshan

Computer Science and Engineering
Indian Institute of Technology Bombay

Introduction

- Keyword searching -important paradigm of searching.
- External memory BANKS could perform better if the nodes that are connected to each other are retrieved together.
- **Cluster**: a collection of objects that are similar to each other

- **Clustering**: process of finding out clusters in the given set of objects
- **Community**: set of real-world entities that form a closely knit group

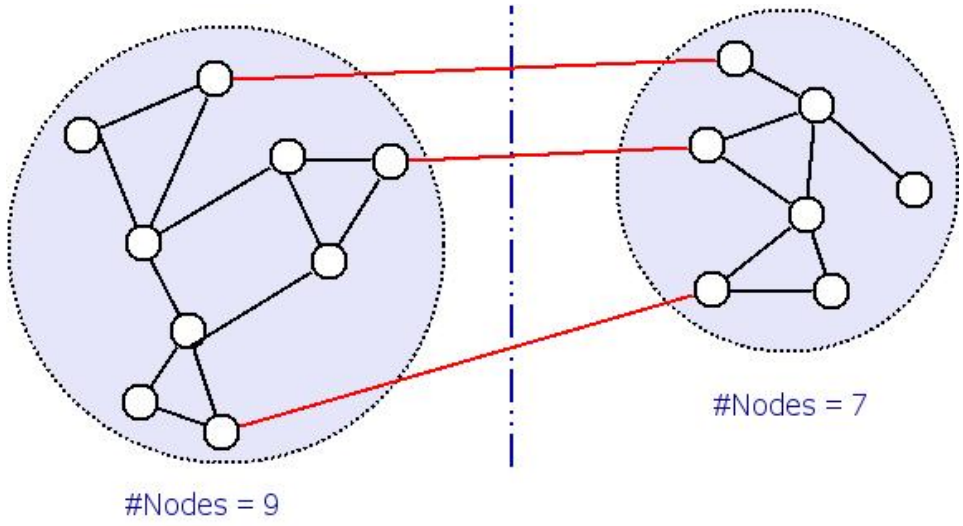


Fig. 1

Aspects of Clustering

Objective Function:
Cut Set Size:

- Objective: minimize cut set size

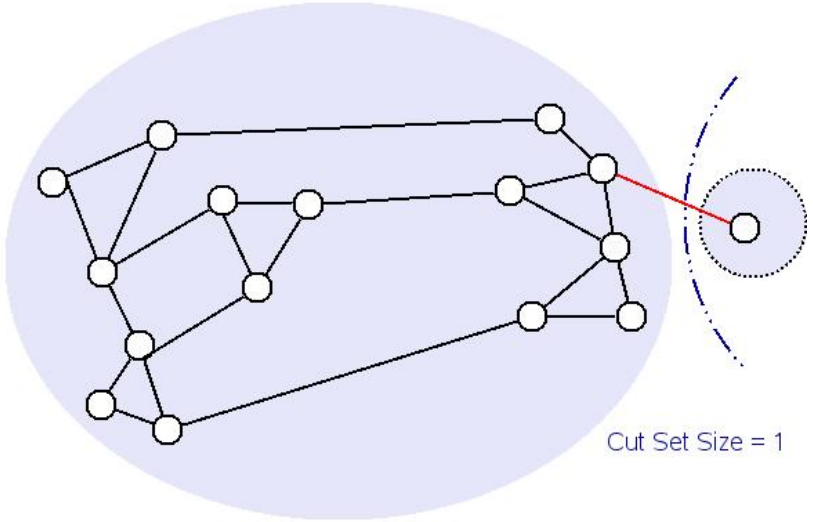


Fig. 2

- Constraint on k or size of clusters.
- No intuition for having a fixed number or size of communities.

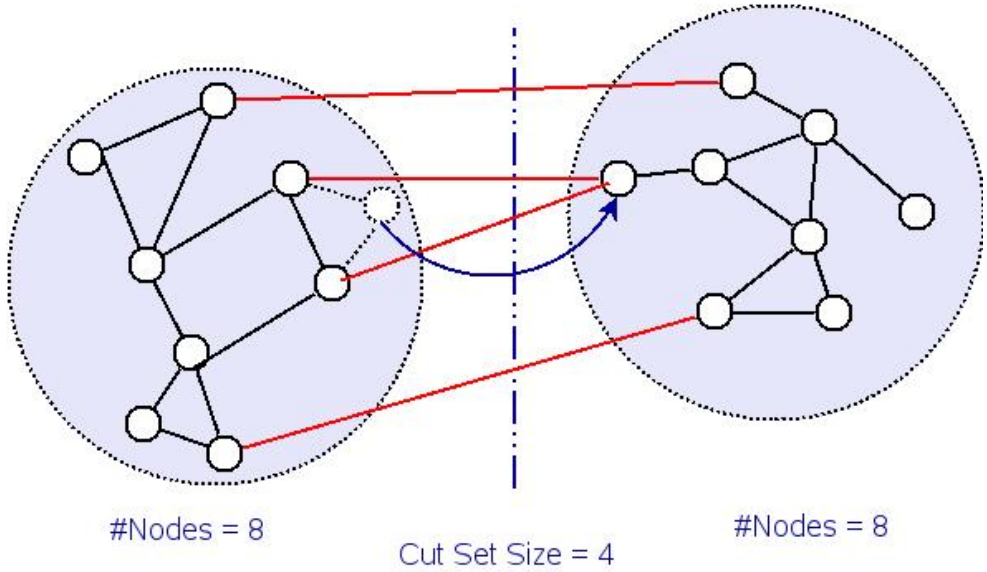


Fig. 3

Community Related Measures:

