Fast Minimum Spanning Tree Based Clustering Algorithms on Local Neighborhood Graph

R. Jothi(✉), Sraban Kumar Mohanty, and Aparajita Ojha

Indian Institute of Information Technology, Design and Manufacturing Jabalpur, Madhya Pradesh, India
{r.jothi,sraban,aojha}@iiitdmj.ac.in
http://www.iiitdmj.ac.in

Abstract. Minimum spanning tree (MST) based clustering algorithms have been employed successfully to detect clusters of heterogeneous nature. Given a dataset of n random points, most of the MST-based clustering algorithms first generate a complete graph G of the dataset and then construct MST from G. The first step of the algorithm is the major bottleneck which takes $O(n^2)$ time. This paper proposes two algorithms namely MST-based clustering on K-means Graph and MST-based clustering on Bi-means Graph for reducing the computational overhead. The proposed algorithms make use of a centroid based nearest neighbor rule to generate a partition-based Local Neighborhood Graph (LNG). We prove that both the size and the computational time to construct the graph (LNG) is $O(n^{3/2})$, which is a $O(\sqrt{n})$ factor improvement over the traditional algorithms. The approximate MST is constructed from LNG in $O(n^{3/2} \lg n)$ time, which is asymptotically faster than $O(n^2)$. The advantage of the proposed algorithms is that they do not require any parameter setting which is a major issue in many of the nearest neighbor finding algorithms. Experimental results demonstrate that the computational time has been reduced significantly by maintaining the quality of the clusters obtained from the MST.

Keywords: Clustering · MST · K-means · Bi-means · Local neighborhood graph

1 Introduction

Graph-based clustering algorithms have been used extensively in cluster analysis due to their efficient functionality in a wide range of problem domains [1][2]. Graph clustering identify similar subgraphs based on the topological properties of the graph. Several graph construction methods have been widely studied in the context of clustering, to name a few [10][11]. The motivation of many of the graph learning methods is to obtain a robust and sparse affinity graph and applying clustering algorithms such as spectral clustering on the affinity graph.

In recent years, Minimum Spanning Tree (MST) based graph clustering algorithms have drawn much attention, as they are capable of identifying clusters irrespective of their shapes and sizes [8].

C.-L. Liu et al. (Eds.): GbRPR 2015, LNCS 9069, pp. 292–301, 2015.
DOI: 10.1007/978-3-319-18224-7_29

The MST-based clustering method comprises of the following steps [3]:

1. Given a set of points, compute pairwise dissimilarity matrix which defines the adjacency matrix of the graph G.
2. Construct MST of G.
3. Remove the inconsistent edges until the required number of connected components are found. Each resulting component corresponds to a cluster.

Most of the previous work on MST-based clustering algorithms focuses on improving the quality of the clusters [4][5][6][7][8]. But computational efficiency is also an important issue to be considered. Very few algorithms were proposed in this direction [9][12][13]. Wang et al. proposed a MST based clustering algorithm using a divide-and-conquer scheme [9]. Using cut and cycle properties, the long edges are identified at an early stage, so as to save distance computations. However, the worst-case complexity of the algorithm remains as $O(n^2)$.

A fast approximate MST algorithm was proposed by C.Zhong et al. in [12]. By dividing the dataset into k subsets using K-means algorithm and applying exact MST algorithm on each of these subsets separately, they obtain approximate MST in $O(n^{3/2})$ complexity. However, the actual running time of the algorithm and the accuracy of the MST mainly depend on the distribution of the partitions from K-means [12].

X.Chen proposed two graph clustering algorithms namely clustering based on a near neighbor graph (CNNG) and clustering based on a grid cell graph (CGCG) [13]. While CNNG algorithm construct MST based on δ-nearest neighbors, the CGCG algorithm determines the nearest neighbors by dividing the attribute space into grid cells. The worst-case complexity of these algorithms is $O(n^2)$ and they speed up the clustering process using multidimensional grid partition and index searches. But the algorithms are sensitive to the parameters such as δ-near neighbors and interval length in each dimension of grid cells [13].

