Analysis and Optimization of Max Flow Min-cut

Nitin Mukesh Tiwari
Department of IT
NIT Srinagar

Swatie Bansal
Department of Computer Engineering

Abhishek Tripathi
Department of Computers
NIT Srinagar

ABSTRACT
Today we are working with the networks all around and that's why it becomes very important to find the effective flow of the commodity within that network. This paper aims to provide an analysis of the best known algorithm for calculating maximum flow of any network and to propose an approximate algorithm, which can solve the same problem with lesser complexity, desirably lesser than the complexity of the known Stoer-Wagner algorithm. This paper addresses this problem with a new approach, which uses upper bound values of each node in the network.Results are compared with fixed number of nodes and variable number of nodes in the network. Moreover networks with variable densities are also considered. Results are obtained by programming the both algorithms in C++.Unix scripts are also used for formatting the results.

Keywords
Min-cut, maximum flow, edge weighted graphs,.

1. INTRODUCTION
Ever since the evolution of networks, they have been the most important part of our lives. Today we have networks of computers, communities, railways, distributors, retailers, film industry etc. The underlying concept of each network is flow of information. No network can exist without information. This flow of information inside the network or between two or more networks is done on the cost of some resources. Computer scientists are trying to maximize the flow of information with minimum possible exploitation of resources. This journey started when a computer scientist Ford Fulkerson came up with a solution to this problem. He tried to find a augmentation path in the graph. If any augmentation path exists that means the flow is not maximized. After this Gomory and Hu proved that maximum flow through a graph is the flow through its min-cut. Moreover they also proved that min-cut or max flow between all pairs of vertices in an undirected graph can be computed by doing only n-1 max flow computation rather than naïve $\binom{n}{2}$ max flow computations. They proposed a Gomory Hu tree(minimum cut tree) which needs to create through the process and which gives the exact value of maximum flow of min-cut through that network. Since then many scientist have done enormous amount of work to achieve the minimum cut tree and luckily they succeed but now the major problem was not to solve the problem, it was to minimize the complexity of algorithm. Stoer and Wagner tried to solve the same problem but with minimum complexity. They proposed a solution having minimum complexity among the algorithms to solve this problem known till now. One thing that was common to all algorithms is that all use max flow subroutines. In this paper we proposed an approximation algorithm to solve the problem. In which we are calculating upper bound values of each vertex (node) and then we relax that upper bound value till we found the minimum cut of the graph.

2. PROBLEM STATEMENT
In any edge weighted graph having a vertex set V with edge set E, the problem is to build a tree such that $\forall$ u ,v ϵ V, edge having minimum weight on the unique path connecting u and v in the tree represents the value of minimum cut of graph separating u and v. This tree is known as minimum cut tree..

2.1 Min Cut Tree[7]
In any edge weighted graph having a vertex set V with edge set E,and weight function w:E ϵ R, It can be shown that out of all possible ${}^{n}C_{2}$ pairs of nodes there can be n-1 min-cuts. These n-1 min-cuts are represented by a (not necessarily unique) tree, called Min-Cut tree, and has the following properties:

1. The tree consists of the same number of nodes as that in initial graph and each edge is assigned a value which is not directly related to the weights of the initial graph.
2. We can find the minimum cut for every pair s,t in the tree by following the unique path between the nodes. For example if an edge have the minimum value on the path then that value is also the minimum cut in the original graph.
3. In order to find the cut between s and t we remove the edge with the minimum value on s-t path. The min-cut between s and t in the initial graph G is also defined by two connected sub sets of nodes in the tree.

2.2 Notations [7]
Cut: A cut of graph is basically partitioning its vertices in two subsets (V,S-V) and it is represented by C. Weight of a cut can be defined as the summation of all the edges that connect the two subsets V and S-V.

$w(C)= \sum w(u, v)$ where $(u,v)\epsilon E$ and $u\epsilon S$, $v\notin S$

s-t Cut: S-t cut for any two vertices s and t can be defined as a cut such that s ϵ S and t $\notin$ S .