Graph Conductance

Objective: Minimize Conductance
For $S \subseteq V$:

- $Vol(S) = \sum_{v \in S} d(v)$
- $\partial(S)$: Cut Set

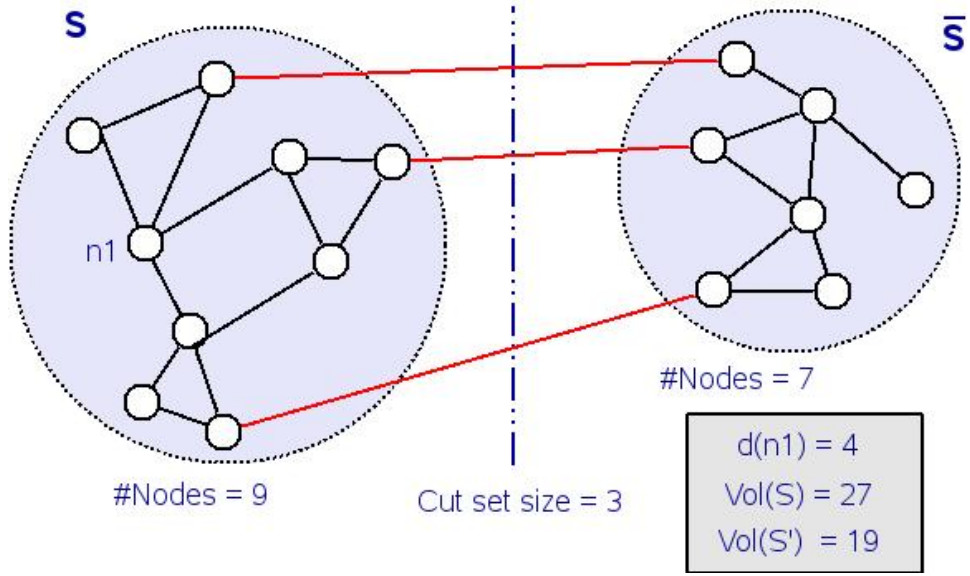


Fig. 4

$$\Phi(S) = \frac{|\partial(S)|}{min(Vol(S), Vol(\bar{S}))}$$

Example: Conductance for Fig. 4 $= \frac{3}{19} = 0.15$, for Fig. 3 $= \frac{4}{22} = 0.18$

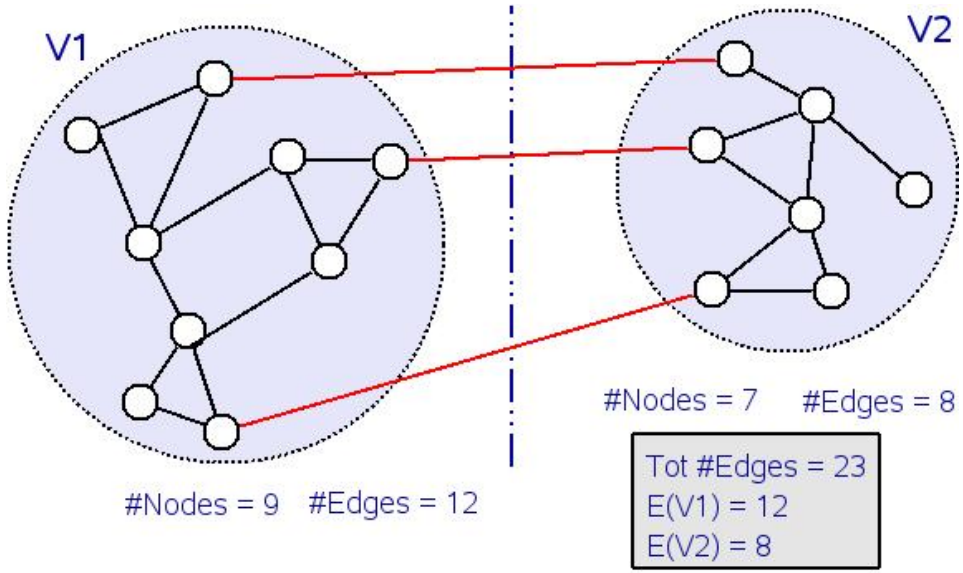
Modularity

Objective: Maximize Modularity

Clusters: V_1, ... V_k (k is not input parameter).

- $E(V_i)$: edges with both endpoints in V_i
- $Ex(V_i, \mathcal{G})$: expected number of such edges in a random graph from a given $\mathcal{G}$, with a vertex set V_i.

