Clustering the Graph Representation of Data

Rose Catherine K.
Roll no: 07305010

M. Tech. Project Stage 2

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**: set of nodes such that, connections within it is dense; inter-cluster edges are low.
- **Community**: set of real-world entities that form a closely knit group
- **Objective function**: distance-based measures, cut-size, community-related measures: modularity, conductance
- **Graph Conductance**:

For $S \subseteq V$:

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

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

Finding Communities using Random Walks on Graphs

Random walks:

- a graph traversal technique.
- Probability distribution of a walk: probability of a random walk of k steps, started at a particular *startNode*, to be at a particular node at the instant/step of inspection (*nodeProbability*).

Clustering using Random walks:

- Objective: find the cluster to which a particular node belongs, or the enclosing cluster of a seed set.
- Intuition:
  - Walk started from a node in the cluster will remain within it, with a large probability.
  - Probability distribution of the random walk gives a rough ranking of the nodes of the graph.
  - A good cluster can be obtained by considering the highest ranking nodes, and by using conductance to choose the best.

Clustering using Nibble Algorithm [ST04]

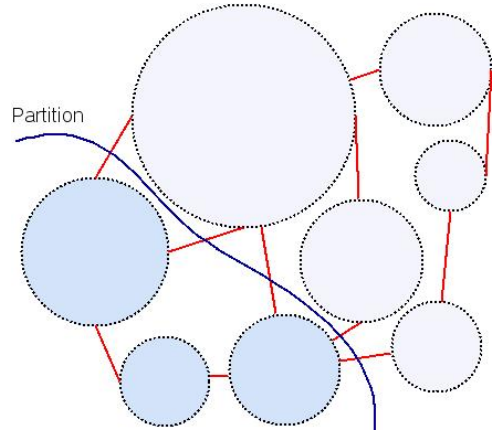
Objective: find the cluster to which the seed node belongs

Nibble Algorithm:
input: Start node v, Graph G, Max Conductance θ_0

1. Compute the bound on maxIterations, t_0, and threshold, ϵ.
2. Start spreading probabilities from v.
3. Truncate the walk by setting *nodeProbability* to 0 where it is $< \epsilon$
4. Sort the nodes in the decreasing order of their degree normalized probabilities.
5. Check if a j exists such that:
   - Conductance of the first j nodes $\leq \theta_0$
   - The above set of nodes satisfy predefined requirements on its volume.
6. If a j was found, then return the first j nodes of the sorted set.
7. Otherwise, do the next step of spreading probabilities and repeat from Step (3).

Random Nibble Algorithm:

1. Choose v randomly with probability proportional to its degree
2. Call `Nibble`(G, v, θ_0)



Partition Algorithm:

1. Compute the bound on maxIterations, t
2. Call `RandomNibble` with current graph and θ_0.
3. Keep the cluster returned by `RandomNibble` on hold. If there is already a cluster on hold, then merge them.
4. If on merging in step (3), the volume exceeds a predetermined fraction of G, then stop and return the merged cluster.
5. Else, remove these nodes from the graph and repeat.

Multiway Partition Algorithm:

- Call `Partition` recursively on the graph, for a predetermined number of iterations.

Shortcomings of the `Nibble` algorithm

- Specify the conductance of the clusters, apriori.
- May terminate at larger conductance, before finding the best.
- User cannot control the cluster size.
- No control over the spread of the walk.
- Couldn't find the intuitive clustering, based on `Department` on `etd2`.

Shortcoming of the overall algorithm

- Processes the entire graph in a top-down manner - difficult for large graphs.

Clustering using Seed Sets [AL06]

Objective: find the enclosing community of a "seed set" of nodes

Algorithm:

1. Assign equal probabilities to all nodes in the seed set, and start spreading probabilities.
2. Sort the vertices in descending order of their degree-normalized probabilities.
3. Truncate the walk for nodes with probabilites lesser than a predefined threshold.
4. Find a j such that the set of first j nodes, C, satisfy the test for a good community: the probability outside C is lesser than a predetermined fraction of $\Phi(C) \times \#numSteps$
5. If a j is found, stop and return that set as the community.
6. Else, continue the random walk from step (2) onwards.

Shortcoming:

The seed set is chosen manually.

Clustering using `Modified-Nibble` algorithm : Outline

Clustering Algorithm

1. Choose a start node.
2. Nibble out a cluster for the start node, and remove it from the graph.
3. Repeat from step (1), until the entire graph is processed.

- Proceed by removing one cluster at a time, rather than processing the entire graph at once.

`Modified-Nibble` Algorithm

1. Set the initial probability of the start node to 1 and start spreading probability from it, for a specific number of steps (batch).
2. Find the best cluster for the currently active nodes, using `FindBestCluster` algorithm.
3. If the cluster obtained has same or higher conductance than the best cluster of the previous iteration, make a greedy decision to stop, assuming that further processing may not give any better results.
4. Else, if the conductance has reduced, we assume that doing more walks may improve the results. Hence, continue spreading of probabilities from all the active nodes, again for a specific number of steps, and repeat from step (2).

- The conductance of clusters are not taken as input from the user.
- The algorithm finds the cluster of best conductance.

`FindBestCluster` Algorithm

1. Consider the nodes in the decreasing order of probabilities.
2. The candidate clusters C_i contain nodes from 1 to i of the sorted set, $i = 1, 2, ...$
3. Compute the conductance of all the candidate clusters.
4. Choose the one with smallest conductance as the best cluster.

- The algorithm always finds a cluster, unlike the `Nibble` algorithm, which will return a cluster only if it satisfies some specific requirements.