Min s-t Cut: Among all the s-t cuts of the Graph G, cut having minimum value is known as Min s-t cut.

Min-Cut: Min cut of an undirected edged weighted graph G is set of edges with minimum sum of weights, such that removal of this leads to unconnected of the graph.

3. EXISTING APPROACHES
3.1 Max Flow Min Cut Theoram
According to the **max-flow min-cut theorem** in a flow network, the maximum possible flow from the *source* node to the *sink node*, is equal to the minimum capacity which when removed from the network causes no flow. The max-flow min-cut theorem states **the maximum value of an s-t flow is equal to the minimum capacity of an s-t cut**..

3.2 Gomory-Hu Algorithm[4]
According to the Gomory and Hu, a graph having n nodes can have n-1 numerically different flows. So all flows can be deduced after only n-1 different flows have been computed. Consider a flow network whose nodes have been separated in

two sets A and A', by a minimum cut(A, A'), in such a way that nodes $N_i \in A$ and $N_j \in A'$. Now if all nodes in A' are replaced by a single node P to which all the arcs of the cut are attached then a new condensed network will result. In this condensed network consider the maximum flow between two ordinary nodes N_e and N_k. Gomory and Hu proposed the following lemma:

Lemma II: the flow between two ordinary nodes N_e and N_k in the condensed network is numerically equal to the flow f(e,k) in the original network.

3.3 Stoer Wagner Algorithm[3]

M.Stoer and F.Wagner have given a simple and compact algorithm for finding the minimum cut of a graph. The algorithm is remarkably simple and has the fastest running time so far. The algorithm uses the following very interesting theorem.

Theorem I: Let s and t be two vertices of a graph G. Let G/{s, t} be the graph obtained by merging s and t. Then a minimum cut of G can be obtained by taking the smaller of a minimum s-t-cut of G and a minimum cut of G/{s, t}.

4.5.2.1 Proof: The theorem holds since either there is a minimum cut of G that separates s and t, then a minimum s-t-cut of G is a minimum cut of G; or there is none, then a minimum cut of G/{s, t} does the job. So a procedure finding an arbitrary minimum s-t-cut can be used to construct a recursive algorithm to find a minimum cut of a graph.

Procedure:
MINIMUMCUTPHASE (*G*, *w*, *a*)
$A \leftarrow \{a\}$
While ($A \neq V$)
Add to *A* the most tightly connected vertex
Store the cut-of-the-phase and shrink *G* by merging the two vertices added last.

A subset *A* of the graphs vertices grows starting with an arbitrary single vertex until *A* is equal to *V*. In each step, the vertex outside of *A most tightly connected* with *A* is added. Formally, we add a vertex

$z \notin A$ such that $w(A, z) = \max\{w(A,y) | y \notin A\}$
Where $w(A, y)$ is the sum of the weights of all the edges between *A* and *y*. At the end of each such phase, the two vertices added last are *merged*, that is, the two vertices are replaced by a new vertex, and any edges from the two vertices to a remaining vertex are replaced by an edge weighted by the sum of the weights of the previous two edges. Edges joining the merged nodes are removed.

The cut of *V* that separates the vertex added last from the rest of the graph is called the *cut-of-the-phase*. The lightest of this cuts-of-the-phase is the result of the algorithm, the desired minimum cut.

MINIMUMCUT(G, w, a)
while |V|> 1
MINIMUMCUTPHASE(G, w, a)
if the cut-of-the-phase is lighter than the current minimum cut
then store the cut-of-the-phase as the current minimum cut

Notice that the starting vertex *a* stays the same throughout the whole algorithm. It can be selected arbitrarily in each phase instead.

3.4 Run Time Complexity

The algorithm consist of |V|-1 identical phases each of which requires O(|E|+|V|log|V|) time yielding an overall running time of $O(|V||E|+|V|^2 \log|V|)$..

