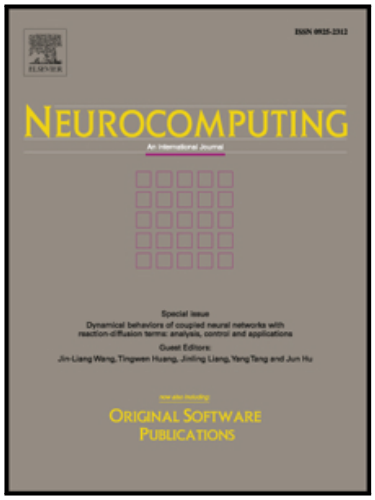
Communicated by Hung-Yuan Chung

Accepted Manuscript



Fast approximate minimum spanning tree based clustering algorithm

R. Jothi, Sraban Kumar Mohanty, Aparajita Ojha

PII: S0925-2312(17)31295-X
DOI: 10.1016/j.neucom.2017.07.038
Reference: NEUCOM 18726

To appear in: *Neurocomputing*

Received date: 25 April 2016
Revised date: 1 May 2017
Accepted date: 16 July 2017

Please cite this article as: R. Jothi, Sraban Kumar Mohanty, Aparajita Ojha, Fast approximate minimum spanning tree based clustering algorithm, *Neurocomputing* (2017), doi: 10.1016/j.neucom.2017.07.038

Highlights

- A fast approximate minimum tree based clustering algorithm is proposed
- A novel centroid based nearest neighbor rule is presented
- Much faster as compared to exact MST algorithms
- Experiments on both synthetic and real datasets are carried out
- Quality of clusters from the approximate MST is maintained

Fast approximate minimum spanning tree based clustering algorithm

R.Jothi*, Sraban Kumar Mohanty, Aparajita Ojha

Indian Institute of Information Technology, Design and Manufacturing Jabalpur, Madhya Pradesh, India.

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 an algorithm namely MST-based Clustering on Partition-based nearest neighbor graph for reducing the computational overhead. By using a centroid based nearest neighbor rule, the proposed algorithm first generates a sparse Local Neighborhood Graph (*LNG*) and then the approximate MST is constructed from *LNG*. We prove that both size and 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 $O(n^2)$. Experimental analysis on both synthetic and real datasets demonstrates that the computational time has been reduced significantly by maintaining the quality of clusters obtained from the approximate MST.

Keywords: Clustering, Minimum Spanning Tree, Local Neighborhood Graph

1. Introduction

Clustering is an unsupervised learning technique that is predominantly used for various exploratory analysis such as data mining and pattern recognition. There are a number of clustering methods and the traditional methods such as hierarchical and partitional methods lack in the ability to detect clusters which are not defined by regular geometric curves [1, 2].

Detecting clusters of non-convex shapes is becoming a research interest in recent years. Specifically, clustering algorithms that use Minimum Spanning Tree (MST) are drawing much attention in recent years [3, 4, 5, 6, 7, 8, 9, 10]. Given a weighted, undirected graph, MST is a connected acyclic subgraph such that the total weight of the tree is minimized. MST is a well-known combinatorial optimization problem which has been widely used in many practical applications such as network design, image segmentation, cluster analysis, etc.

MST-based clustering algorithm was initially proposed by Zahn [11]. Basic idea is to identify and remove the inconsistent edges from the MST until required number of clusters are formed. Key advantage of this type of clustering algorithms is that the points of a cluster are neither grouped around cluster center nor separated by a geometric boundary. Thus performance of these clustering algorithms do not depend on the shape of clusters and they can detect clusters of irregular shapes.

Efficient construction of MST and identification of inconsistent edges to partition the MST to get desired clusters are the two major issues in MST-based clustering algorithms. Most of the previous work on MST-based clustering algorithms focus on devising an objective function to find the optimal partition of the tree [3, 5, 6, 7, 8, 9]. But the computational efficiency is also an important issue to be considered. Very few algorithms were proposed in this direction [12, 13, 14].