Sample execution of the `Modified-Nibble` algorithm

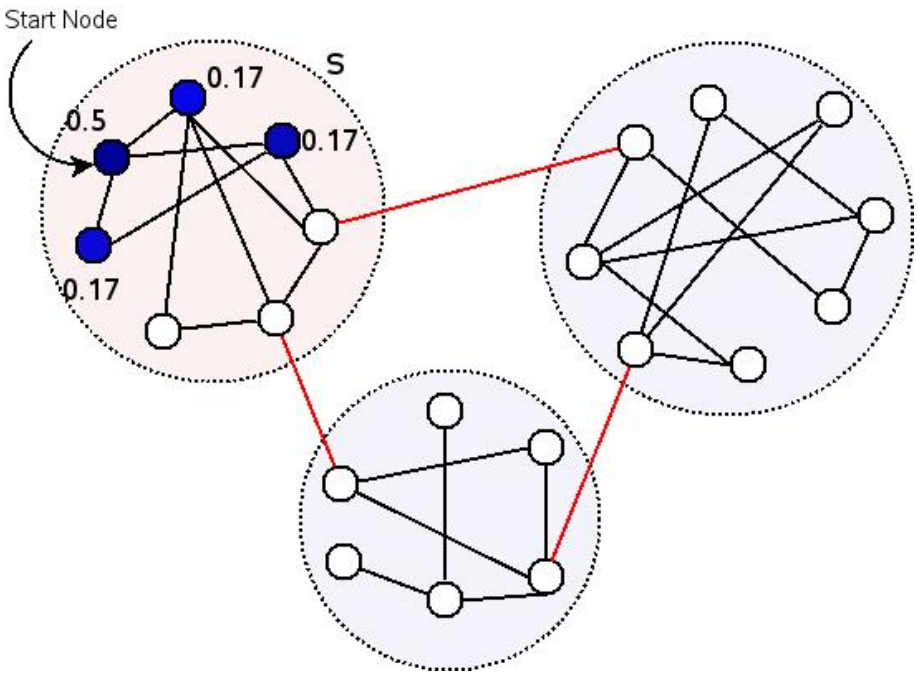


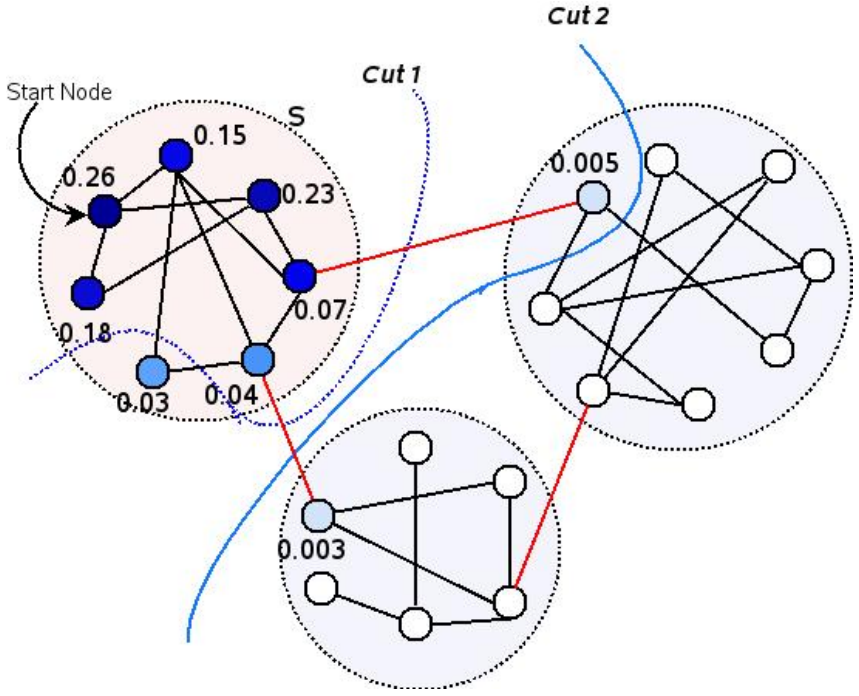
Fig: Prob. distrn. after 1 step

Batch 1

$\Phi(\textit{best cluster}) = \frac{4}{12} = 0.33$

Preferred cluster S, not found yet.

Fig: Prob. distrn. after 3 steps



Batch 2

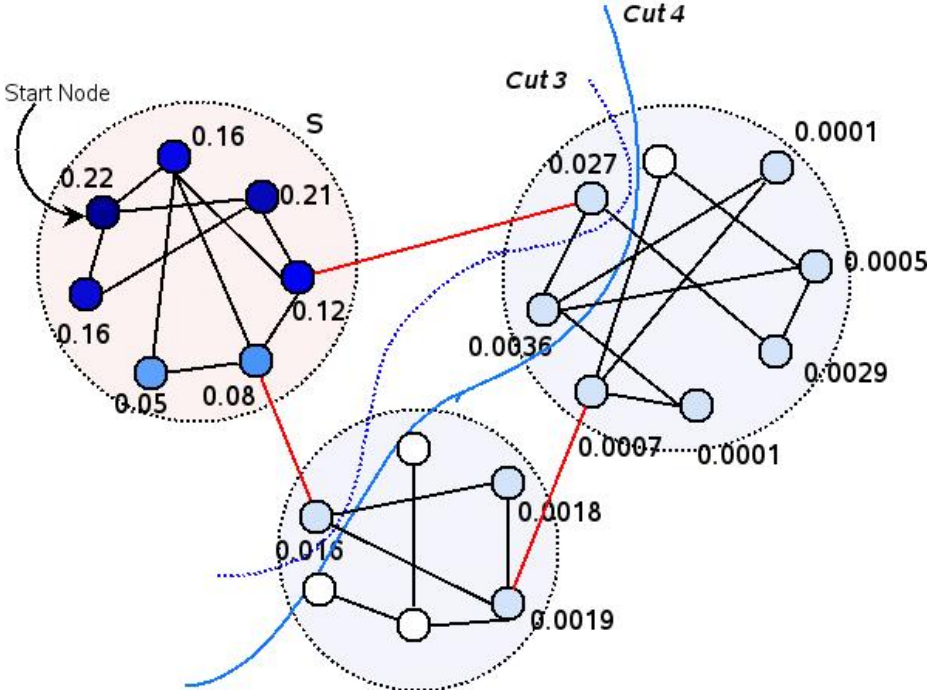
$\Phi(S) = \frac{2}{22} = 0.09$

$\Phi(Cut1) = \frac{4}{22-2} = 0.2$

$\Phi(Cut2) = \frac{3}{22+3} = 0.12$

Best Cluster $= S$

Fig: Prob. distrn. after 5 steps



Batch 3

$\Phi(Cut3) = \frac{4}{28} = 0.14$

$\Phi(Cut4) = \frac{6}{32} = 0.18$

Best Cluster $= S$

Terms

- `maxClusterSize`: Upper bounds number of nodes in a cluster.
- `maxActiveNodeBound`: Upper bounds the number of nodes that can be active, at any time, thus controlling the spread of the walk.

$$\texttt{maxActiveNodeBound} = \texttt{f} \times \texttt{maxClusterSize}$$

where `f` is called the 'factor'.