4. Our Approach

This work proposes a new approximation algorithm for constructing the minimum cut tree. Upper-bound value for each node of the graph is calculated. Upper-bound value of any node is defined as the value of edge, which upon removal separates this node from rest of the graph i.e.

Upper bound(u)= $\sum w(u, v)$ where $v \in$ Adj(u)
Lemma IV: The value of minimum cut of a graph G separating Ni and Nj is less than or equal to minimum of the upper bound values of two nodes Ni and Nj.

Proof: Simple reasoning can be given to prove it. Let (A,A') represent the minimum cut of G, which separates both Ni and Nj. Upper-bound(Ni) and Upper-bound(Nj) are the values of two cuts which also separate Ni and Nj. Therefore,

$w(A,A') \leq$ min(Upper-bound(Ni), Upper-bound(Nj)) where $w(A,A') = \sum w(Ni, Nj)$ where $Ni \in A$ and $Nj \in A'$

4.1 Algorithm

Find an edge uv such that upon merging the two nodes N_u and N_v upper bound values reduces i.e.
max(Upperbound(N_u),Upperbound(N_v))>Upperbound(N_u)+Upp-erbound(N_v)-2*w(u,v)

Start from the node having the minimum upper-bound value and check for all of the edges leaving it. If upper bound values can be reduced by merging it with any of the nodes, then nodes are merged and procedure is repeated. If reduction of upper bound values is not possible , then check for rest of the nodes in the increasing order of upper-bound values. The reason behind considering the nodes in increasing order of upper-bound values will be clear in next lemma.

After all the nodes in the graph are merged and it has only one node left, construct the min-cut tree by using the information from intermediate stages. Move from last to first stage and at each stage we see the two nodes that were merged during last stage and separate the node with smaller of the two upper-bound values from the other by an arc bearing the value equal to the smaller of the two upper-bound values. Since we separate the two merged nodes in the tree by an arc having the value equal to smaller of the two upper-bound values, it is necessary to consider the nodes during merging process in the increasing order of upper-bound values so that the node with less upper-bound value will be merged first , if possible all.

Lemma V: If nodes to be merged are considered in the increasing order of upper-bound values, and while examining the adjacency list of Ni, Nj is found, as the node which upon merging with Ni will reduce the upper-bound value of Ni, then

1. Either upper-bound(Nj) will also be reduced.
2. Or upper-bound(Ni) can't be reduced.

Proof: Nodes are considered in the increasing order of upper-bound values, and so
Case 1: upper-bound(Nj) ≥ upper-bound(Ni)
Since merging Ni and Nj reduces the upper-bound value of Ni, therefore
Upper-bound(Nj) + upper-bound(Ni) – 2*w(i, j) ≤ upper-bound(Ni)
It is clear from above two equations :

Upper-bound(Nj) + upper-bound(Ni) – 2*w(i, j) ≤ upper-bound(Nj)
i.e. upper-bound(Nj) is also reduced
Case 2: upper-bound(Nj) < upper-bound(Ni)
Since **N**odes are considered in the increasing order of upper-bound values, , checking for Ni itself implies that Nj has already been checked and it was not possible to reduce its upper-bound value at all. So in this case upper-bound(Nj) can't be reduced.

If it is not possible to merge any node at any stage , then node causing minimum increase in Upper bound values is merged.

4.2 Assumptions

This algorithm is based on the assumption that if two nodes Ni and Nj are being merged and if upper-bound(Ni) < upper-bound(Nj) then it is not possible to merge Nj with any other node which will result in a node having upper-bound value which is less than upper-bound(Ni). However , this assumption is not true always.