Contribution. Most of the previous algorithms assume complete graph of the dataset [7]. If we could represent the underlying structure of the dataset using a local neighborhood graph instead of a complete graph, the cost of MST-based clustering algorithms can be reduced. With this motivation, we present two efficient algorithms to improve the run time overhead caused in clustering based on MST. A centroid based nearest neighbor rule is proposed in this paper for identifying the local neighborhood of the points. We show that the number of edges in the local neighborhood graph generated by the proposed algorithms as well as the cost of graph generation is $O(n^{3/2})$. Also the proposed algorithms do not require any parameters.

The rest of the paper is organized as follows. The details of the proposed algorithms are explained in Section 2. The complexity of the proposed algorithms is discussed in Section 3. The experimental analysis is shown in Section 4. The conclusion and future scope are given in Section 5.

2 Proposed Method

Let $X = \{x_1, x_2, \cdots, x_n\}$ be a dataset of n d-dimensional points. A weighted undirected graph $G = (V, E)$ is constructed from X, where the vertex set $V = \{X\}$ and the edge set $E = \{(x_i, x_j) \mid x_i, x_j \epsilon V\}$. The edges are weighted using Euclidean distance $d(x_i, x_j)$. In most of the previous algorithms, as all-pair edges are assumed, $|E| = n(n-1)/2$, resulting in a complete graph.

As choosing the edges for MST comes from local neighborhood principle, the points which are far apart are not connected by an edge in the MST. Hence an initial edge pruning strategy is required in order to rule out the longest edges which play no role in the construction of MST. Fig. 1 illustrates this.

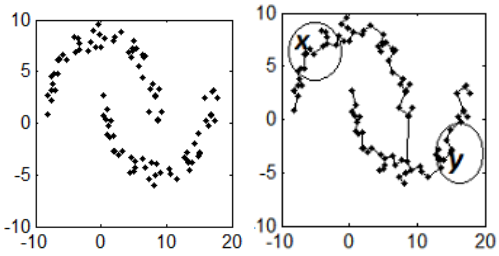


Fig. 1. MST connects points in the neighborhood:(a) A given dataset. (b) The points x and y are located far apart and the distance computation between x and y is not necessary.

In order to reduce the size of nearest neighbor search during MST construction, the proposed algorithms carry out a preliminary partition which gives us an intuition about the approximate nearest neighbors for each point. This saves much of the pair-wise distance computations and as a result the size of the graph is reduced from $O(n^2)$ to $O(n^{3/2})$. This is the key idea used in our proposed algorithms namely MST-based clustering on K-means Graph (KMGClust) and MST-based clustering on Bi-means Graph (BMGClust).

Let X be divided into a set of partitions $S = \{S_1, S_2, \cdots, S_k\}$, where k is the number of partitions. Let μ_i be the center of the partition S_i, where $1 \leq i \leq k$. Based on the partitions and their neighboring nature, we obtain local neighborhood of the points. A brute-force search on distance between centers of all pairs of partitions would easily reveal their adjacent nature. Let ω_{ij} be the distance between centers of a pair of partitions S_i and S_j. Let ω be the average distance between centers of all such pairs. ω is computed as follows.

$$\omega = \frac{1}{T_p} \sum_{S_i \epsilon S} \sum_{S_j \epsilon S} d(\mu_i, \mu_j)$$

where $T_p = k \times (k-1)/2$ and $d(a, b)$ denotes the Euclidean distance between a and b. The two partitions S_i and S_j are said to be neighbors iff $d(\mu_i, \mu_j) \leq \omega$.

With the above information, we define Local Neighborhood (LN) and Local Neighborhood Graph (LNG) as follows.

Definition 1. [Local Neighborhood] Consider a point $x_i \epsilon X$. Let S_i be the partition containing x_i. The local neighborhood of the point x_i is defined as:

$$LN(x_i) = \begin{cases} x_j, & \forall x_j \epsilon S_i \\ x_j, & \forall x_j \epsilon S_j, i \neq j,\ S_i\ \&\ S_j \text{ are neighboring partitions} \end{cases}$$

The above definition states that the points are likely to be in the neighborhood if either they belong to the the same partition or they fall in the boundary of the two partitions. Such boundary points are recognized using the Centroid Based Rule (CBR) which is stated as follows:

Definition 2. [Centroid Based Rule (CBR)] Let S_i and S_j be any two adjacent partitions with μ_{S_i} and μ_{S_j} as their centers respectively. Let x_i and x_j be any two points in the dataset X, such that $x_i \epsilon S_i$ and $x_j \epsilon S_j$. Let us define D_{ii}, D_{ij}, D_{jj} and D_{ji} as: $D_{ii} = d(x_i, \mu_{S_i})$; $D_{ij} = d(x_i, \mu_{S_j})$; $D_{jj} = d(x_j, \mu_{S_j})$; $D_{ji} = d(x_j, \mu_{S_i})$. Then, the points x_i and x_j are said to be boundary points iff $(D_{ij} \leq 2D_{ii})\ OR\ (D_{ji} \leq 2D_{jj})$.

Definition 3. [Local neighborhood graph] $G_{LN} = (V, E)$ is a weighted undirected graph, where $V = \{X\}$ and E is defined as follows:

$$E = \{(x_i, x_j) \mid x_i, x_j \epsilon V\ \&\ (x_i \epsilon LN(x_j)\ OR\ x_j \epsilon LN(x_i))\} \tag{1}$$

Once the local neighborhood graph G_{LN} is constructed with respect to the above definition, MST can be generated from this graph. As only the points in the closer proximity are considered in the edge set of G_{LN}, much of the distance computations are saved.

2.1 MST-Based Clustering on K-means Graph

The KMGClust algorithm is briefed as follows. First, the given dataset X is divided into k partitions using K-means algorithm, where k is set to $\sqrt{n}$ [7]. Then, the local neighborhood graph G_{LN} is obtained by considering the points in the neighboring partitions. Finally, the MST is constructed from $G_{LN} = (V, E)$.

We divide the edge set E of G_{LN} as intra-partition edges (E_{intra}) and inter-partition edges (E_{inter}). Let $S = \{S_1, S_2, \cdots, S_k\}$ be the set of partitions produced by K-means with μ_i as the center of each partition S_i , where $1 \leq i \leq k$. Each partition S_i is a complete subgraph and hence each pair of points within a partition S_i will be connected by an intra-partition edge. The inter-partition edges are computed only between the adjacent partitions so as to rule out the comparisons between partitions which lie significantly far apart.

Once the adjacent partitions are determined, the adjoining edges are computed between these partitions. The points which lie on the boundary of a partition S_i will likely to have nearest neighbors from the boundary points of another

Algorithm 1. *KMGClust Algorithm.*

Input: *Dataset X.*
Output: *Approximate MST of X.*

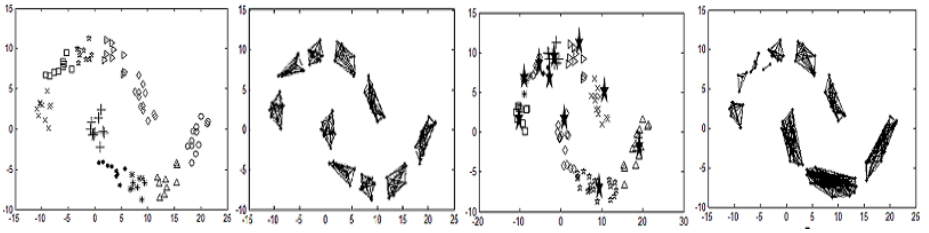
1 ***Partition Phase.***
1.1 Divide the dataset X using K-means.
1.2 Let S be the set of partitions with μ_i as their respective centroids.
2 ***MST Generation Phase***
2.1 Let $G_{LN} = (V, E)$ be the local neighborhood graph to be constructed, where $V = \{X\}$ and $E = \{\phi\}$.
2.2 Compute intra-partition all pair-edges as follows:
For each point x_j in subset S_i,
For each point x_k in subset S_i, where $j \neq k$,
Compute $d(x_j, x_k)$ and add this edge (x_j, x_k) to E.
2.3 Compute inter-partition edges based on CBR as follows:
For each pair of neighboring partitions S_i and S_j, where $S_i \neq S_j$,
For each pair of elements x_i and x_j, where $x_i \epsilon S_i$ and $x_j \epsilon S_j$,
Compute D_{ii}, D_{ij},D_{jj} and D_{ji} as follows:
$D_{ii} = d(x_i, \mu_{S_i})$; $D_{ij} = d(x_i, \mu_{S_j})$; $D_{jj} = d(x_j, \mu_a)$; $D_{ji} = d(x_j, \mu_b)$
If $(D_{ij} < 2D_{ii})$ || $(D_{ji} < 2D_{jj})$
Compute $d(x_i, x_j)$ and add this edge (x_i, x_j) to E.
2.4 Run Kruskal's algorithm on the graph G_{LN} to obtain MST.

