A Novel Approaches on Clustering Algorithms And it's Applications

B.Venkateshwar Reddy, T. Asha Latha

Abstract— Graph clustering algorithms are Random walk and minimum spanning tree algorithms. Random walk has been used to identify significant vertices in the graph that receive maximum flow while minimum spanning tree algorithm has been used to identify significant edges in the graph .We believe these two graph algorithms have useful applications in clustering, namely for identifying centroids and for identifying edges to merge or split clusters such that intra-cluster similarity is maximized while inter-cluster similarity is minimized. This paper investigates the graph algorithms, graph-based clustering algorithms, and their applications. graph algorithms and graph-based clustering algorithms, we propose novel variants of Star clustering algorithm that use different techniques for identifying centroids, and two novel graph-based clustering algorithms: MST-Sim and Ricochet. The variant graph algorithms and graph based clustering algorithms achieve higher performance in terms of effectiveness and efficiency for the applications of document clustering, k-member clustering, opinion mining, clustering for part-of-speech tagging.

Keywords— Clustering algorithms, graph based clustering algorithms,

1.INTRODUCTION

A graph is a finite set of nodes with edges between nodes Formally, a graph G is a structure (V,E) consisting of a finite set V called the set of nodes, and set E that is a subset of VxV. that is, E is a set of pairs of the form (x, y) where x and y are nodes in V. Given an undirected, connected graph G(V,E) with |V| = n, |E| = m a random "step" in G is a move from some node u to a randomly selected neighbor v. A random walk is a sequence of these random steps starting from some initial node. Given a connected, weighted, bi-directed graph G = (V, E), a spanning tree of G is a subgraph which is a tree that connects all the vertices in G. The weight of a spanning tree is the sum of the weights of edges in that spanning tree. A minimum spanning tree (MST) of G is a spanning tree whose weight is less than or equal to weight of every other spanning tree of G. More generally, any weighted, bi-directed graph (not necessarily connected) has a minimum spanning forest, which is a union of MSTs of its connected components. Well known algorithms for finding MST are Kruskal's algorithm [1], Borůvka's algorithm [2], and Prim's algorithm [3], which are all greedy algorithms. Faster randomized MST algorithm has been developed in [4].

- Mr.B.Venkateshwar Reddy is working as Asst.Professor in Department of Computer Science & Engineering, Anurag Group of Institutions, Hyderabad, India. E-mail: stragreddi@gmail.com.
- *Mrs. Asha Latha is* working as Asst.Professor in Department of Computer Science & Engineering, Anurag Group of Institutions, Hyderabad. India. E-mail: *asha.thandu@gmail..com*

2. CLUSTERING ALGORITHMS

Several algorithms have been proposed for clustering. They can be grouped into the following categories: partitioning algorithms, hierarchical algorithms, and graph-based algorithms.

2.1 Partitioning Clustering Algorithm

K-means clustering algorithm [5] divides the set of vertices of a graph into K clusters by first choosing randomly K seeds or candidate cetroids. It then assigns each vertex to the cluster whose centroid is the closest. K-means iteratively re-computes the position of the exact centroid based on the current members of each cluster, and reassigns vertices to the cluster with the closest centroid until a halting criterion is met (e.g. centroids no longer move). The number of clusters K is defined by user a priori and does not change.

2.2 Hierarchical Clustering Algorithms

Hierarchical algorithms [6] can be categorized into agglomerative and divisive ones. Agglomerative algorithms treat each vertex as a separate cluster, and iteratively merge clusters that have the greatest similarity from each other until all the clusters are grouped into one. The objective function of hierarchical clustering is intra-cluster similarity; i.e. greatest similarity at each merger. Divisive algorithms start with all vertices in one cluster, and subdivide it into smaller clusters.

3. GRAPH-BASED CLUSTERING ALGORITHMS

Graph-based algorithms for clustering create clusters by cutting or removing edges that are deemed unimportant according to some measurement. We have seen several graph-based clustering algorithms which include Minimum Spanning Tree (MST) clustering, Chameleon, Markov clustering, and Star clustering.

3.1 MST Clustering

Zahn's MST clustering algorithm [7] is a well known graph-based algorithm for clustering [8]. The implementation of Zahn's algorithm starts by finding a minimum spanning tree in the graph and then removes inconsistent edges from the MST to create clusters [9]. Inconsistent edges are defined as edges whose distances are significantly larger (e.g. *c* times larger) than the average distance of the nearby edges in the MST; where *c* is a measure of significance and is a userdefined constant. The objective function of Zahn's algorithm is inter-cluster sparsity; i.e. maximum distances in-between clusters. Zahn's algorithm requires constructing the complete MST to determine inconsistent edges.