Overall time complexity of the MST-based clustering algorithm is dominated by the construction of MST. Given a dataset of n random points, most of the previous approaches assume a complete graph of the given dataset [6, 12]. Thus the time complexity of constructing MST turns out to be $O(n^2)$. This time factor lim-

*Corresponding author: r.jothi@iiitdmj.ac.in Tel.: +91-761-2632273;

Email addresses: r.jothi@iiitdmj.ac.in (R.Jothi), sraban@iiitdmj.ac.in (Sraban Kumar Mohanty), aojha@iiitdmj.ac.in (Aparajita Ojha)

its the application of MST-based clustering algorithms to massive datasets. The major bottleneck here is the cost of nearest neighbor search. As every pair of points is associated with an edge, the cost of finding nearest neighbor of a point is $O(n)$ and the total cost for all the n points turns out to be $O(n^2)$. If we could represent the underlying structure of the dataset using a local neighborhood graph instead of a complete graph, we can minimize the cost of these algorithms to some extent.

With this motivation, we propose a method to speed up the minimum spanning tree based graph clustering. Key contribution of the paper is given as follows. We propose a method to identify the dense-subgraphs through preliminary partition of the dataset. A novel centroid-based rule is introduced to determine the neighboring relationship of the partitions. We give a formulation of partition-based neighborhood graph using the subgraphs and their proximity relation. The graph constructed by the proposed algorithm depicts the structure of a dataset through tight dense-subgraphs loosely coupled with their neighbors in the closer proximity. We call this graph as local neighborhood graph *LNG*. We show that the number of edges in *LNG* as well as the cost of graph generation is $O(n^{3/2})$. The proposed algorithm does not require any user defined parameters.

Experimental analysis on synthetic as well as real datasets reveals that the runtime of approximate MST construction is reduced significantly. The proposed algorithm outperforms the exact MST algorithms by a larger margin. The degree to which the approximate MST matches with the exact MST is shown using edge-error and weight-error analysis. By carrying out the cluster quality analysis, we show that the approximate MST generated by the proposed algorithm does not miss any information which is essential for clustering.

The paper is organized as follows. In Section 2, an overview of existing MST algorithms is presented with particular attention given to the cluster analysis. Our contribution to the fast approximate MST construction for cluster analysis is described in Section 3. The results of experimental validation are reported and discussed in Section 4, and Section 5 presents the conclusion.

2. Related work

The classical algorithms for constructing MST are Boruvka's, Prim's and Kruskal's algorithms [15]. Boruvka's algorithm computes the MST through successive applications of edge-contraction on an input graph. It starts with each vertex of the graph being a tree. For each tree, it selects the lightest edge connecting the tree to rest of the graph and a new tree is formed with the union of the connectivity of the incident vertices of the chosen edge. This edge contraction process continues until all the trees are connected. Computational complexity of the algorithm is $O(E \log V)$, where V is the set of vertices and E is the set of edges [15].

Kruskal's algorithm first sorts the edges in non-decreasing order by their weights. It starts with every vertex being its own tree and iteratively combines the trees by adding edges in the sorted order, discarding the edges whose addition results in cycle, until all the vertices are connected by the tree. The complexity of the algorithm is $O(E \log V)$ [15].

The Prim's algorithm starts from an arbitrary root vertex as the tree and then repeatedly adds the shortest edge that connects a new vertex to the tree until the tree connects all the vertices. The running time of the Prims algorithm is $O(E \log V)$. If Fibonacci heap is employed, the complexity of the algorithm reduces to $O(E + V \log V)$ [15].

Several fast MST algorithms have been studied in the literature. Cheriton and Tarjan presented a fast MST algorithm which runs in $O(E \log \log V)$ time [16]. Using Fibonacci heap for the implementation of priority queue, Fredman and Tarjan proposed an MST algorithm with the complexity $O(E\beta(E, V))$, where $\beta(E, V) = \min\{i | \log^{(i)} V \leq E/V\}$ [17].

In case of cluster analysis, a set X of n random points are given in d dimensional space. A weighted undirected graph $G = (V, E)$ is constructed from X, where each node in the graph corresponds to a data point and an edge between two nodes mark the dissimilarity (similarity) of the points. As all pair-edges are assumed, the resulting graph is a complete graph with $|E| = O(n^2)$. The worst-case complexity of constructing MST from the complete graph is also $O(n^2)$. Such a quadratic complexity limits the application of classical MST algorithms for cluster analysis of very large datasets.