partition S_j, where S_i and S_j are adjacent partitions. The boundary points are recognized using CBR. The details of the KMGClust algorithm is given in Algorithm 1.

Issues with KMGCLust Algorithm. The purpose of creating an initial partition of the dataset is to minimize the number of edges in the nearest neighbor search during MST construction phase. The purpose can be achieved only if the partitions are of approximately equal size and the inter-partition distance, the distance between centers of the two partitions, is maximized. K-means algorithm may not produce balanced partitions, as a consequence, the neighborhood graph G_{LN} tends to have more number of intra-partition edges. This is explained in Fig. 2.

2.2 MST-Based Clustering on Bi-means Graph

As the initial centers for K-means partition are chosen randomly, the two or more centers may collide in a nearest region. This is shown Fig. 2c with solid stars representing the center of the partitions. In an effort to maximize the distance



(a) (b) (c) (d)

Fig. 2. Relation between equilibrium of partition and size of graph: (a) Approximately equal sized partitions. (b) Subgraphs of the partitions in (a). (c) An unbalanced partition with colliding centers. (d) Subgraphs of the partitions in (c).

between the centers of two partitions, BMGClust algorithm makes use of Bi-means algorithm which is a recursive bi-partitioning of the dataset with two centers chosen in a wise manner.

The Bi-partitioning method recursively split the dataset into a binary tree of partitions with different partitioning criteria [14]. The Bi-means algorithm used in this paper makes use of bi-partitioning method but in a different manner. Partitioning of the dataset continues as long as the size of the subset to be partitioned is greater than $\sqrt{n}$. The subsets, which cannot be partitioned further, are stored in the set S. The Bi-means algorithm is explained in Algorithm 2.

The local neighborhood graph G_{LN} can be generated from the set of partitions S returned by Bi-means Algorithm. Then, approximate MST is constructed by applying Kruskal's algorithm on the graph G_{LN}. The complete steps involved in BMGClust algorithm is described in Algorithm 3.

In the context of generating local neighborhood graph, the Bi-means algorithm has few advantages over K-means algorithm. First, unlike K-means, the Bi-means algorithm does not require to preset the number of initial partitions. Second, as the centers chosen from each of partition level are widely separated, the chance of getting colliding centers is very less.

3 Theoretical Analysis

The number of edges in the local neighborhood graph $G_{LN} = (V, E)$ resulting from KMGClust and BMGClust is bounded by the relation $|E| \leq O(n^{3/2})$. Thus, the complexity of constructing MST by KMGClust algorithm is $O(n^{3/2} \lg n)$, considering the average case. The worst case of BMGClust is $O(n^{3/2} \lg n)$.

4 Experimental Results

In order to demonstrate the efficiency of our algorithms on various clustering problems, we consider four types of artificial datasets as described in Table 1.

Algorithm 2. *Bi-means Algorithm.*

Input: *Dataset X.*
Output: *Binary partition tree of X.*

1 $n' = |X|$.
2 If $n' > \sqrt{n}$
  2.1 Find the center μ of the dataset X.
  2.2 Choose a point $o \epsilon X$ such that $d(o, \mu)$ is minimum.
  2.3 Compute the distance from o to all other points.
  2.4 Choose two centers $p \epsilon X$ and $q \epsilon X$ such that p is farthest from o and q is farthest from p.
  2.5 Split the dataset X into two subsets X_p and X_q according to centers p and q.
  2.6 Bi-partitioning(X_p).
  2.7 Bi-partitioning(X_q).
3 else Append X to the table S.
4 return S.