3.2 Chameleon

Chameleon [10] is a graph-based clustering algorithm that stems from the need for dynamic decisions in place of static parameters. Chameleon first constructs a sparse graph of K-nearest neighbour as an initial threshold. It then uses a min-cut based algorithm to partition this graph. Finally, it iteratively and dynamically merges the sub graphs considering their measures of relative inter-similarity and closeness based respectively, on the sum and average weight of edges in the min-cut of and in-between the clusters. The objective function of Chameleon is intercluster sparsity (by using min-cut to partition the graph) and intra-cluster similarity (by using relative inter-similarity and closeness to merge subgraphs). Like hierarchical clustering, Chameleon produces a dendrogram of possible clusters at different levels of granularity. In effect, Chameleon uses three parameters: the number of nearest neighbors, the minimum size of the sub graphs, and the relative weightage of inter-similarity and closeness

3.3 *Markov Clustering*

Markov Clustering (MCL) algorithm is a form of graph-based clustering algorithm that is based on simulation of stochastic flow (or random walks) in graphs [11]. The aim of MCL is to separate the graph into regions with many edges inside and with only a few edges between regions. Once inside such a region, the flow (or a random walker) has little chance to flow out [12]. To do this, the graph is first represented as stochastic (Markov) matrices where edges between vertices indicate the amount of flow between the vertices: i.e. similarity measures or the chance of walking from one vertex to another in the graph. MCL algorithm simulates flow using two alternating simple algebraic operations on the matrices: expansion, which coincides with normal matrix multiplication, and inflation, which is a Hadamard power followed by a diagonal scaling.The expansion process causes flow to spread out and the inflation process represents the contraction of flow: it becoming thicker in regions of higher current and thinner in regions of lower current. The flow is eventually separated into different regions, yielding a cluster interpretation of the initial graph. Like Chameleon, the objective function of MCL is intra-cluster similarity and inter-cluster sparsity. However, MCL is computationally expensive as it involves expensive matrix operations.

3.4 Star Clustering

Star clustering is another graph-based algorithm. The algorithm, first proposed by Aslam et al. In 1998 [13], replaces the NP-complete computation of a vertex-cover by cliques by the greedy, simple and inexpensive computation of star shaped dense sub graphs. Star clustering starts by removing edges whose weight is heavier than certain threshold. It then assigns each vertex to its adjacent star center, which is a vertex with equal or higher degree. Each star center and its adjacent vertices is a cluster. The objective function of Star is both intra-cluster similarity (by selecting star centers with high degree to potentially maximize intra-cluster similarity) and inter-cluster sparsity (by removing edges whose weights are above certain threshold).

4. PROPOSED METHODS FOR GRAPH BASED CLUSTERING ALGORITHMS

In this research, we study existing graph-based clustering algorithms and evaluate their choice of similarity metrics, empirically and theoretically using the basis of graph theory; and using different measure of threshold when applicable. We have performed a study on Star clustering and MST clustering. Combining the ideas from Star clustering, MST, and K-means, we propose a novel family of graph-based clustering algorithms called Ricochet that does not require any Parameters to be defined a priori.

4.1 Graph Algorithms

4.1.1 A Variant of Randomized MST Algorithm

Here we present our proposed variant of randomized MST algorithm. The randomized MST algorithm several drawbacks of the algorithm. Firstly, the decision on how many Borůvka's steps applied at each recursive call of the algorithm is not clear. Applies 2 Borůvka's steps at each recursive call of the algorithm while [14] applies 4 Borůvka's steps at each recursive call of the algorithm. If there are too many Borůvka's steps, the randomized MST algorithm is no different than Borůvka's algorithm; too few and the randomized algorithm will not be able to run in linear time. It is not clear what the best number is for the application of Borůvka's steps in the algorithm.Secondly, given a graph G and a random subgraph G1 obtained from G, the algorithm recursively finds the MST T of G1 and use T to identify *T-heavy* and *T-light* edges in G. Based on the cycle property however; any random tree T in the graph (not necessarily MST) can be used to identify *T-heavy* edges. The choice of the tree is irrelevant to the result, i.e. the MST of G. By recursively finding the MST of G1, the algorithm may incur additional cost when they could have