Wang et al. [12] proposed a fast MST-inspired clustering algorithm based on a divide-and-conquer scheme. 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 $O(n^2)$.

Chen [13] proposed two graph clustering algorithms namely clustering based on a near neighbor graph (CNNG) and clustering based on a grid cell graph (CGCG). While CNNG algorithm constructs 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].

A fast approximate MST algorithm was proposed by Zhong et al. [14]. 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, actual running time of the algorithm and accuracy of the MST mainly depend on the distribution of partitions from K-means [14].

Several fast nearest neighbor techniques have been studied in the recent years [18, 19]. Kd-tree is one of the most popular partitioning tree used for minimizing the cost of nearest neighbor search. However Kd-tree is effective in lower dimensional spaces and its performance degrades with the increase in dimensions [12, 19]. Hashing based nearest neighbor search has also been studied in the literature [20]. A fast approximate nearest neighbor search algorithm using Error Weighted Hashing (EWH) was proposed by [20]. They showed that EWH was ten times faster than locality sensitive hashing (LSH) based nearest neighbor search. The performance of hashing methods is sensitive to the quality of the hashing functions [19].

3. Partition based Nearest Neighbor Graph

Let $X = \{x_1, x_2, \cdots, x_n\}$ be a dataset of n d-dimensional points. Most of the graph-based clustering algorithms construct a weighted undirected graph $G = (V, E)$ from X, where the vertex set $V = \{X\}$ and the edge set $E = \{(x_i, x_j) \mid x_i \epsilon V, x_j \epsilon V\}$. The edges are weighted using a distance measure, for e.g., Euclidean distance. 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. This is illustrated in Fig. 1. A sample dataset and its MST are shown in Fig. 1a and Fig. 1b respectively. As MST reflects structure of the dataset, removing the longest edge partitions the dataset into two clusters. It is worthwhile to note that the edges between points which are significantly far apart neither play a role in the construction of MST nor during clustering stage as shown in Fig. 1c. 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. This saves the cost of nearest neighbor search to certain extent. In order to achieve this, the proposed algorithm carries out a preliminary partition which gives us an intuition about the approximate nearest neighbors for each point. The preliminary partition of the dataset shown in Fig. 1a is given in Fig. 1d. It is clear from the figure that, the nearest neighbors of a point can be determined from the locality of the points with respect to the group in which they are present. 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 of the proposed algorithm whose preliminary version has been presented in a conference [21]. The summary of the proposed algorithm is given as follows.

1 Divide the dataset into k partitions.
2 Compute intra-partition edges considering each partition as a complete subgraph (Clique).
3 Identify neighboring partitions as well as the boundary points in the neighboring partitions.
4 Compute inter-partition edges between the neighboring partitions using the boundary points.
5 Construct approximate MST from the resulting graph.

3.1. Some Definitions

Let X be divided into a set of preliminary partitions $S = \{S_1, S_2, \cdots, S_k\}$. Points within a partition are considered to be neighbors. In order to obtain the neighbors of points among the different groups, we should determine the adjacent nature of the partitions. A brute-force search on all-pair distances between centers of the partitions would easily reveal their neighboring nature. The following definition is used to determine the neighboring partitions among all pairs of partitions.

Definition 1[Neighboring partitions] Let $S = \{S_1, S_2, \cdots, S_k\}$ be the set of preliminary partitions generated by the proposed algorithm. Let μ_i be the center of the partition S_i and let T_μ be the MST constructed from the centers of the partitions. Two partitions S_i and S_j are said to be neighbors iff the following is true.

$$d(\mu_i, \mu_j) \leq \max_{e_i \epsilon E(T_\mu)} \{w(e_i)\}$$

where $w(e_i)$ denote the weight of edge e_i in the MST T_μ.