Algorithm 3. *BMGClust Algorithm.*

Input: *Dataset X.*
Output: *Approximate MST of X.*

1 Let S be the set of partitions returned by Bi-means algorithm (ref. Algorithm 2).
2 Find Intra-partition edges as in KMGClust Algorithm (ref. Algorithm 1).
3 Find Inter-partition edges as in KMGClust Algorithm (ref. Algorithm 1).
4 Run Kruskal's algorithm on the graph G_{LN} to obtain MST.

Experiments were conducted on a computer with an Intel Core2 Duo Processor 2GHz CPU and 2GB memory running Ubuntu Linux. We have implemented all the algorithms on C++.

The efficiency of the proposed algorithms are assessed in terms of size of the graph they generate, time required to construct MST and the validity of the clusters obtained from the MST. For the sake of clarity, let us denote the complete graph as CG and the clustering algorithm on the MST of CG as CGClust. Also We denote the local neighborhood graphs generated by our proposed algorithms KMGClust and BMGclust as KMLNG(K-means local neighborhood graph) and BMLNG(Bi-means local neighborhood graph) respectively.

Table 2 demonstrate the experimental results of the proposed algorithms on the datasets mentioned in Table 1. Fig.3 illustrates the comparison of time to construct MST ($Time$) from CGClust, KMGClust and BMGClust. In case of KMGClust and BMGClust algorithms, $Time$ includes both the time spent in initial partitioning and the time spent in constructing MST. It is clear from the

Table 1. Details of the Dataset: No. of points (n), No. of clusters (k)

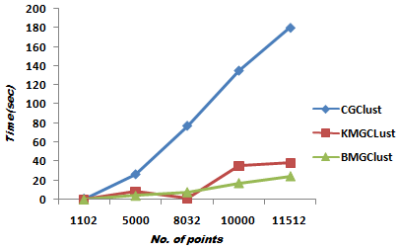
Dataset	n	k	Description
Data11	5000	15	Gaussian clusters
Data12	5000	15	
Data21	11522	2	Half-moon clusters
Data22	11522	2	
Data31	5000	2	Cluster inside another cluster
Data32	10106	2	
Data41	10000	2	Well-separated clusters
Data42	10000	2	

Table 2. Comparison of size of the graph ($|E|$), $Time$ to construct MST and $Weight$ of MST on different datasets

Dataset	Method	$\|E\|$	$Time$	$Weight$
Data11	CGClust	12497500	25.32	2.3430e+07
	KMGClust	824130	4.77	2.3859e+07
	BMGClust	782189	4.33	2.3430e+07
Data12	CGClust	12497500	26.77	2.7858e+07
	KMGClust	891126	15.15	2.7952e+07
	BMGClust	842218	5.13	2.7858e+07
Data21	CGClust	66372481	180.3	1026.42
	KMGClust	4783583	38.22	1027.2
	BMGClust	1933932	24.61	1026.44
Data22	CGClust	66372481	216.9	1118.45
	KMGClust	1204878	43.72	1193
	BMGClust	1893911	26.83	1113.21
Data31	CGClust	12497500	26.42	156.633
	KMGClust	607361	5.72	156.634
	BMGClust	447691	2.87	156.633
Data32	CGClust	51060565	133.76	158.017
	KMGClust	1692949	36.11	158.018
	BMGClust	1341682	11.42	158.017
Data41	CGClust	49995000	135.28	10501.8
	KMGClust	4792022	35.63	13782.2
	BMGClust	2845821	17.41	10412.9
Data42	CGClust	49995000	134.54	9711.24
	KMGClust	6118279	36.94	13501
	BMGClust	1581607	16.76	9459.83

figure that the running time of the proposed algorithms is significantly smaller as compared to CGClust.