selected and used any tree in G. Given any tree T in the graph, the linear time MST verification algorithm [15] can be used to identify *T-heavy* edges in the graph. Given a tree T from G, the verification algorithm [15] first builds a Borůvka tree B by applying Borůvka algorithm on T. Since T is a tree with N-1 edges, B can be constructed in O (N) time where N is the number of vertices in G. The property of Borůvka tree B necessitates that the weight of the heaviest edge in the path between two vertices x and y in T equals the weight of the heaviest edge in the path between x and y in B, which can be found in unit time once B is constructed [15]. An edge (x, y) with weight c (x, y) in G is *T-heavy* (and therefore is not in the MST of G) if the heaviest edge in the path between x and y in B is lighter than c (x, y). We abandon the idea of Borůvka steps application and the recursive discovery of MST in random sub graphs to propose a variant of randomized MST algorithm. Our proposed algorithm does not need to decide how many Borůvka steps to apply and it fully utilizes the cycle and cut property of graphs. The algorithm is detailed in figure 1. The complexity of the algorithm is at most O (MN) because it needs to iterate through at most M edges in G. Each iteration may cost a re-construction of the Borůvka tree when an edge is added or removed from the tree. A construction of the Borůvka tree costs at most O (N) since there are only N-1 edges in the tree [15]. Hence the complexity of the algorithm is at most O (MN). The complexity of our algorithm depends on its selection of the initial random subgraph T. The algorithm can run faster if it selects a good T from G, i.e. T which is close to the MST of G.

Algorithm: A variant of randomized MST Algorithm
1. Obtain a subgraph T of G by randomly including N-1 edges from G
2. Construct a Borůvka tree B from T
3. Identify edges in T that are not selected to construct B, delete these edges from G and T
4. Using a linear-time verification algorithm, identify T-heavy edges in G and delete them from G and T, i.e. for each edge *e* = (x, y) in G:
   a. If *e* is T-heavy, delete it from G and T
   b. If *e* is T-light:
      i. identify the heaviest edge *e'* in the path between x and y in B that makes *e* Tlight and remove *e'* from G and T
      ii. add *e* to T
      iii. reconstruct the Borůvka tree B from T
      iv. delete edges that are not selected to construct B from G and T
   c. If *e* connects disconnected component in T:
      i. add *e* to T
      ii. reconstruct the Borůvka tree B from T
      iii. delete edges that are not selected to construct B from G and T
5. Return remaining edges in G

Figure 1: A variant of randomized MST Algorithm

In this research, we study existing graph-based clustering algorithms and evaluate their choice of similarity metrics, empirically and theoretically using the basis of graph theory; and using different measure of threshold when applicable. We have performed a study on Star clustering and MST clustering. Combining the ideas from Star clustering, MST, and K-means, we propose a novel family of graph-based clustering algorithms called Ricochet that does not require any parameters to be defined a priori.

4.1.2 Star Clustering

To produce reliable document clusters of similarity σ (i.e. clusters where documents have pairwise similarities of at least σ, where σ is a user-defined threshold), the Star algorithm starts by representing the document collection by its σ-similarity graph. A σ-similarity graph is an undirected, weighted graph where vertices correspond to documents and there is an edge from vertex vi to vertex vj if their cosine similarity in a vector space is greater than or equal to σ. Star clustering formalizes clustering by performing a minimum clique cover with maximal cliques on this σ-similarity graph (where the cover is a vertex cover). Since covering by cliques is an NPcomplete problem, Star clustering approximates a clique cover greedily by dense sub-graphs that are star shaped. A star shaped sub-graph of m + 1 vertices consisting of a single Star center and m satellite vertices, where there exist edges between the Star center and each satellite vertex. Star clustering guarantees pair-wise similarity of at least σ between the Star and each of the satellite vertices. However, it does not guarantee such similarity between satellite vertices. By investigating the geometry of the vector space model, Aslam et al. derive a lower bound on the similarity between satellite vertices and predict that the pair-wise similarity between satellite vertices in a Starshaped sub-graph is high. In their derivation of expected similarity between satellite vertices in a Star cluster [16], Aslam et al. show that the lower bound of similarity cos (γi,j) between two satellite vertices vi and vj in a Star cluster is such that:

$$\cos(\gamma i,j) \geq \cos(\alpha i)\cos(\alpha j) + (\sigma / \sigma + 1)\sin(\alpha i)\sin(\alpha j)$$

where cos (αi) is the similarity between the Star center v and satellite vi and cos (αj) is the similarity between the Star center v and satellite vj. They also show empirically that the right hand side of inequality above is a good estimate of its left hand side.