Fig. 5



$$Q = \frac{1}{m} \sum_{i=1}^{k} (|E(V_i)| - Ex(V_i, \mathcal{G}))$$

Example: (with Erdös-Rényi Random Graph Model), Q for Fig.5 $= \frac{9.075}{23}$, for Fig. 3 $= \frac{8.267}{23}$

Methods:

- Hierarchical Clustering Methods: Agglomerative and Divisive
- Partition Refinement Method: Given n objects and a number k:
  - Create an initial partitioning of the data into k clusters
  - Improve the partitioning by moving objects from one group to another
- Random Walk based Techniques:
  - Find the cluster to which a particular node belongs, or
  - Find the enclosing cluster for the given seed set

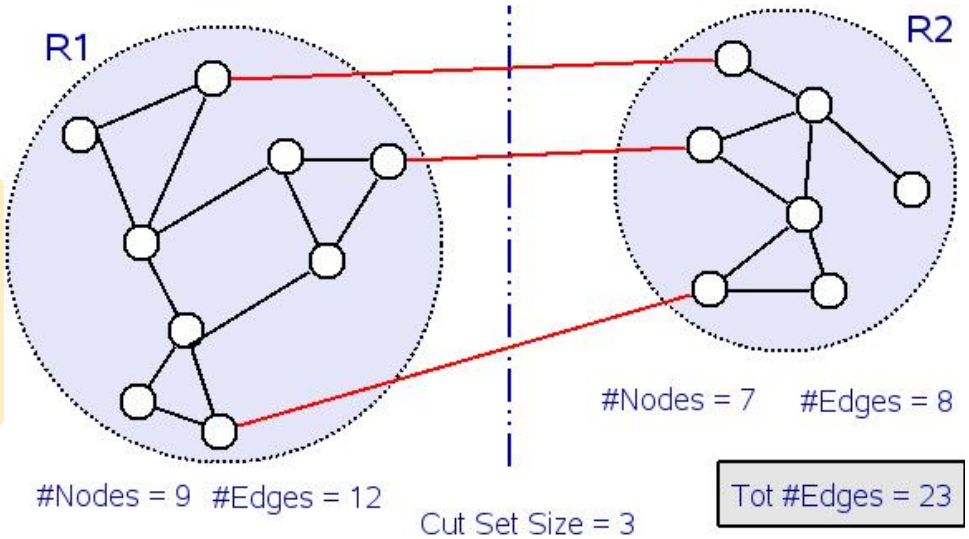
Clustering for Finding Communities - Partition Refinement Methods

Extremal Optimization

- Objective: Maximize Modularity

$$Q = \sum_r (e_{rr} - a_r^2)$$

Fig. 6

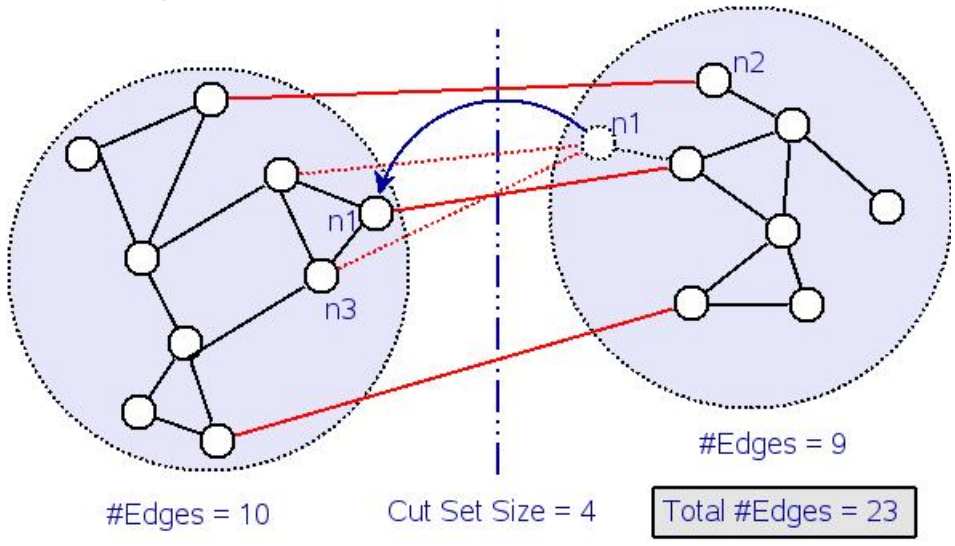


where, e_{rr} = fraction of edges with both end points in cluster r,
a_r = fraction of edges with atleast one end point in cluster r.
Example:(for Fig.6) $e_{R_1R_1} = \frac{12}{23}$, $a_{R_1} = \frac{15}{23}$, $Q = 0.22$
for Fig.3, $Q = 0.14$

Extremal Optimization

Idea: To optimize a global variable (Q) by improving extremal local variables (contribution of individual nodes)

Fig. 7



fitness of a node i

$$\lambda_i = \frac{\kappa_{r(i)}}{k_i} - a_{r(i)}$$

$\kappa_{r(i)}$ = number of links that i has to nodes in its community
k_i = degree of i
$a_{r(i)}$ = fraction of links with atleast one endpoint in the community of i.

Move the node with Least *fitness*, to the other partition

Example: (Fig. 7) $\kappa_{r(n_1)} = 1$, $k_{n_1} = 3$, $a_{r(n_1)} = \frac{13}{23}$, $\lambda_{n_1} = -0.23$, $\lambda_{n_2} = -0.07$, $\lambda_{n_3} = 0.06$.

Extremal Optimization - Algorithm

1. Create two random partitions of equal size
2. Calculate *fitness* of all nodes.
3. *Self Organization step:*
   1. Rank the nodes according to increasing value of *fitness*
   2. Choose a node of rank q, with $P(q) \propto q^{-\tau}$ for some $\tau > 0$
   3. Move the chosen node to other partition
4. Recalculate *fitness* values of nodes.
5. Repeat from step (iii) until an "optimal state" with a maximum value of Q is reached.
6. Delete all links between both partitions and recurse on each of the resulting components.
7. The algorithm stops when the value of modularity cannot be further improved.