During the course of this algorithm, if a pair of nodes Ni and Nj are being merged such that:

- upper-bound(Ni) < upper-bound(Nj) and,
- It was possible to merge Nj with some other node Nk such that upper-bound value of the resulting node(let it be *val*) would have been less than upper-bound(Ni),

In this case, the resulting min-cut tree will not be correct and will give wrong min-cut values for some pair of nodes. More precisely, it would give the value of min Ni-Nj cut as upper-bound(Ni) but the correct value is *val*. We call such a pair of nodes Wrong pair to merge.

For sparse graphs, the probability of choosing the wrong pair of nodes to merge is high due to the less number of available pairs among which to choose i.e. due to less number of edges.

For dense graph, the probability choosing the wrong pair of nodes to merge is very less because of the large number of available pairs among which to choose..

For dense graph this algorithm produces surprisingly good results. After running the procedure with more than 15000 randomly generated graphs we have figured out that for all graph densities, success rate of algorithm is approximately 100%.

4.3 Time Complexity $O(V^2.logV+V^2.d)$

Where V is no. of vertices in given graph and d is degree of graph.

Min-Cut Tree(G)

Input: Edge-weighted Undirected graph G

Output: Min-Cut Tree

Calculate the upper-bound values for each node.

While (number of vertices in the current graph >1)

Loop (Consider the vertices in the increasing order of upper-bound value)

If (upper-bound value can be reduced by merging a node with any adjacent node)

Then those two adjacent nodes are merged

Break:

End if

End loop

If (It is not possible to merge any pair of nodes)

Pair of nodes which results in minimum increment of the upper-bound value are merged

End if

End While

Min-Cut Tree T is constructed by using the information collected from intermediate stages as described:

a. Move from last to first stage.

b. At each stage check the two nodes that were merged during last stage.

c. Separate the node with lower upper-bound value from the other by an arc bearing the value equal to the lower upper-bound value.

Return T

5. RESULTS

5.1 Random Graph with random number of nodes

We generated 7500 random graphs of different densities and number of nodes in them were also random(5-55). Edges weights were also random and were between 1-300.Results of running our algorithm with these graphs are summarized as follows:

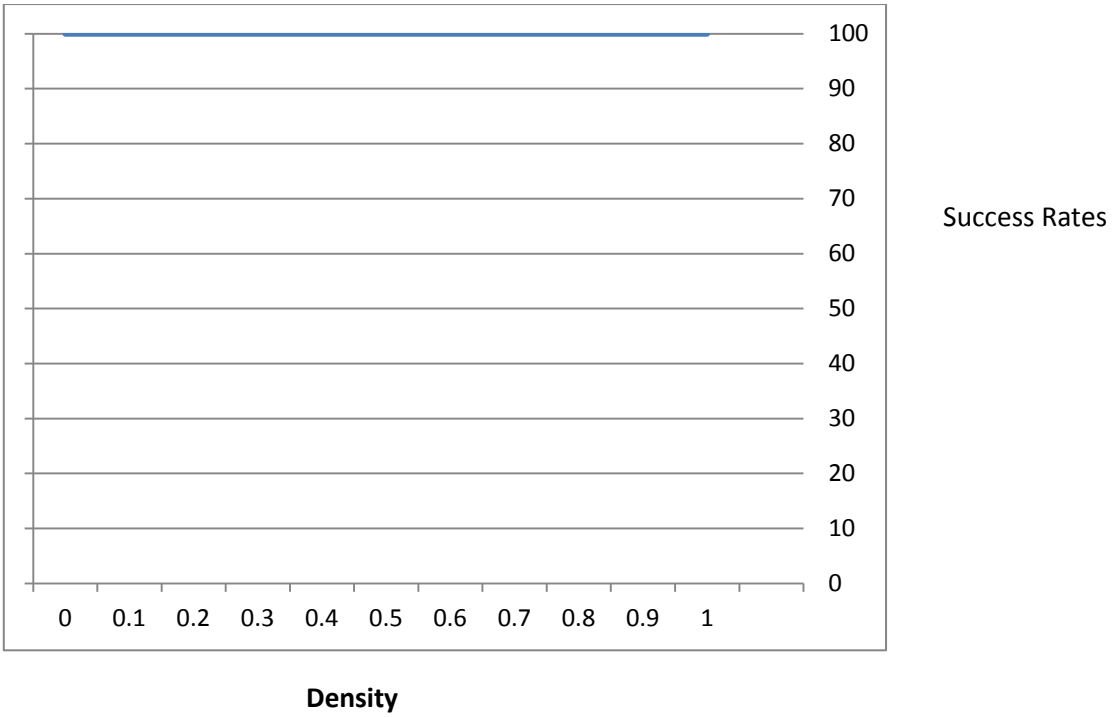

Figure 1 Density v/s Success for graph having random number of nodes

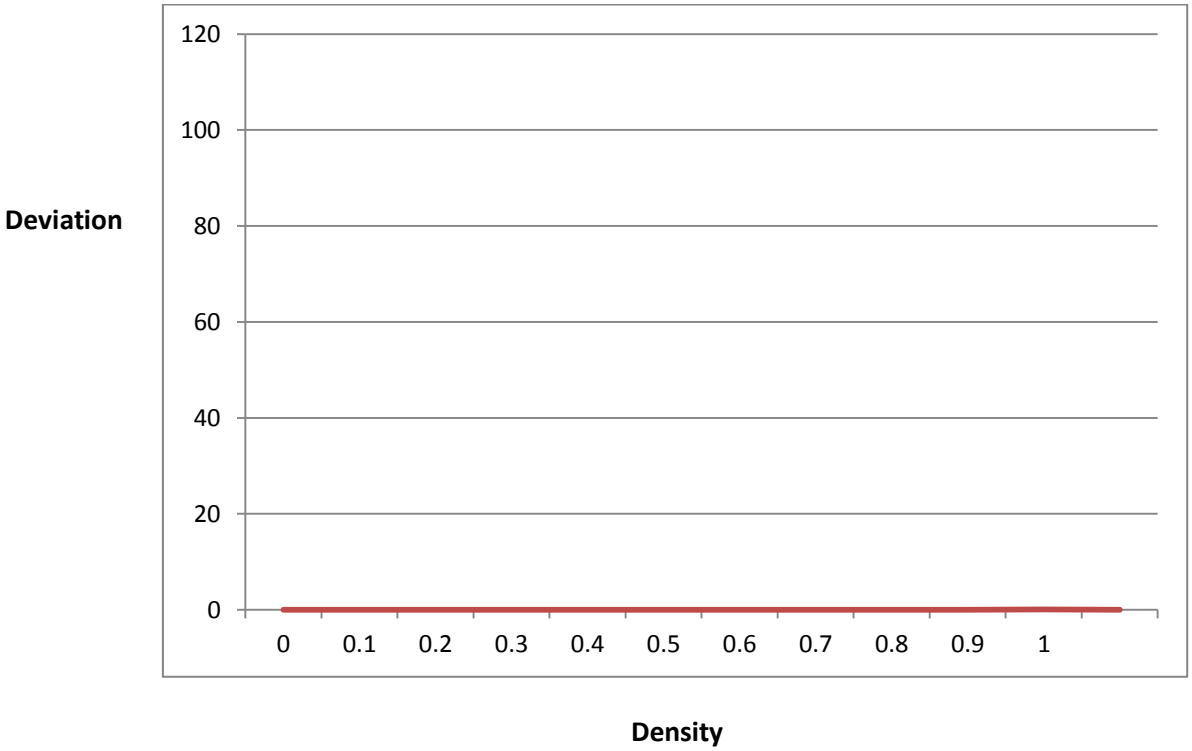

Figure 2 Density v/s Deviation graph for random number of nodes in Graph

It is clear from the graph that success rate is around 100%. But the results may have an error of 0.5% for the cases of extremely sparse graphs. But as the connectivity of graph increases the error is eliminated.