4.1.3 MST-Sim Clustering

In this research, we propose a novel MST-based graph-based clustering algorithm, called MSTSim, that draws inspiration from Kruskal's algorithm and Borůvka's algorithm for finding minimum spanning tree (MST) in the graph. There are several well known algorithms for finding MST in a weighted graph. Kruskal's algorithm sorts edges in ascending order of their weight. Edges and the vertices they connect are

added to the minimum spanning tree in this order. The algorithm terminates when all vertices have been connected into a single component. Borůvka's algorithm scans the set of vertices and connects each vertex to the vertex or sub-tree with the lightest edge. This step is iterated for sub-trees until all vertices are connected into a single component. Prim's algorithm starts with an arbitrary vertex and repeatedly connects a new adjacent vertex with the lightest edge. Reverse-delete algorithm [17] deletes edges from the original graph in a descending order of their weight and that do not disconnect the graph until the result is a tree.

Our proposed family of MST-Sim clustering algorithms is different in several aspects from the existing MST clustering algorithms. Firstly, our algorithms do not require any input information such as desired number of clusters or threshold to be provided before clustering. At most they use an inflation parameter that gives a fine tuning capability for effectiveness. Secondly, our algorithms find clusters while they are constructing the MST rather than a posteriori. Thirdly, our simple algorithms consider both inter-cluster sparsity – edges in-between clusters and intracluster density – edges within clusters when constructing clusters.

4.2 Ricochet: a Family of Unconstrained Graph-based clustering

Unlike Star clustering algorithm that is parameterized by the edge threshold and MST clustering that is parameterized by the inflation parameter, Ricochet is unconstrained. Ricochet algorithms alternate two phases: the choosing of vertices to be the centroids of clusters and the assignment of vertices to existing clusters. The motivation underlying our work is that: (1) Star clustering algorithm provides a metric of selecting Star centers that are potentially good cluster centroids for maximizing intra-cluster similarity, (2) K-means provides an excellent vertices assignment and reassignment, and a convergence criterion that increases intra-cluster similarity at each iteration, (3) minimum spanning tree algorithm provides a means to select edges to merge clusters that potentially maximizes intra-cluster similarity.

The Ricochet family is twofold. In the first Ricochet sub-family, centroids are chosen one after the other ('stones are thrown one by one'). In the second Ricochet sub-family, centroids are chosen at the same time ('stones are thrown together'). We call the former algorithms Sequential Rippling, and the latter Concurrent Rippling. The algorithms in the Sequential Rippling, because of the way they select centroids and assign or re-assign vertices, are intrinsically hard clustering algorithms, i.e. they produce disjoint clusters. The algorithms in the Concurrent Rippling are soft clustering algorithms, i.e. they produce possibly overlapping clusters.

4.2.1 Sequential Rippling (SR)

The first algorithm of the subfamily is call Sequential Rippling (or SR). In this algorithm, *vertices* are ordered in descending order of the average weight of their adjacent edges (later referred to as the weight of a vertex). The vertex with the highest weight is chosen to be the first centroid and a cluster is formed by assigning all other vertices to the cluster of this first centroid. Subsequently, new centroids are chosen one by one from the ordered list of vertices. When a new centroid is added, vertices are re-assigned to a new cluster if they are closer to the new centroid than they were to the centroid of their current cluster (if no vertex is closer to the new centroid, no new cluster is created). If clusters are reduced to singletons during re-assignment, they are assigned to the nearest non-singleton cluster. The algorithm stops when all vertices have been considered. The pseudocode of the Sequential Rippling algorithm is given. The worst case complexity of Sequential Rippling algorithm is O(N3) because in the worst case the algorithm has to iterate through at most N vertices, each time comparing the distance of N vertices to at most N centroids.