Complexity: $O(N^2 ln^2(N))$

Advantages:

- Final result is independent of initialization, due to probabilistic selection method

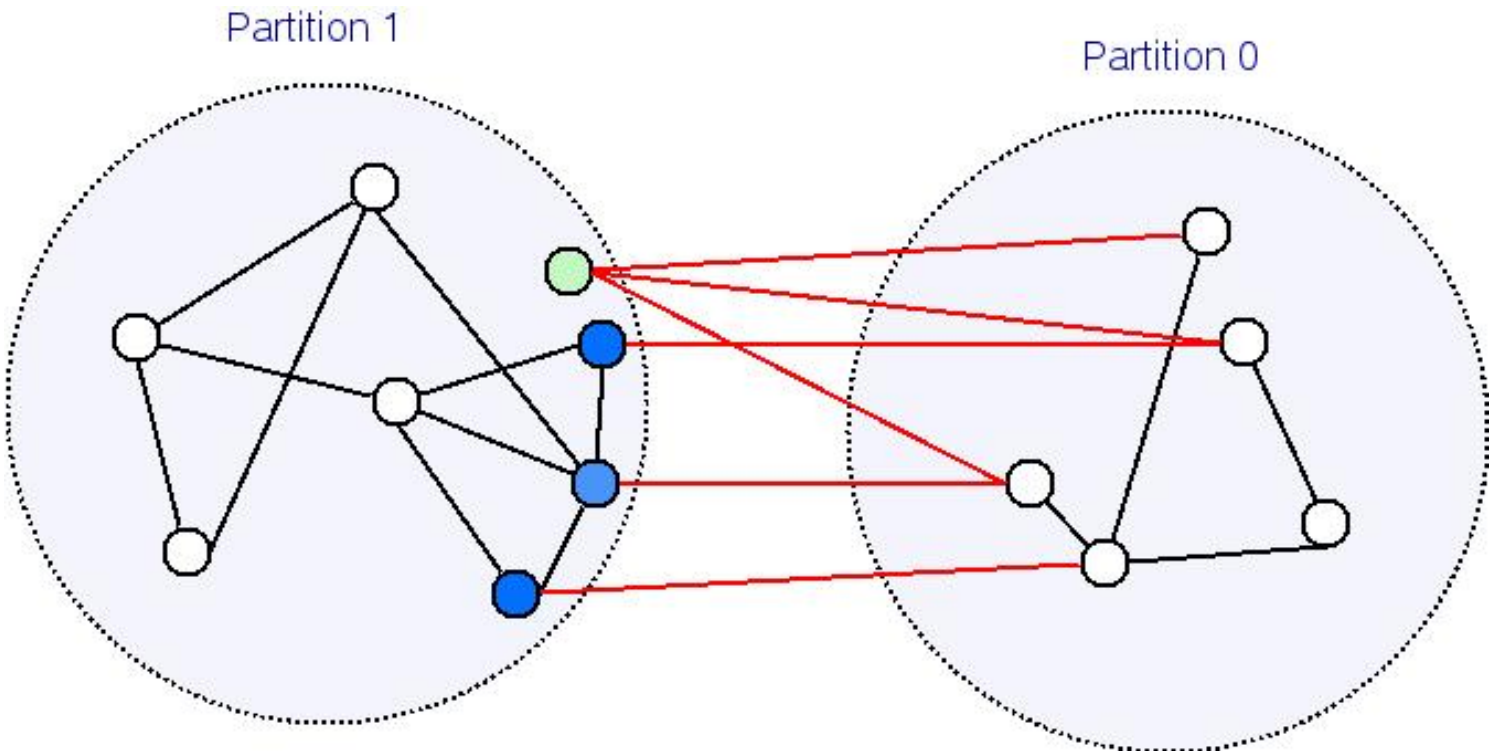
Limitations:

- All nodes must be examined in every round, for selecting the next node.

Improvement: Modularity-Weight Prioritised BFS algorithm by Prasang Upadhyaya.

Modularity-Weight Prioritised BFS

- Modification of Extremal Optimization algorithm



Algorithm:

1. Initially, all nodes are in `partition 1`; initialize `fringeNodeVector` with lowest degree vertex; set `maxIters` $= n$
2. Move the `top` element of `fringeNodeVector` to `partition 0`; Add neighbors of the `top` element, which are in `partition 1`, to `fringeNodeVector`
3. Choose a fixed number of elements, randomly, from `fringeNodeVector`. From these, move all nodes whose every neighbor is in the other partition, to `partition 0`.
4. For other nodes, calculate the increase in modularity on moving it to `partition 0`. Make that node with the highest increase, the `top` element of the `fringeNodeVector`.
5. Repeat steps from (2) onwards.
6. If highest increase in modularity obtained in step (4) is less than zero, then, this iteration is called as an *iteration-without-improvement*. If the number of such iterations exceed a particular number, then, stop iterating.

7. Once the iteration is stopped, then, undo all node exchanges done after the one which gave the maximum value of modularity over all the iterations, which gives two partitions. Recursively call `Modularity-Weight Prioritised BFS` on these two partitions.
8. If the size of a partition goes below a minimum value, it is not partitioned further.

Time Complexity: $O((n+m)\ ln(k))$
n = number of nodes; m = number of edges; k = number of clusters approximately desired ($k \sim \frac{2n}{size}$, where *size* is the user-specified lower bound on the cluster size).