- Arithmetic plus Geometric Progression (APGP):
  - Sorting after each step of walk is costly.
  - Invoke `FindBestCluster` only at regular intervals.
  - Arithmetic Progression (AP) and Geometric Progression (GP) did not perform well.

$$t_i^{apgp} = (a + id) + (a\ r^i),\ i = 0, 1, 2, ...$$

Random walk considered:

- self-transition probability $= 0.5$.
- edge-weights are not considered while spreading probability.
- no truncation of probabilities.

`ModifiedNibble` procedure

1. Set the *nodeProbability* of s to 1, and that of all other nodes to 0. Set *totalWalkSteps* to 0.
2. For i^{th} iteration, get t_i in the APGP series; perform a batch of random walks for (t_i − *totalWalkSteps*) steps. But, if $t_i >$ *maxClusterSize*, the batch is of (*maxClusterSize* − *totalWalkSteps*) steps.
3. If at any time in step (2), the number of active nodes exceed *maxActiveNodeBound*, then directly proceed to step (4).
4. Invoke `FindBestCluster` to get the best cluster.
5. If conductance of the best cluster returned $\geq$ conductance of previous best cluster, return the latter.
6. Else if the conductance has decreased:
   - If $t_i \geq$ *maxClusterSize*, or, if the number of active nodes $=$ *maxActiveNodeBound*, stop and return the best out of the current and previous clusters.
   - Otherwise, set *totalWalkSteps* to t_i and repeat from step (2).

`FindBestCluster` procedure

1. Find the degree-normalized probabilities of all nodes.
2. Sort the nodes in the decreasing order of their degree-normalized probabilities.
3. Find a j s.t. the first j entries of the sorted set have the smallest conductance. $1 \leq j \leq min(\texttt{numActiveNodes}, \texttt{maxClusterSize})$.
4. Return the first j entries of the sorted set as the best cluster.

Heuristics

- Using APGP to invoke `FindBestCluster` procedure atmost `log(maxClusterSize)` times.
- `Compaction` procedure: merge clusters of small size

Different flavors to the algorithm

- Choosing `startNode`: largest degree vs smallest degree.
- Proceeding when `maxActiveNodeBound` is reached: continue the execution, but move only to neighbors which are already explored.
- Remove the hub nodes before starting the clustering procedure.

Advantages of clustering using `Modified-Nibble` algorithm

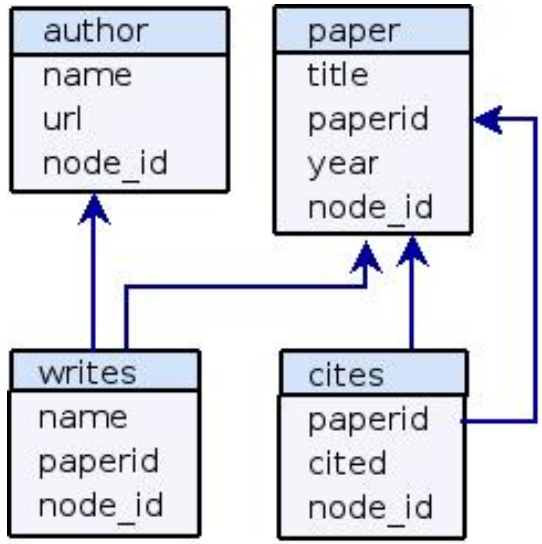
- Can find good communities by touching only a small neighborhood of the *startNode*, without exploring the entire graph.
- Doesn't require the user to input any bounds on the conductance of clusters. The algorithm figures out the best available cluster on its own.
- Space requirements to find a community for a particular node can be made independent of the graph size, by bounding the number of active nodes.
- Can proceed by removing one cluster at a time, rather than processing the entire graph at once.

Experiments and Analysis

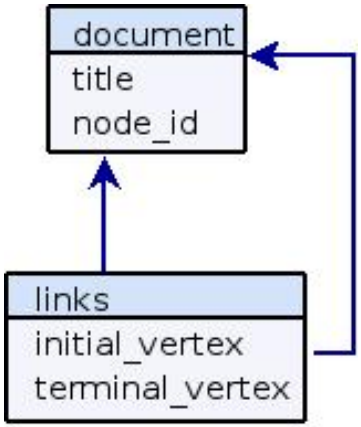
Data Sets

`Digital Bibliography Library Project (dblp)` (2003 version)

- Tables: `author`, `cites`, `paper`, `writes`
- Number of nodes: 1,771,381
- Number of edges: 2,124,938
- max degree = 784



`Wikipedia` (2008 version)



- Tables: `document`, `links`
- Number of nodes: 2,648,581
- Number of undirected edges: 39,864,569
- max degree = 267,884