Algorithm: SR ( )
Sort V in order of *vertices'* weights
Take the heaviest vertex v from V
listCentroid.add (v)
Reassign all other vertices to v's cluster
While (V is not empty)
    Take the next heaviest vertex v from V
    Reassign vertices which are more similar to v than to other centroid
    If there are re-assignments
        listCentroid.add (v)
        Reassign singleton clusters to its nearest centroid
For all i ε listCentroid return i and its associated cluster

Figure 2: Sequential Rippling (SR)

4.2.2 Concurrent Rippling (CR)

The first algorithm of the sub-family is called Concurrent Rippling (CR). In this algorithm, for each vertex, the adjacent *edges* are ordered in descending order of weights. Iteratively, the next heaviest edge is considered. Two cases are possible: (1) if the edge connects a centroid to a noncentroid, the non-centroid is added to the cluster of the centroid (notice that at this point the noncentroid belongs to at least two clusters), (2) if the edge connects two centroids, the cluster of one centroid is assigned to the cluster of the other centroid (i.e. it is 'engulfed' by the other centroid), if and only if its weight is smaller than that of the other centroid.

Algorithm: CR ( )
1. Sort E in order of the edge weights
2. CentroidChange = true
3. index = 0
4. While (CentroidChange && index < N-1 && E is not empty)
5. CentroidChange = false
6. For each vertex v, take its edge evw connecting v to its next
closest neighbor w; i.e. w = v.neighbor [index]
7. Store these edges in S
8. Find the lowest edge weight in S, say *low*, and empty S
9. Take all edges from E whose weight >= *low*

Figure 3: Concurrent Rippling (CR)

The two clusters are merged and the smaller weight centroid becomes a non-centroid. The algorithm terminates when the centroids no longer change. Concurrent Rippling algorithm requires O (N2logN) complexity to sort the N-1 neighbors of the N vertices. It requires another O (N2logN) to sort the N2 number of edges. In the worst case, the algorithm has to iterate through all the N2 edges. Hence, in the worst case the complexity of the algorithm is O (N2logN).

5. APPLICATIONS

5.1 Document Clustering

We illustrate and evaluate the performance of our graph-based clustering algorithms for the offline and on-line document clustering task. Documents are vertices of the graph and edges are weighted with the inverse of the tf-idf cosine-similarity of the documents they connect. The graph is a clique. We evaluate the performance of our proposed algorithms empirically against other state of the arts clustering algorithms: Markov Clustering, Star Clustering, K-means,Single-link Hierarchical Clustering, and Zahn's MST Clustering. For Star clustering, by default and unless otherwise specified, we set the value of threshold σ for Star clustering to be the average similarity of documents in the given sub-collection

5.2 k-member Clustering

One of the premises of our work is the opportunity to easily adapt our MST-Sim clustering algorithms to related problems of local and dynamic nature such as k-member clustering. We illustrate and evaluate the performance of our MST-Sim algorithm for k-member clustering with the k-member greedy clustering algorithm proposed in [18], which has been experimentally shown to outperform Median Partitioning and K-Nearest Neighbor algorithms for k anonymization in terms of information loss. Records as vertices of the graph and edges are weighted with distance metric. a variant of *MST-Sim* that can be used to solve k-member clustering problem, called *MST-k*, which uses only one condition for merging, i.e. the condition that the size of the resulting merged cluster must be lesser than or equal to k. we see that all our algorithms are slower than k-member greedy algorithm on all k-values due to their pre-processing time to construct the graph and compute all pair wise similarities between records. graph-based clustering algorithms, the pre-processing time is a bottleneck for *MSTk.*When pre-processing time is not factored in, *MST-k* is much faster than k-member greedy clustering. In average, it is 94.4% faster when pre-processing time is not a factor. This further highlights the need for ways to reduce the pre-processing time of graph-based clustering algorithms.

5.3 Opinion-based Ranking

We illustrate and evaluate our method for ranking movies based on opinions in their reviews, by using reviews of box office movies written by users of a popular movie review site. We pick 50 movies randomly from box office list of November 2007 to February 2008. For each movie, we download all its users' reviews. For each movie we note its box office figure, its overall quantitative user rating, and its genre. The movies are of genre action, animation, children, comedy, drama, foreign film, horror, musical, romance, science fiction, chick flick, crime, political, or psycho. We use *Top-k* and *Granularity-g* method for evaluating performance. For each of the evaluation, we present metrics for measuring ranking performance. We also present interesting result for the ranking of adjectives by genre.

5.4 Time Series Correlation

We illustrate and evaluate the performance of our proposed method that computes time series correlation as similarity between their corresponding gradient sequences. We use correlation measures produced by our method to identify pairs of time series with high similarity in their gradient sequences. We conduct the evaluation on both synthetic and real time series data.