Advantages:

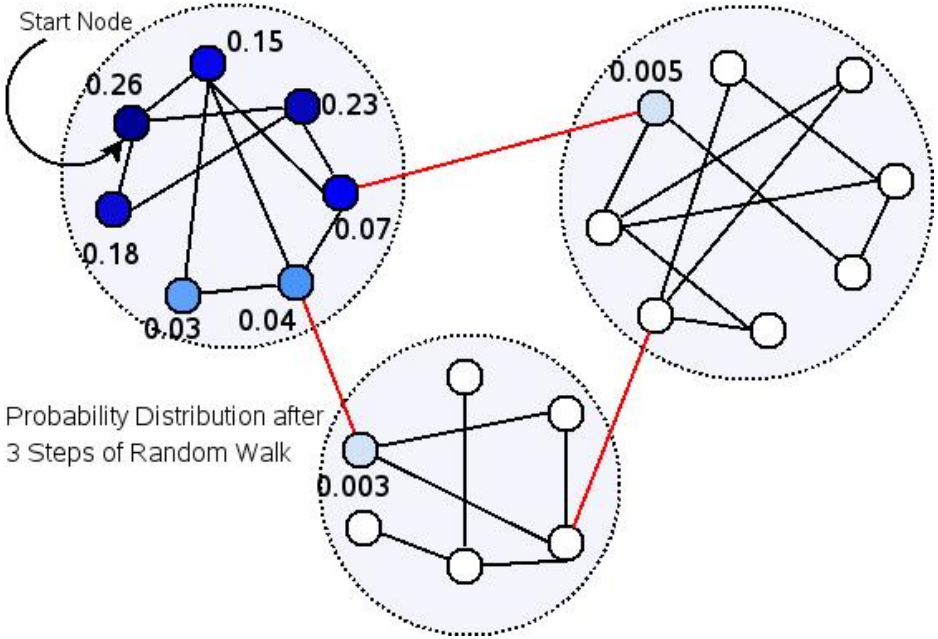
- The time complexity of $O(n\ ln(k))$ is one of the best achieved upper bounds.

Limitations:

- Probability of getting stuck at a local optimum is higher
- The solution obtained could be an approximate one.

Random Walk based Methods

- Objective: find the cluster to which a particular node belongs, or the enclosing cluster of a seed set.
- Start random walk from the specified node.
- Intuition: walk will remain within the cluster, with a large probability.
- Truncated walk: if probability is below a certain threshold, set to 0
- Computation will be more local.

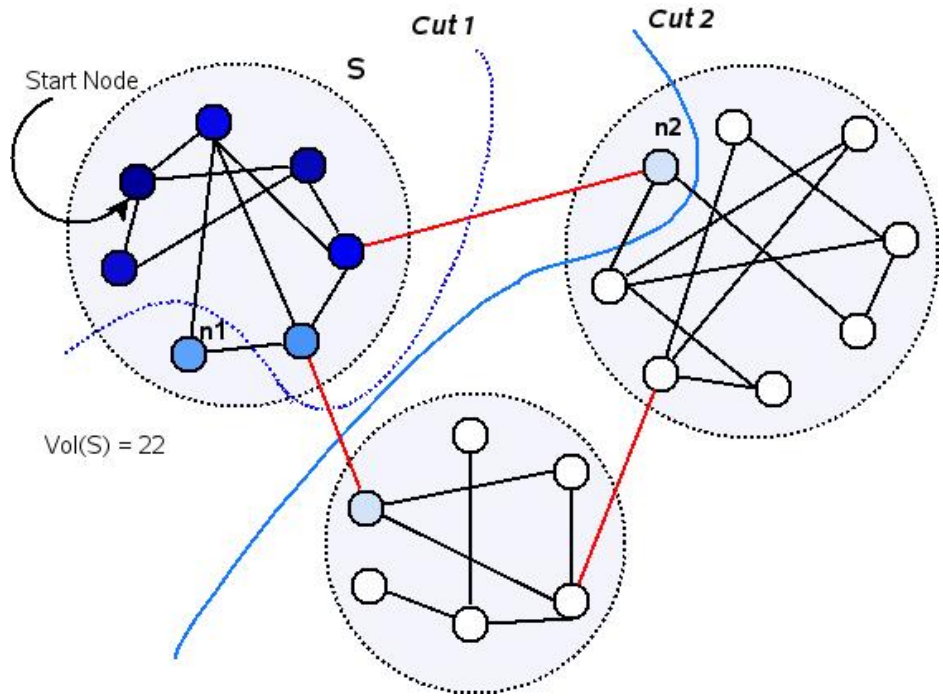


- Sudden drop in probability, outside the cluster boundary

$\Phi(S) = \frac{2}{22} = 0.09$

Cut 1: $\Phi(S - n_1) = \frac{4}{22-2} = 0.2$

Cut 2: $\Phi(S + n_2) = \frac{3}{22+3} = 0.12$



- Dip in conductance at cluster boundary

Clustering using Nibble Algorithm

Objective: find the cluster to which seed node belongs

Nibble Algorithm:

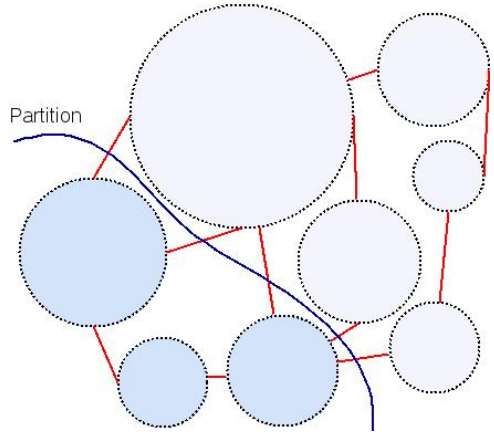
input: Start Vertex v, Graph G, Conductance θ_0, a positive integer b

1. Compute t_0 ($\propto ln(m)/\theta_0^2$), γ ($\propto \theta_0/ln(m)$), ϵ_b ($\propto \theta_0/ln(m)t_0 2^b$)
2. Start a lazy random walk from v
3. At each step: (until t_0)
   - Do the Truncation Operation with threshold $= \epsilon_b$
   - Sort the nodes in the decreasing order of their probabilities
   - Check if a $\tilde{j}$ exists such that:
     - $\Phi(\{1, ..., \tilde{j}\}) \leq \theta_0$
     - $Pr(\tilde{j}) \geq \gamma/Vol(\{1, ..., \tilde{j}\})$
     - $Vol(\{1, ..., \tilde{j}\}) \leq \frac{5}{6} Vol(V)$, then, output $C = \{1, ..., \tilde{j}\}$
4. Do the next step of random walk and repeat from Step (3)

Random Nibble Algorithm:

input: G, θ_0

1. Set v to be the largest degree vertex of G
2. Choose b in $1, ..., \lceil log(m) \rceil$ according to
$$Pr[b = i] \propto 2^{-i}$$
3. Call $\texttt{Nibble}(G, v, \theta_0, b)$



Partition Algorithm:

input: G, θ_0, $p \in (0, 1)$

1. Compute number of iterations j $(\propto m\lceil lg(1/p) \rceil$
2. Start with the entire graph, i.e., set W to V
3. Call $\texttt{RandomNibble}(G(W), \theta_0)$
4. Add the cluster nodes returned by `RandomNibble` to the answer
5. Now, remove these nodes from W
6. If $Vol(W) \leq \frac{5}{6} Vol(V)$, then stop
7. Else, repeat from Step (3)

Multiway Partition Algorithm:

input: G, θ, p

1. Set θ_0 to $(5/36)\theta$
2. Compute number of iterations t ($\propto (lg\ m)^2$)
3. Start with the entire vertex set, i.e, set $\mathcal{C}_1$ to V
4. In each step: For each component $C \in \mathcal{C}_t$,
   Call `Partition`$(G(C), \theta_0, p/m)$
5. Add the two partitions returned to $\mathcal{C}_{t+1}$ and repeat from Step 4
6. Final clustering is given by $\mathcal{C}_{t+1}$

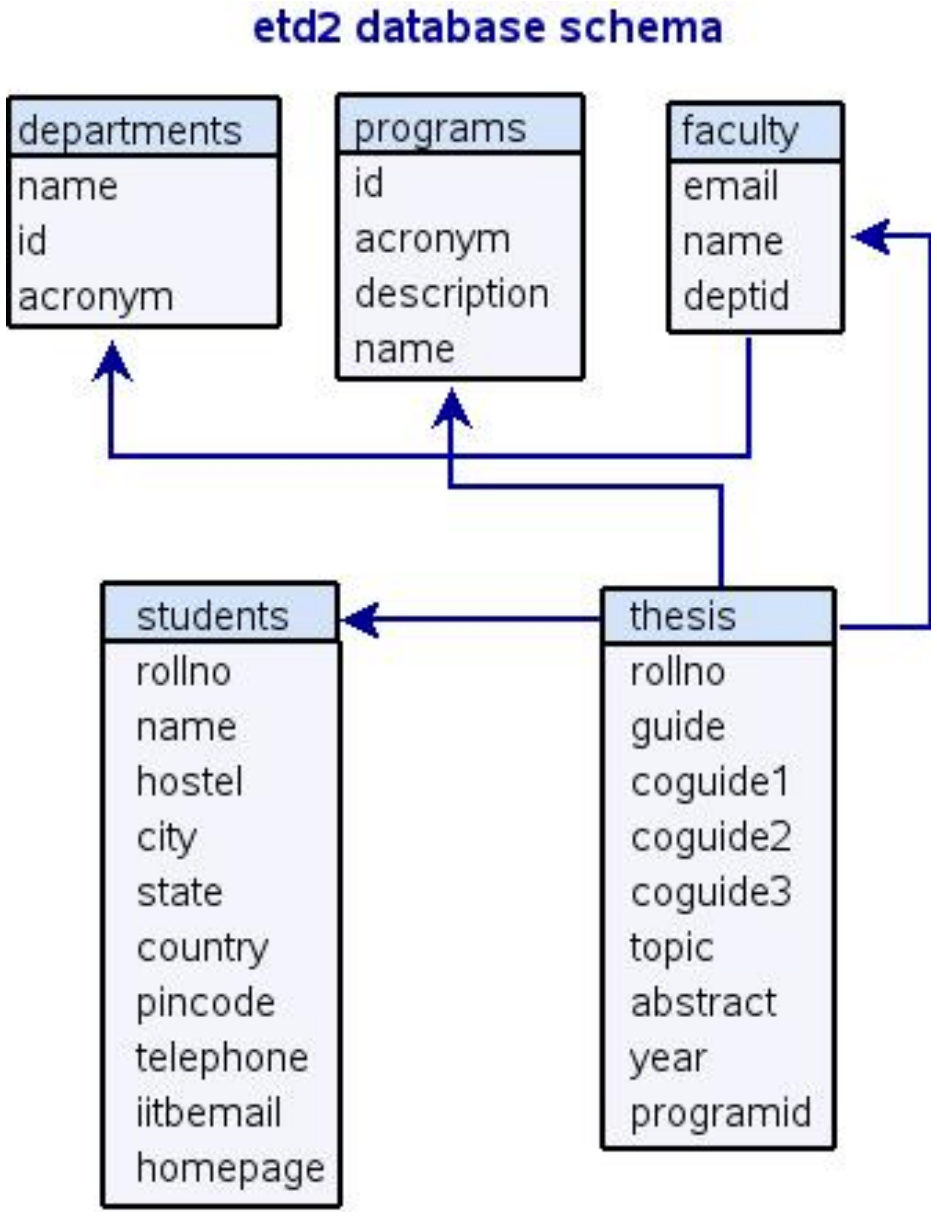
Running Time:

`Nibble` : $O(2^b ln^4(m)/\theta_0^5$

`Multiway Partition` : $O(m\ (lg(1/p)\ lg^{O(1)}(m))/\theta^5)$

Experiments and Analysis

- `Nibble` was implemented in Java
- Executed on `etd2` database
- 4329 nodes and 21432 edges



Observations:

- Nodes grouped together were related to the start node.
- Sizes of the clusters differ drastically - 2189, 226, 62 and 39
- Effect of `Start Node`: clustering on `program` basis - `MTech., PhD., MDes., MPhil.` etc. which were the starting nodes.
- Effect of `conductance`: conductance seems to be affecting the cluster size inversely.
- Effect of `maxClusterSize`:
  - The clustering on the basis of `department` - `School of Management`, `Humanities and Social Science Department`.
  - The effect of `start node` is no more visible and it itself was not added to both of the clusters.
  - Varying the value of `maxClusterSize` did not affect the clustering.

Conclusions:

- The first implementation of `Nibble` not very successful in finding the more intuitive clustering - based on `Department`.
- The modified version of `Nibble` with the bound `maxClusterSize` could find clusters on the basis of `Department`, but unable to find all the clusters.
- Seed vertex not in the cluster

Conclusions and Future Work I

- Clustering: technique of finding the underlying structure of a graph.
- Different aspects: objective functions, methods.
- Studied few objectives and clustering techniques.
- `Nibble` - community finding method based on random walks, implemented and executed on the `etd2` database
- identified few shortcomings

Conclusions and Future Work II

Proposed direction of future work:

- Improve upon the `Nibble` algorithm: new method for choosing the seed vertex, finding all the clusters and handling the case of the seed not being included in the cluster.
- When some information is known in addition to just the graph structure, the clustering algorithm must do better. e.g. the basis of clustering is intuitive and known beforehand
- Given a small graph and a mapping of the nodes of a larger graph to nodes in the former, find a good clustering for the latter, by finding a good clustering for the smaller graph.
- Obtain high node and edge compression for the supernode graph.
- Formulate an objective of clustering, which reduces the query answer time of BANKS system
- Test Modularity-Weight Prioritised BFS Algorithm.

Extra Slides

Clustering using Seed Sets I

Objective: discover the enclosing community of a given cohesive "seed set" of nodes, that has small conductance

- Extended `Nibble` algorithm for a set of start nodes
- Examines only a small portion of the entire graph
- Intuition: for a random walk that begins from the seed nodes, much of the walk will be contained in the cluster. As soon as we move outside the cluster, the probability will fall, revealing the cluster boundary.

Define:
The initial probability distribution p_0 is set to ψ_S:

$$\psi_S = \begin{cases} d(x)/Vol(S) & \text{if } x \in S \\ 0 & \text{otherwise} \end{cases}$$

Random walk transition matrix, M as $1/2(I + AD^{-1})$

Clustering using Seed Sets II

Algorithm:

1. Simulate the next step of the random walk to obtain the probability distribution, $p_{t+1} = Mp_t$.
2. Sort the vertices in descending order of their degree-normalized probabilities: $r_t(v) = p_t(v)/d(v)$.
3. For the truncated walk, set the probability on any vertex for which $r_t(v) \leq \epsilon$ to 0, where ϵ is a constant, called the threshold.
4. Let v_i^t be the i^{th} vertex after sorting, such that $r(v_i^t) \geq r(v_{i+1}^t)$. Then, this ordering defines a collection of sets $S_0^t, ..., S_J^t$, where $S_j^t = \{v_i^t | 1 \leq i \leq j\}$, and J is the number of vertices with nonzero values of $p(u)/d(u)$.
5. Each of the sets S_j^t, are tested for a good community.
6. If none of the sets qualify as a good community, then the random walk is continued, from step 1 onwards.

Clustering using Seed Sets III

Good seed sets:

- Seed set S for a community C which has a small conductance, if the amount of probability that has escaped from C after T steps, is not much larger than $\phi(C)\ T$.
- Any set that is fairly large and nearly contained in the target community.
- Sets chosen randomly from within a target community.

Advantages:

- Explores only local locality.
- Can find nested clusters that enclose the seed set.

Disadvantages: Selection of the seed set: identify the target cluster set initially, and choose nodes randomly from it, to form the seed set.