Results on dblp3 for Flavor1

Flavor1: MaxDegree startNode, terminate on `maxActiveNodeBound`

Edge compression and avg conductance

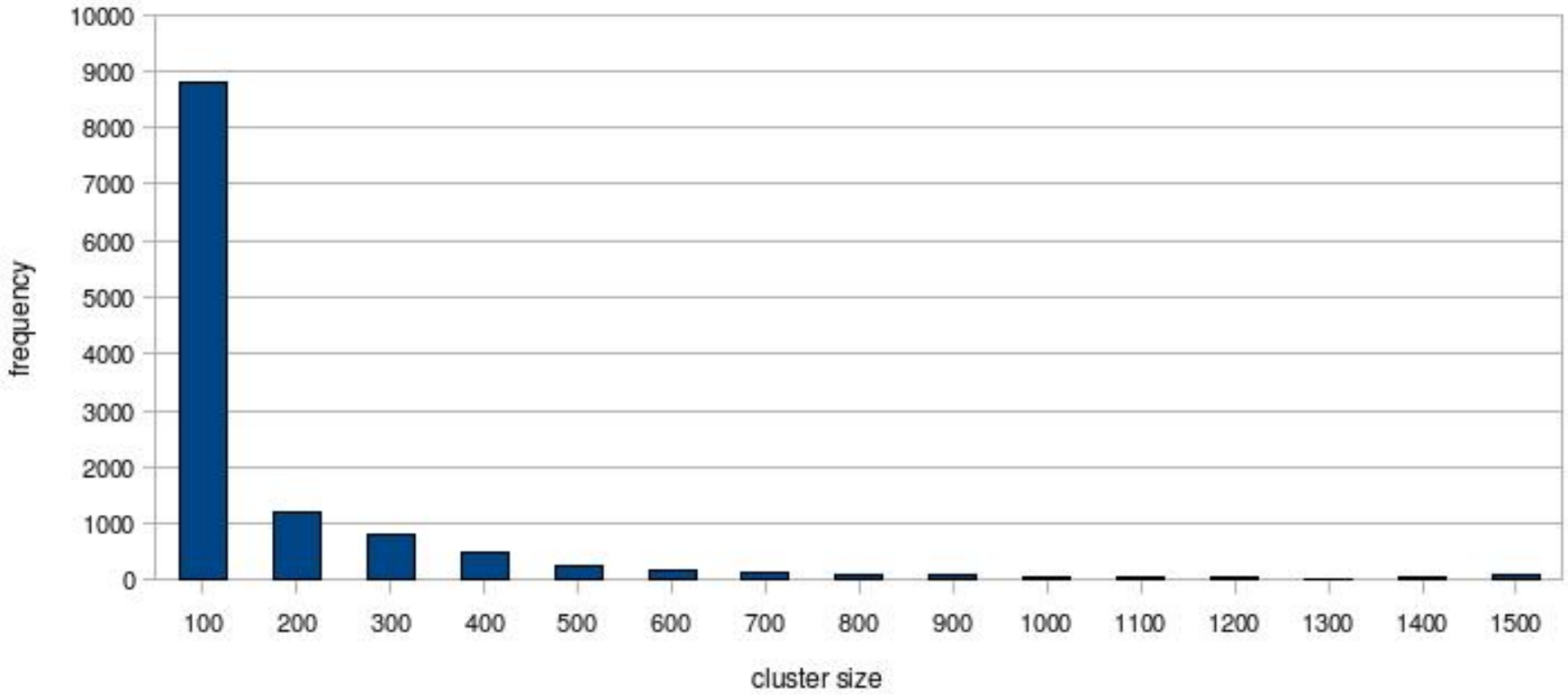
max Cluster Size	# clusters	# inter-cluster edges	edge compression	avg conductance
100	24,113	206,040	10.31	0.0838
200	12,698	166,219	12.78	0.0689
400	6,709	136,784	15.53	0.0579
800	3,505	114,536	18.55	0.0499
1500	1,909	90,574	23.46	0.0401

($a = 2$, $d = 7$, $r = 1.5$, $f = 500$ and with compaction)

With increasing `maxClusterSize`:

- Compression improves by 2 times and conductance becomes half.

Chart of cluster size vs. frequency of `dblp3`



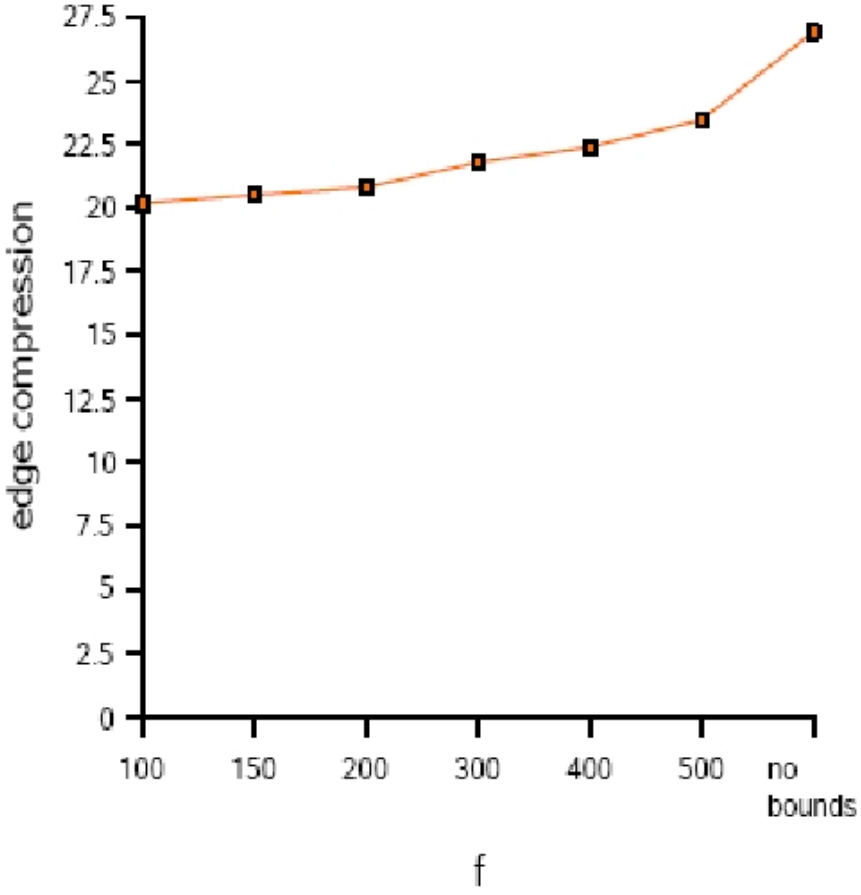
- Indicates that the inherent clusters of `dblp3`, are mostly of size 100 to 400.