6. CONCLUSION

a variant of randomized MST algorithm that does not use Borůvka steps and selects any random tree from the graph to identify *heavy* edges. The algorithm incrementally finds the MST of the graph by building on the initial random tree. The algorithm is iterative and it is simpler in implementation, based purely on the cycle and cut property of graphs. MST-Sim that uses minimum spanning tree algorithm and adds to it intra- and inter cluster similarity metrics, which have basis in graph theory. MST-Sim algorithms are generally very fast and efficient in comparison with other state of the art graph-based clustering algorithms Ricochet that does not require any parameter to be defined a priori. While the fact that Ricochet is unconstrained is already an advantage, Ricochet algorithms are competitive to other state of the art clustering algorithms. Among our three proposed families of algorithms – variant of Star, MST-Sim, and Ricochet, MST-Sim is the most effective and efficient.

7. ACKNOWLEDGMENT

The author is very thankful to respected Dr.G.Vishnu Murthy, Head of the Department, Computer Science & Engineering, School of Engineering, Anurag Group of Institutions, Hyderabad, India. And to my parents who have always been very understanding and supportive both

financially and emotionally.

7. REFERENCES

[1] Kruskal J. B., On the shortest spanning subtree and the traveling salesman problem. In Proceedings of the American Mathematical Society. 7, pp. 48–50, 1956.
[2] Boruvka O., "O jistém problému minimálním (About a certain minimal problem)", Práce mor. prírodoved. spol. v Brne III, pp. 3: 37–58, 1926.
[3] Skiena S. S., The Algorithm Design Manual, Telos/Springer-Verlag, New York, 1998.
[4] Karger D.R., Klein P.N., and Tarjan R.E., A Randomized Linear-Time Algorithm to Find Minimum Spanning Trees, Journal of the ACM, Vol. 42, pp. 2:321-328, 1995.
[5] MacQueen J. B., Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. Berkeley, University of California Press, 1:281-297, 1967.
[6] Johnson S. C., Hierarchical Clustering Schemes. Psychometrika, 2:241-254, 1967.
[7] Zahn C.T., Graph Theoretical Methods for Detecting and Describing Gestalt Clusters. IEEE Transactions on Computers, Vol. C-20, No. 1, January 1971.
[8] Jain A.K., Murty M. N., Flynn, P.J., Data Clustering: A Review, ACM Computing Surveys,Vol. 31, No. 3, September 1999.
[9] Algorithm 479: A Minimal Spanning Tree Clustering Method [Z]. Communications of the ACM, Vol. 17, Issue 6, Pages: 321 – 323, June 1974.
[10] Karypis G., Han E., Kumar V., CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. IEEE Computer Vol. 32 No. 8, 68-75, 1999.
[11] Van Dongen S.M., Graph clustering by flow simulation - [S.l.]: [s.n.], 2000 - Tekst. -Proefschrift Universiteit Utrecht, 2000.
[12] Nieland H., Fast Graph Clustering Algorithm by Flow Simulation. Research and Development ERCIM News No. 42 - July 2000.
[13] Aslam J., Pelekhov K., and Rus D., the Star Clustering Algorithm. In Journal of Graph Algorithms and Applications, 8(1) 95–129, 2004.
[14] Motwani R. and Raghavan P., Randomized Algorithms, Cambridge University Press, 1995.
[15] King V., A Simpler Minimum Spanning Tree Verification Algorithm, Workshop on Algorithms and Data Structures, 1995.
[16] Aslam J., Pelekhov K., and Rus D., The Star Clustering Algorithm. In Journal of Graph Algorithms and Applications, 8(1) 95–129, 2004.
[17] Kleinberg J., Tardos E., Algorithm Design, Pearson Education Inc., New York, 2006.
[18] Byun J., Kamra A., Bertino E., and Li N., Efficient k-anonymity Using Clustering Techniques, Proceedings of International Conference on Database Systems for Advanced Applications (DASFAA), 2007.

AUTHORS BIOGRAPHY

Mr.B.Venkateshwar Reddy Received M.Sc Mathematics from Osmania University and M.E Computer Science and Engineering from Sathyabama University, Chennai. Presently working as a Assistant Professor in school of Engineering, Anurag Group of Institutions, Hyderabad, India. Published three papers in various National and International Conferences, Journals. His area of interest includes Data Mining, Machine Learning and Pattern Recognition.

Mrs.T.Asha Latha Received B.Tech Computer Science and Engineering from JNTUH. Pursuing M.Tech Computer Science and Engineering from JNTUH. Presently working as a Assistant Professor in school of Engineering, Anurag Group of Institutions, Hyderabad, India.