Suppose two partitions S_i and S_j are determined to be neighbors according to definition 1. Then the points which lie in the boundary of S_i and S_j are likely to have nearest neighbors. Such boundary points are

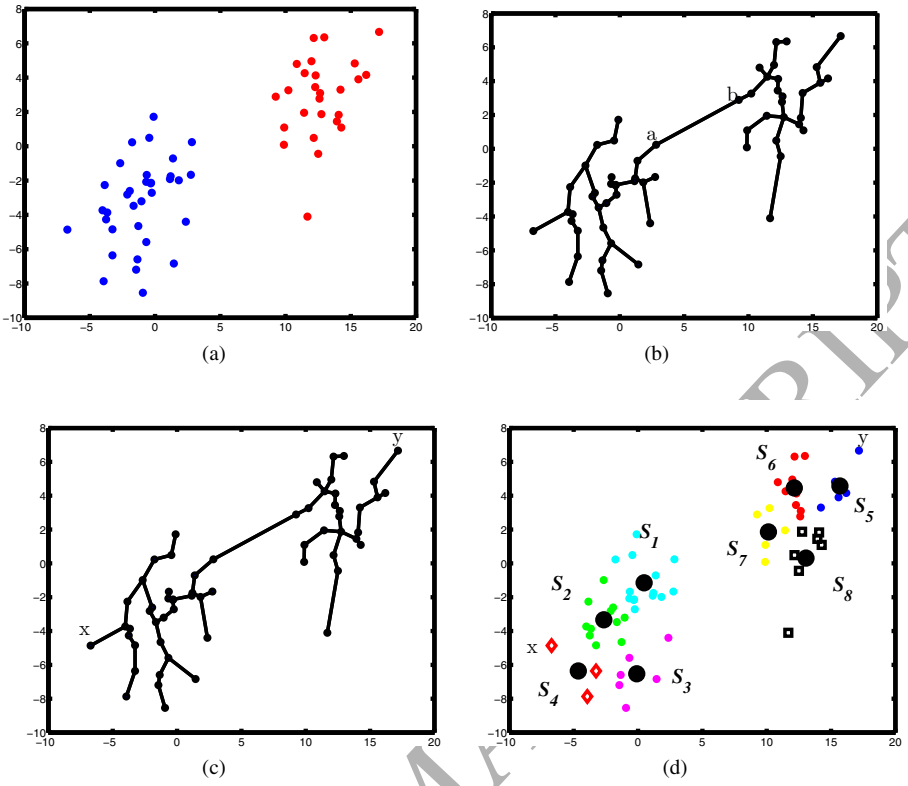


Fig. 1. Early detection of longest edges using Preliminary partitions: (a) An example dataset with two clusters. (b) MST constructed from complete graph of the dataset, removing the longest edge (a,b) results in desired clusters. (c) Points x and y are far apart, edge (x, y) is not included in MST and hence distance computation between x and y play no role in clustering phase. (d) Preliminary partitions of the dataset $S = \{S_1, S_2, \cdots, S_8\}$, center of each S_i marked in black solid dot; the nearest neighbors of x belong to the subsets S_2 and S_3, and hence distance computation between x and y is avoided.

recognized using the following definition.

Definition 2 [Centroid Based Rule (CBR)] Let S_i and S_j be any two neighboring partitions with μ_i and μ_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: $D_{ii} = d(x_i, \mu_i)$; $D_{ij} = d(x_i, \mu_j)$; $D_{jj} = d(x_j, \mu_j)$; $D_{ji} = d(x_j, \mu_i)$. Then, the points x_i and x_j are said to be boundary points iff $(D_{ij} \leq 2D_{ii})$ OR $(D_{ji} \leq 2D_{jj})$.

Fig. 2 illustrates the use of CBR to identify the points which lie in the boundary of neighboring partitions.

With respect to the partitions, we define Local Neighborhood (LN) and Local Neighborhood Graph (LNG) as follows.

Definition 3 [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 same partition or they fall in the boundary of the two partitions.

Definition 4 [Local neighborhood graph] $G_{LN} = (V, E)$ is a weighted undirected graph, where $V = \{X\}$ and E is defined as follows:

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

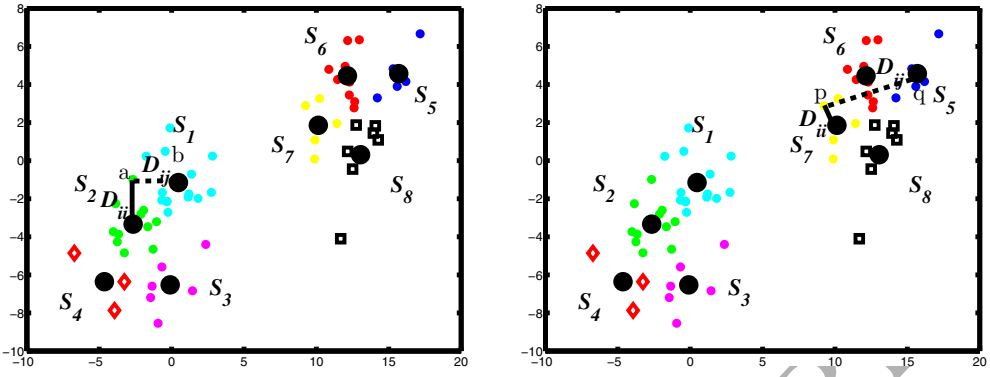


Fig. 2. Centroid Based Rule (CBR): (a) For points a and b, ($D_{ij} \leq 2D_{ii}$) and the edge (a, b) is considered during MST construction. (b) Points p and q lie far apart ($D_{ij} \neq 2D_{ii}$) and the edge (p, q) is not necessary for MST construction.

Once the neighborhood graph 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.

The proposed algorithm is named as Partition based Nearest Neighbor Graph (PNNG), as it generates the local neighborhood graph G_{LN} using the preliminary partitions. Edge set E of G_{LN} is divided as intra-partition edges (E_{intra}) and inter-partition edges (E_{inter}). 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 neighboring partitions so as to rule out the comparisons between partitions which lie significantly far apart.

The details of the proposed algorithm PNNG is given in Algorithm 1.

3.2. Impact of partitioning phase

Partitioning step plays a key role in the proposed algorithm. The efficiency of graph construction phase depends on the distribution of points among the partitions. We discuss K-means based partitioning and its issues in the Section 3.2.1. Then, we discuss an improved algorithm in the Section 3.2.2.

3.2.1. K-means based partitioning

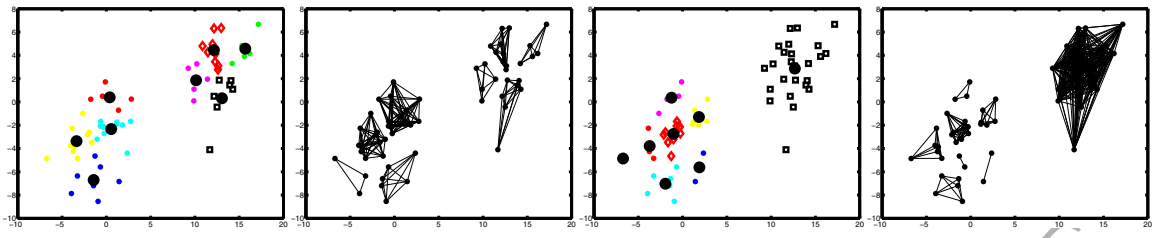
K-means algorithm is a most popular partitioning method which divides the number of points into k groups, where k is the actual number of clusters in the dataset. In order to obtain the preliminary partitions, we apply K-means algorithm on the given dataset. The value of k is set to $\sqrt{n}$ based on the assumption that the number of clusters in a dataset is smaller than the square root of the number of points in the dataset [22]. 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.

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. The initial centers for K-means partitions are chosen randomly, the two or more centers may collide in a nearest region. This leads to imbalanced partitions [23]. As a consequence, the neighborhood graph G_{LN} tends to have more number of intra-partition edges. This is explained in Fig. 3.

3.2.2. Bi-means based partitioning

In an effort to maximize the distance between the centers of two partitions, the proposed algorithm makes use of Bi-means algorithm which is a form of hierarchical divisive clustering (HDC). Generally HDC starts from the root node containing all the points and recursively split the node meeting certain criteria (for eg., maximum variance) into sub-clusters. This splitting continues until required number of clusters are obtained [24].

The Bi-means algorithm used in this paper makes use of HDC, but in a different manner. For each split, two farthest points in the node are chosen to maximize the distance between the centers of two partitions. Split-



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

Fig. 3. 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).

ting procedure continues as long as the size of the node to be partitioned is greater than $\sqrt{n}$. The partitions returned by the algorithm is maintained in a binary tree called as Bi-means tree (BT) as shown in Fig. 4. The subsets, which cannot be partitioned further, are marked as leaf nodes. Let S denotes the set of all leaf nodes from the tree BT. In Fig. 4, all the leaf nodes are marked in square and each leaf node in the tree corresponds to a partition S_i.