In order to demonstrate that the local neighborhood graphs KMLNG and BMLNG will not miss any information that is significant for clustering, we test the performance of the clustering results on the approximate MST obtanied from KMLNG and BMLNG using Zahn's clustering algorithm [3]. The results



(a)

Fig. 3. Comparison of proposed algorithms

Table 3. Comparison of quality indices of clusters obtained from proposed methods with CGClust on synthetic datasets

Dataset	Method	Rand	Jaccard	FM	ARand
Data11	CGClust	0.87565	0.34796	0.58910	0.46329
	KMGClust	0.90200	0.40371	0.63461	0.53074
	BMGClust	0.87565	0.34796	0.58910	0.46329
Data21	CGClust	1.0000	1.0000	1.0000	1.0000
	KMGClust	1.0000	1.0000	1.0000	1.0000
	BMGClust	1.0000	1.0000	1.0000	1.0000
Data31	CGClust	1.0000	1.0000	1.0000	1.0000
	KMGClust	1.0000	1.0000	1.0000	1.0000
	BMGClust	1.0000	1.0000	1.0000	1.0000
Data41	CGClust	0.83962	0.55453	0.74447	0.61432
	KMGClust	0.81266	0.48295	0.67378	0.53446
	BMGClust	0.83970	0.55465	0.74455	0.61447

are validated using the external quality indices such as Rand, FM, Jaccard and Adjusted Rand [7]. Table 3 shows the quality indices of clustering on various datasets.

5 Conclusion

This paper proposed two efficient algorithms namely KMGClust and BMGClust for obtaining MST in the context of clustering in a less than quadratic time. The theocratical analysis proved that the size of the graph generated by the proposed algorithms is bounded by $O(n^{3/2})$ and thus the time for constructing MST has been reduced, while maintaining the quality of the clusters. Experimental results on various datasets demonstrated the efficiency of the proposed algorithms. As a future work, we will carry out an extensive analysis of local neighborhood graph generated by our proposed algorithms on various graph clustering algorithms.

References

1. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys (CSUR) 31(3), 264–323 (1999)
2. Schaeffer, S.E.: Graph clustering. Computer Science Review 1(1), 27–64 (2007)
3. Zahn, C.T.: Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Transactions on Computers 100(1), 68–86 (1971)
4. Xu, Y., Olman, V., Xu, D.: Minimum spanning trees for gene expression data clustering. GENOME INFORMATICS SERIES, pp. 24–33 (2001)
5. Laszlo, M., Mukherjee, S.: Minimum spanning tree partitioning algorithm for microaggregation. IEEE Transactions on Knowledge and Data Engineering 17(7), 902–911 (2005)
6. Luo, T., Zhong, C.: A neighborhood density estimation clustering algorithm based on minimum spanning tree. In: Yu, J., Greco, S., Lingras, P., Wang, G., Skowron, A. (eds.) RSKT 2010. LNCS, vol. 6401, pp. 557–565. Springer, Heidelberg (2010)
7. Zhong, C., Miao, D., Fränti, P.: Minimum spanning tree based split-and-merge: A hierarchical clustering method. Information Sciences 181(16), 3397–3410 (2011)
8. Wang, X., Wang, X.L., Chen, C., Wilkes, D.M.: Enhancing minimum spanning tree-based clustering by removing density-based outliers. Digital Signal Processing 23(5), 1523–1538 (2013)
9. Wang, X., Wang, X., Wilkes, D.M.: A divide-and-conquer approach for minimum spanning tree-based clustering. IEEE Transactions on Knowledge and Data Engineering 21(7), 945–958 (2009)
10. Cheng, B., Yang, J., Yan, S., Fu, Y., Huang, T.S.: Learning with L1-graph for image analysis. IEEE Transactions on Image Processing 19(4), 858–866 (2010)
11. Liu, H., Yan, S.: Robust graph mode seeking by graph shift. In: Proceedings of the 27th International Conference on Machine Learning (ICML 2010), pp. 671–678 (2010)
12. Zhong, C., Malinen, M., Miao, D., Fränti, P.: Fast approximate minimum spanning tree algorithm based on *K*-means. In: Wilson, R., Hancock, E., Bors, A., Smith, W. (eds.) CAIP 2013, Part I. LNCS, vol. 8047, pp. 262–269. Springer, Heidelberg (2013)
13. Chen, X.: Clustering based on a near neighbor graph and a grid cell graph. Journal of Intelligent Information Systems 40(3), 529–554 (2013)
14. Chavent, M., Lechevallier, Y., Briant, O.: DIVCLUS-T: A monothetic divisive hierarchical clustering method. Computational Statistics and Data Analysis 52(2), 687–701 (2007)