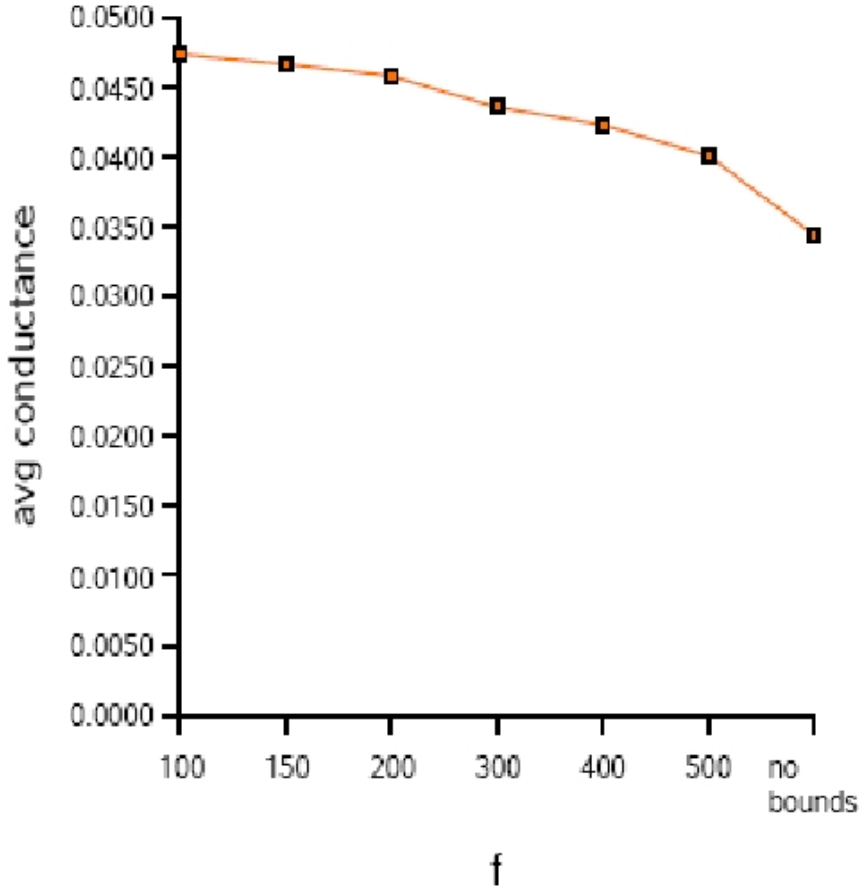
Effect of f on compression & conductance

f	# clusters	# inter-cluster edges	edge compression	avg conductance	Time (approximate)
100	1,965	105,290	20.18	0.0474	1.5 hrs
150	1,946	103,603	20.51	0.0467	2 hrs
200	1,945	102,080	20.82	0.0458	3 hrs
300	1,934	97,529	21.79	0.0436	9.5 hrs
400	1,921	94,872	22.39	0.0423	15 hrs
500	1,909	90,574	23.46	0.0401	1 day
no bounds	1,862	78,973	26.91	0.0344	2.5 days

(maxClusterSize = 1500, dataset = dblp3)

- Better compression and conductance for larger values of f.





- Values comparable for `f` ranging from 100 to 500.
- Lower values may suffice, also considering the cost.
- Sharp improvement for 'no bounds'.
- Terminating when `maxActiveNodeBound` is reached, may be affecting cluster quality.

Effect of compaction on compression

Different cluster sizes ($a = 2$, $d = 5$, $r = 1.5$, $f = 100$)

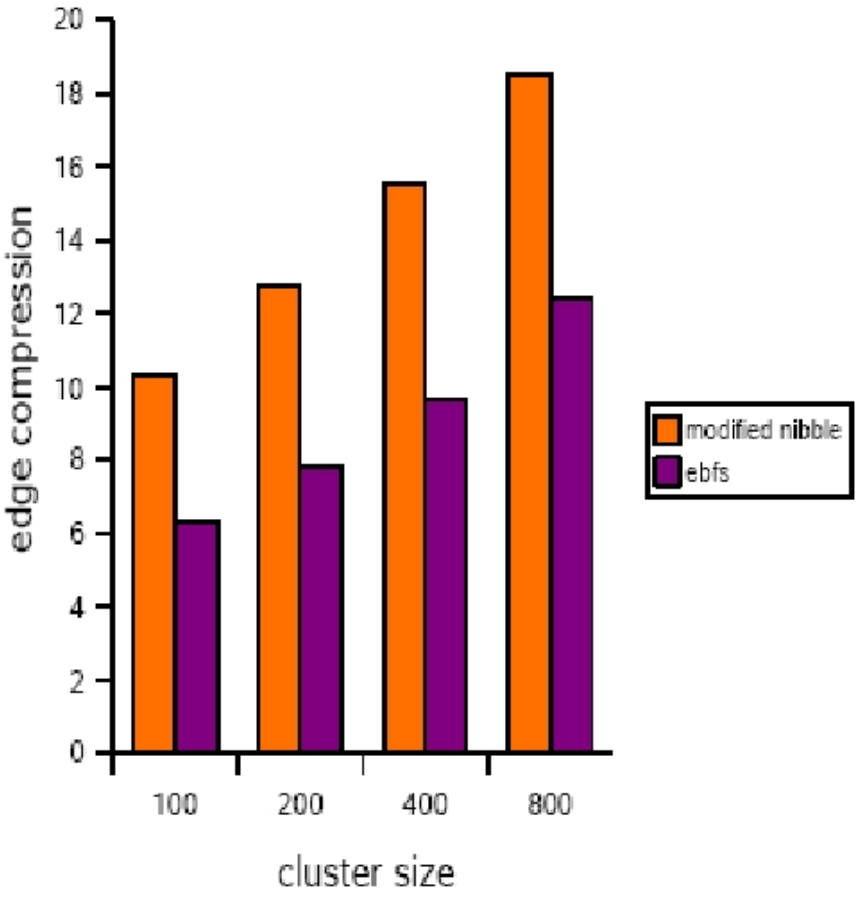
	Edge compression	
Cluster Size	**Without compaction**	**With compaction**
200	12.71	12.78
400	14.46	14.65
800	16.68	17.18