5.2 Random Graph with fixed number of nodes (55 nodes)

We generated 7500 random graphs of different densities and number of nodes in them were also random(5-55). Edges weights were also random and were between 1-300.Results of

running our algorithm with these graphs are summarized as follows:

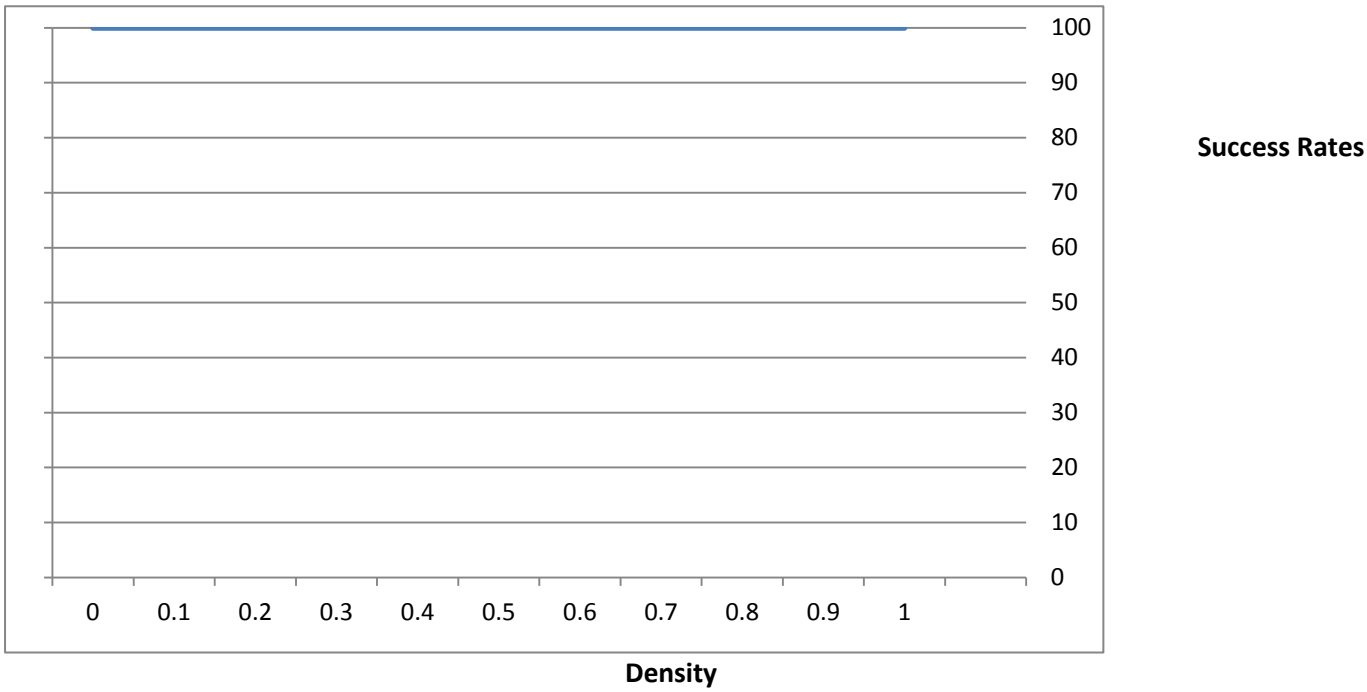

Figure 3 Density v/s Deviation for graph having 55 nodes

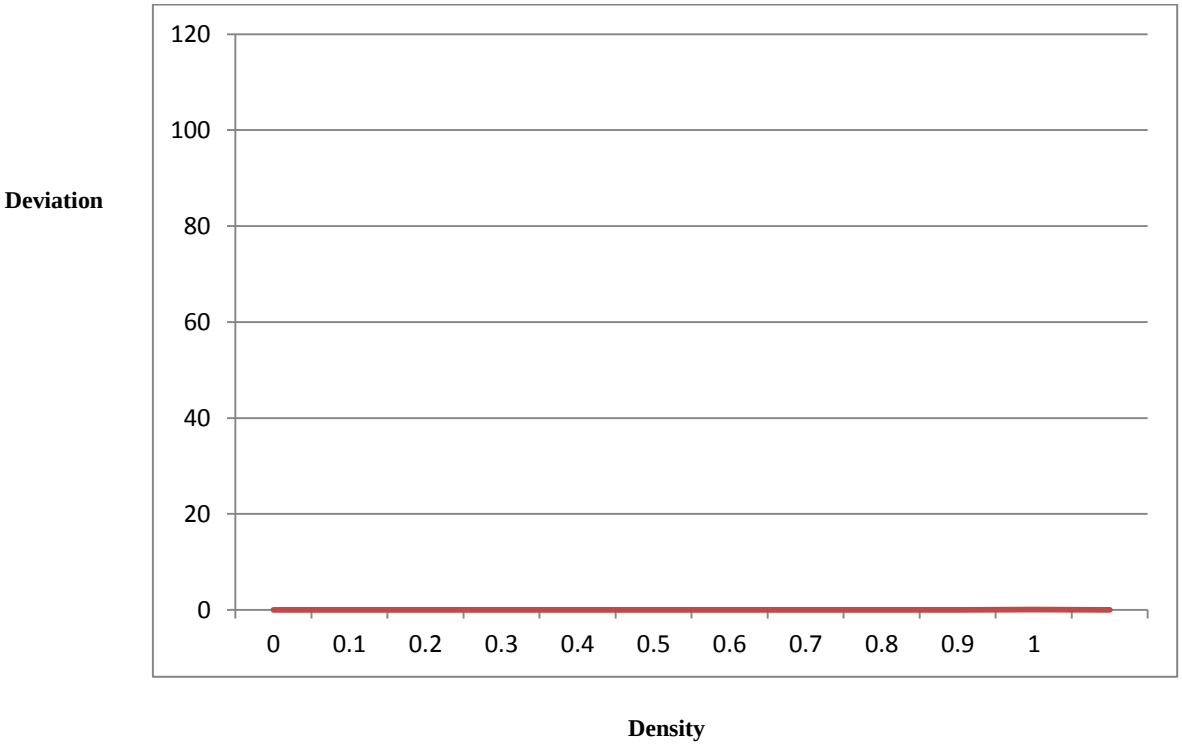

Figure 4 Density v/s Deviation graph for 55 nodes in Graph

t is clear from the graph that success rate is around 100%. But the results may have an error of 0.5% for the cases of extremely sparse graphs. But as the connectivity of graph increases the error is eliminated.

6. ACKNOWLEDGMENTS

Our thanks to my head of department Mrs Roohie Naaz Mir for providing me guidance for the paper and technica aspects.

7. REFERENCES

[1] D. R. Karger, "Minimum cuts in near-linear time." Journal of the ACM, vol. 47, 2000..

[2] GauravAggarwal, Rajeevkumar, DineshSharma, Anshul Meena,Mausham Ghosh, "Min-Cut Tree", 2010

[3] MECHTHILD STOER, FRANK WAGNER, "A Simple Min-Cut Algorithm", 1995

[4] R. E. Gomory, T. C. Hu. "Multi-terminal network ", Journal of the Society for Industrial and Applied Mathematics, vol. 9, 1961.

[5] Schrijver, "Combinatorial Optimization.", Springer-Verlag Berlin Heidelberg, 2003..

[6] S. M. Sadegh, Tabatabaei Yazdi Serap A. Savari ,"A Max-Flow/Min-Cut Algorithm for a Class of Wireless Networks ".

[7] Thomas H. Cormen, Leiserson, Reivest, Stein (MIT), "Introduction to algorithms", MIT Press – 2nd Edition 2001.

[8] Yuri Boykov and Vladimir Kolmogorov, "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision ", 2004

[9] Steven S. Skiena, "Algorithms design Manual".