Clustering using Nibble Algorithm I

Definitions and Mathematical Notations:
Define:

$$bal(S) = \frac{Vol_V(S)}{Vol_V(V)}$$

For the subgraph of G induced by a subset of the vertices $W \subseteq V$, $(S \subseteq W)$ define:

$$Vol_W(S) = \sum_{v \in S} |w \in W : (v, w) \in E|$$

$$\partial_W(S) = \sum_{v \in S} |w \in W - S : (v, w) \in E|$$

$$\Phi_W(S) = \frac{|\partial_W(S)|}{min(Vol_W(S), Vol_W \bar{S})}$$

Clustering using Nibble Algorithm II

Also,

$$\chi_S(x) = \begin{cases} 1 & \text{for } x \in S \\ 0 & \text{otherwise} \end{cases}$$

$$\psi_S(x) = \begin{cases} d(x)/Vol_V(S) & \text{for } x \in S \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Walk Matrix, } P = (AD^{-1} + I)/2$$

where, A: unweighted graph, D: diagonal matrix with $(d(1), ..., d(n))$ on the diagonal
$p_t^v = P^t \chi_v$: Prob. Distrn with start vertex, v, after t steps
The truncation operation:

$$[p]_\epsilon(v) = \begin{cases} p(v) & \text{if } p(v) \geq 2\epsilon d(i) \\ 0 & \text{otherwise} \end{cases}$$

Nibble Algorithm

$C = \texttt{Nibble}(G, v, \theta_0, b)$
G a graph, v a vertex, $\theta_0 \in (0,1)$, b a positive integer.
(1) Set $\tilde{p_0}(x) = \chi_v$
(2) Set $t_0 = 49\ ln(me^4)/\theta_0^2$, $\gamma = \frac{5\theta_0}{7.7.8\ ln(me^4)}$, and $\epsilon_b = \frac{\theta_0}{7.8\ ln(me^4)t_0 2^b}$
(3) For $t = 1$ to t_0
    (a) Set $\tilde{p_t} = [P\ \tilde{p_{t-1}}]_{\epsilon_b}$
    (b) Compute a permutation $\tilde{\pi_t}$ such that
$\tilde{p_t}(\tilde{\pi_t}(i)) \geq \tilde{p_t}(\tilde{\pi_t}(i+1))$ for all i.
    (c) If there exists a $\tilde{j}$ such that
        (i) $\Phi(\tilde{\pi_t}(\{1, ..., \tilde{j}\}) \leq \theta_0$,
        (ii) $\tilde{p_t}(\tilde{\pi_t}(\tilde{j})) \geq \gamma / Vol_V(\tilde{\pi_t}(\{1, ..., \tilde{j}\})$, and
        (iii) $5\ Vol_V(V)/6 \geq Vol(\tilde{\pi_t}(\{1, ..., \tilde{j}\}) \geq (5/7)\ 2^{b-1}$
        then output $C = \tilde{\pi_t}(\{1, ..., \tilde{j}\}$ and quit.
(4) Return $\emptyset$.

Table: Pseudocode for Nibble algorithm

$C = \texttt{RandomNibble}(G, \theta_0)$
(1) Choose a vertex v according to ψ_V
(2) Choose a b in $1, ..., \lceil log(m) \rceil$ according to
$$Pr[b = i] = 2^{-i}/(1 - 2^{-\lceil log(m) \rceil})$$
(3) $C = \texttt{Nibble}(G, v, \theta_0, b)$

Table: Pseudocode for `Random Nibble` algorithm

$D = \texttt{Partition}(G, \theta_0, p)$
where G is a graph, $\theta_0, p \in (0, 1)$.
(0) Set $W_1 = V$
(1) For $j = 1$ to $56m \lceil lg(1/p) \rceil$
(a) Set $D_j = \texttt{RandomNibble}(G(W_j), \theta_0)$
(b) Set $W_{j+1} = W_j - D_j$
(c) If $Vol_{W_{j+1}}(W_{j+1}) \leq (5/6)\ Vol_V(V)$, then go to step (2)
(2) Set $D = V - W_{j+1}$

Table: Pseudocode for `Partition` algorithm

$\mathcal{C} =$ `MultiwayPartition`(G, θ, p)
(0) Set $\mathcal{C}_1 = V$ and $S = \emptyset$
(1) For $t = 1$ to $\lceil log_{17/16} m \rceil \cdot \lceil lg(m) \rceil \cdot \lceil lg(2/\epsilon) \rceil$
  (a) For each component $C \in \mathcal{C}_t$,
    $D =$ `Partition`$(G(C), \theta_0, p/m)$
    Add D and $C - D$ to $\mathcal{C}_{t+1}$
(2) Return $\mathcal{C} = C_{t+1}$

Table: Pseudocode for `Multiway Partition` algorithm

Running Time:
`Nibble` : $O(2^b ln^4(m)/\theta_0^5$
`RandomNibble` : $O(ln^4(m)/\theta_0^5$
`Partition` : $O(mlg(1/p)ln^4(m)/\theta_0^5$
`Multiway Partition` : $m\ (lg(1/p)\ lg^{O(1)}(m))/\theta^5$
If $\mathcal{C}$ is the set of components returned by `Multiway Partition`, with probability at least $(1 - p)$,
$cut - size(\mathcal{C}) \leq (\theta\ log_{17/16} m\ .\ lg\ m\ .\ lg(2/\epsilon))(m/2)$, where
$\epsilon = min(1/16, 1/(4\ \lceil lg\ m \rceil))$