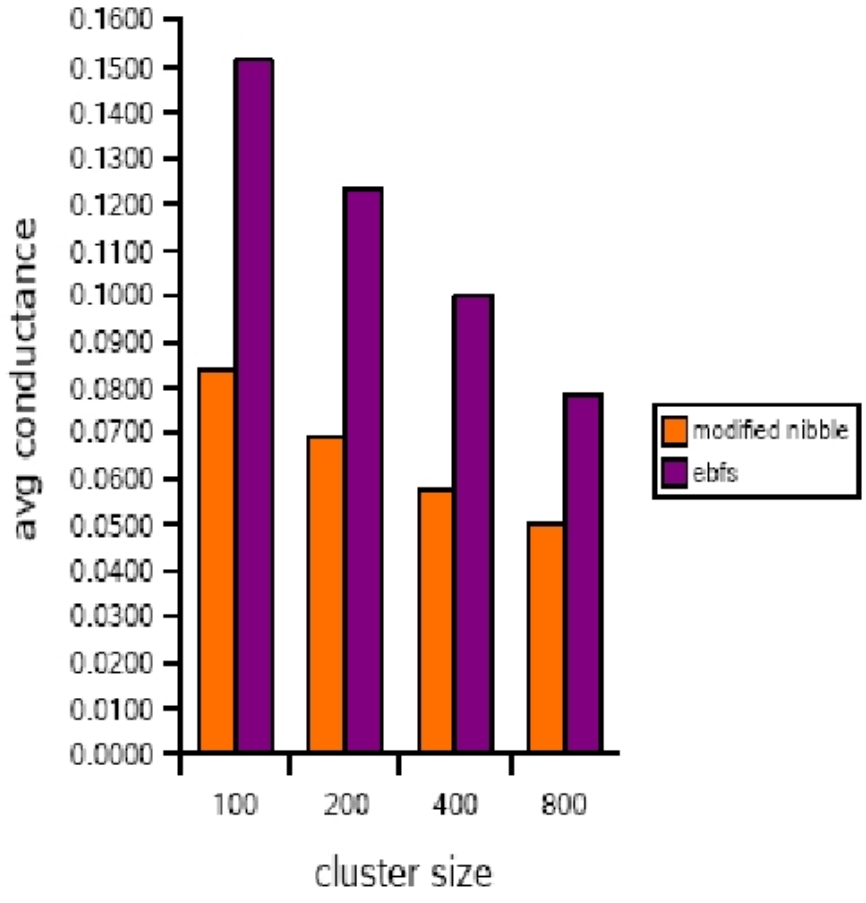
Different `f` ($a = 2$, $d = 5$, $r = 1.5$, `maxClusterSize` $= 1500$)

	Edge compression	
f	**Without compaction**	**With compaction**
100	18.98	20.18
150	19.24	20.51
200	19.53	20.82

- Improvement due to compaction is negligible.
- Indicates that the clustering found by the algorithm is good.

Comparison with EBFS clusters for dblp3





- Better compression and conductance for `Modified-Nibble`.
- Decrease in conductance accompanied by better edge compression - indicates: objective of minimizing the former, can give better values for the latter.

Performance improvement in BANKS

	% reduction			
Cluster Size	**100**	**200**	**400**	**800**
CPU + IO time taken	43.6	36.62	45.27	41.04
Number of nodes explored	-0.23	16.22	57.25	49.41
Number of nodes touched	22.85	26.49	59.09	60.47
Number of cache misses	48.71	36.41	39.74	17.53

- `Modified-Nibble` clusters are outperforming EBFS clusters, by a large margin.
- Clusters with better compression and conductance values, also perform well in the actual search system.

(*Details in [Sav09, Agr09]*)

Results for Flavor2 on dblp3

Flavor2: MinDegree startNode, continue on `maxActiveNodeBound`

Comparison between the two flavors

	MaxStart & terminate on bound (flavor1)	**MinStart & continue on bound (flavor2)**
# clusters	6,709	6,709
# inter-cluster edges	136,784	125,651
edge compression	15.53	16.91
avg conductance	0.0579	0.0549

`maxClusterSize` $= 400$, $a = 2$, $d = 7$, $r = 1.5$, $f = 500$ and with compaction

- Improvement is not significant, possibly because the basic algorithm is robust to the actual choice of start node, behaviour when the `maxActiveNodeBound` is reached.
- Correlation between conductance and compression.

Results for Flavor1 on `wiki` graph

Edge compression and Avg conductance

max Cluster Size	**f**	**# clusters**	**# inter-cluster edges**	**edge compression**	**avg conductance**
200	100	16,208	12,445,795	3.203	0.373

Possible reasons:

- Almost all the inherent clusters in wikipedia have their size > 200.
- The decision to stop as soon as the `maxActiveNodeBound` is reached is hurting the clustering to a very large extent.
- The community structure of `wikipedia` is different from that of `dblp`, and we are missing out on some important aspect of the former, which is absent in the latter.

Results for Flavor2 on `wiki`

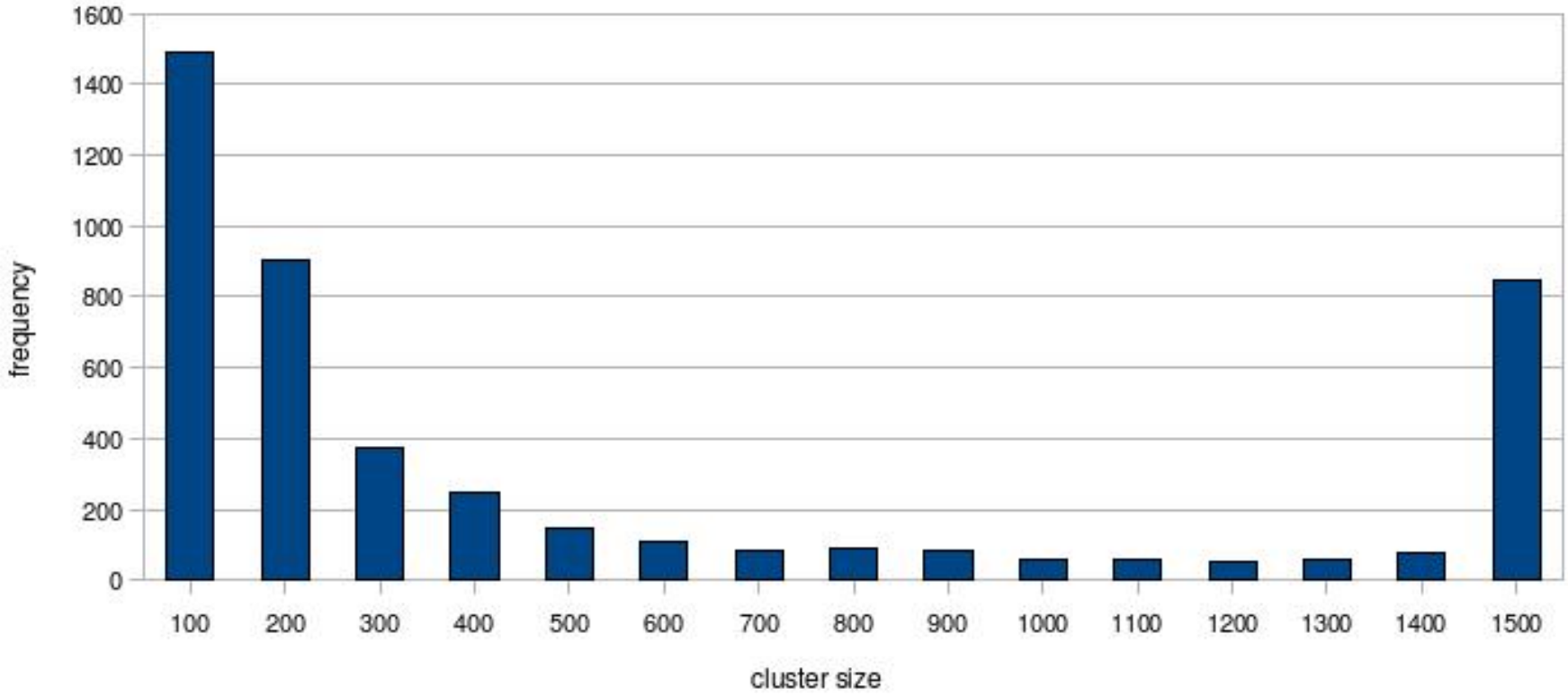
Flavor2: MinDegree startNode, continue on `maxActiveNodeBound`

Edge compression and avg conductance

max Cluster Size	f	# clusters	# inter-cluster edges	edge compression	avg conductance
200	100	17,713	11,485,314	3.471	0.350
200	500	17,413	10,670,635	3.736	0.328
1500	100	2,350	1,777,217	22.43	0.0803

- Not much improvement for cluster size of 200, when compared to Flavor1 - indicates the robustness of the basic algorithm.
- `f` doesn't seem to affect the clusters.
- Edge compression improves by 6.5 times on changing cluster size.
- Avg conductance becomes $1/4^{th}$.

Chart of cluster size vs. frequency of `wiki`



- There are many communities in `wikipedia` of large size.
- The last entry indicates that there are communities of even larger size.

Results on `wiki` for Flavor3

Flavor3: Remove hub nodes & execute Flavor2

Edge compression and Avg conductance

max Cluster Size	f	# clusters	# inter-cluster edges	edge compression	avg conductance
1500	100	2,294	1,334,752	29.867	0.0604

Number of top indegree nodes removed = 1500.

- Compression and conductance have improved.
- This indicates that there are many articles in Wikipedia to which a large number of articles link, even if they are not really related to it.

Summary of the experiment results I

- Compression and conductance improve with increasing cluster size. But, the improvement diminishes once `maxClusterSize` reaches the average size of the inherent clusters of the graph.
- The sizes of inherent clusters in `dblp3` seem to be around 100 to 400, whereas `wiki` seem to have many clusters of larger size.
- Compression and conductance of clusters appear to be strongly correlated.
- Clusters produced by `Modified-Nibble` on `dblp3`, outperforms those by EBFS, both in compression and conductance values, as well as, in the search performance of the BANKS system.

All of the above require more experiments on different datasets and with different parameter settings, for confirmation.

Summary of the experiment results II

Effect of heuristics:

- The basic algorithm seems to be robust to the actual choice of start node and the behaviour when `maxActiveNodeBound` is reached.
- Effect of `f` and compaction on the cluster quality are negligible.

Future Work I

- Test the performance of the BANKS system on `Wikipedia` graph for different queries, using the clusters producted by the `Modified-Nibble` algorithm.
- Clustering on Wikipedia:
  - Large number of hub pages, and disambiguation pages, which connect many unrelated concepts together, thus adding to the noise of the structure.
  - Node-degrees in wiki-graph (maxDegree $>$ 200,000) is much higher when compared to that in dblp3 (maxDegree $<$ 800).
  - The algorithm must be smart enough to deal with misbehaving nodes, to discover the underlying community structure.
- Intelligent compaction: Use the edges between clusters to judge their similarity.

Future Work II

- Improve the speed of clustering process by nibbling out many clusters simultaneously.
- Cluster the web graph with the help of wiki clusters:
  - For each webpage, find the wiki articles to which it is 'most similar'
  - Cluster webpages based on the cluster to which their corresponding wiki articles belong.
- Generalizing: find a good clustering on a larger graph, given a good clustering on a smaller graph, using a mapping from the former to the latter.

References

[Agr09] Rakhi Agrawal. Keyword Search in Distributed Environment. *MTech. Project Stage 2 Report, Indian Institute of Technology, Bombay,* 2009.

[AL06] Reid Andersen and Kevin J. Lang. Communities from Seed Sets. *Proceedings of the 15th international conference on World Wide Web,* pages 223-232, 2006.

[Sav09] Amita Savagaonkar. Keyword Search in Distributed Environment. *MTech. Project Stage 2 Report, Indian Institute of Technology, Bombay,* 2009.

[ST04] Daniel A. Spielman and Shang-Hua Teng. Nearly-Linear Time Algorithms for Graph Partitioning, Graph Sparsification, and Solving Linear Systems. *ACM STOC-04,* pages 81-90, 2004.

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

Objective: find the cluster to which the seed node belongs

Nibble Algorithm:

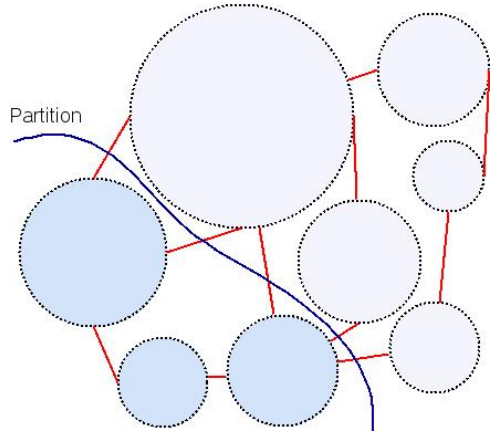
input: Start Vertex v, Graph G, Conductance θ_0, a positive integer b

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. Choose v according to the degree
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:

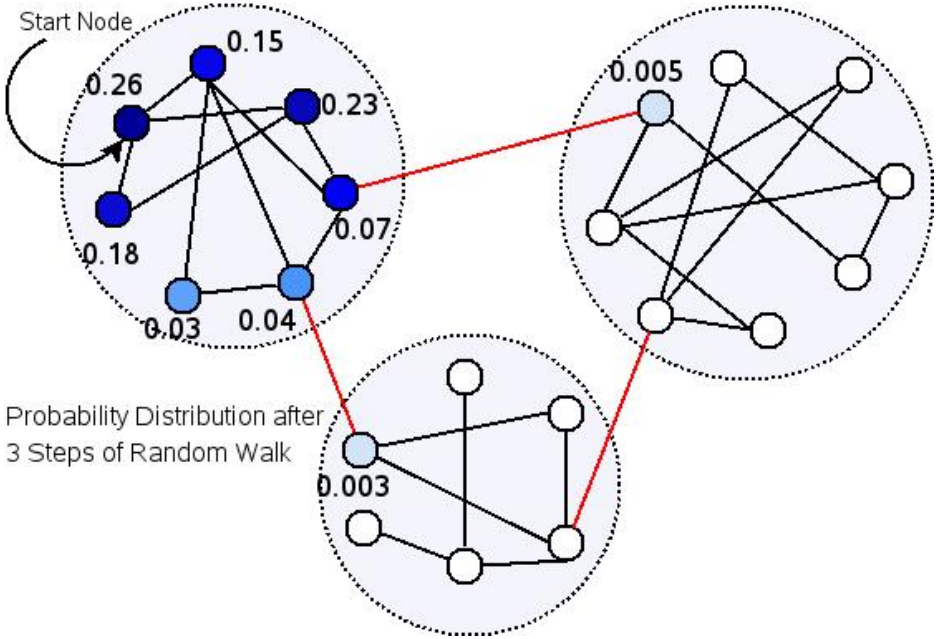
input: G, θ_0, $p \in (0, 1)$

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 $\mathcal{C}_{t+1}$

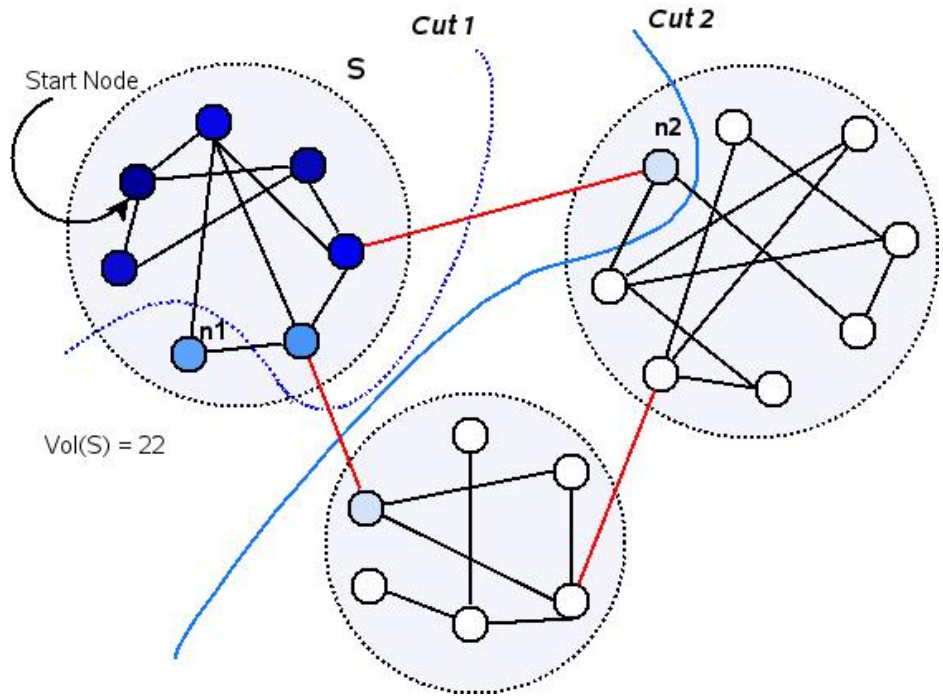


- Sudden drop in probability, outside the cluster boundary

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

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