The Bi-means algorithm is explained in Algorithm 2. 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 the partition level are widely separated, the chance of getting colliding centers is very less. This leads to reduced number of edges in local neighborhood graph.

3.3. Criteria for determining the neighboring partitions

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 μ_i and μ_j be the centers of S_i and S_j respectively. Let ω be the average distance between centers of all such pairs. ω is computed as follows.

$$\omega = \frac{1}{T_p} \sum_{i=1}^{k} \sum_{j=i+1}^{k} d(\mu_i, \mu_j)$$

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 $d(\mu_i, \mu_j) \leq \omega$.

We have used ω as the threshold for determining the neighboring partitions in the preliminary version of this work [21]. However, it was observed that the calculation of the average distance between centers makes sense only in the context of quite homogeneous distribution of distances between points. Indeed, the average distance has no sense when the points distribution is quite heterogeneous (non Gaussian distribution). In order to address this issue, we use weight of the longest edge in the MST of centers T_μ as given in Definition 1. T_μ captures the neighboring relation of the partitions irrespective of the distribution of the points. And hence the proposed algorithm is expected to perform well on both Gaussian and non Gaussian distribution. In case of well-separated clusters, partitions belonging to different clusters do not satisfy CBR and thus it results in disconnected graph. To ensure connectivity of G_{LN} so that a spanning tree can be generated, edges in T_μ are included in the edgeset of G_{LN}. Center μ_i of the partition S_i may or may not be a point in X; in that case, the point $y \in S_i$ nearest to μ_i is chosen. (see Section 5.5 on empirical validation of distance threshold ω.)

4. Performance Analysis

In this section, we provide a theoretical analysis of the proposed algorithm PNNG. For the sake of clarity, PNNG with K-means based partitions is referred as KNNG and PNNG with Bi-means based partitions is referred as BNNG. Similarly the exact MST generated from the complete graph is referred as CG.

First, we derive a theoretical bound on the number of edges in the local neighborhood graph $G_{LN} = (V, E)$ generated by the algorithms KNNG and BNNG. Then, we obtain the complexity of generating MST from G_{LN}.

4.1. Size of the local neighborhood graph

Intra-Partition Edges Let us denote the intra-partition edge set of G_{LN} as E_{intra} and the inter-partition edge set of G_{LN} as E_{inter}. As each partition is a complete

Algorithm 1. *PNNG Algorithm.*

Input: *Dataset X.*

Output: *Approximate MST of X.*

1 ***Partition Phase***.
  1.1 *Divide the dataset X into k preliminary partitions.*
  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:
        $D_{ii} = d(x_i, \mu_i)$; $D_{ij} = d(x_i, \mu_j)$; $D_{jj} = d(x_j, \mu_j)$; $D_{ji} = d(x_j, \mu_i)$
        If $(D_{ij} < 2D_{ii}) \parallel (D_{ji} < 2D_{jj})$
          Compute $d(x_i, x_j)$ and add this edge (x_i, x_j) to E.
  2.4 *Run MST algorithm such as Prims or Kruskal's algorithm on the graph G_{LN} to obtain approximate MST.*

subgraph, the number of intra-partition edges in G_{LN} is given as follows:

$$E_{intra} = \sum_{1 \leq i \leq k} \binom{|S_i|}{2}$$

In case of KNNG algorithm, the number of partitions $k = \sqrt{n}$. Hence size of each of the partitions S_i is $\sqrt{n}$ (assuming K-means results in balanced partitions).

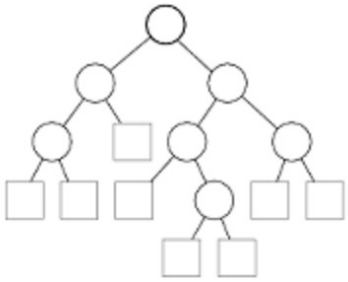


Fig. 4. Binary Tree representing the hierarchy of partitions obtained from Bi-means algorithm. The collection of leaf nodes marked as square correspond to the set of partitions returned by the Bi-means algorithm.

Then,

$$E_{intra} = \sqrt{n} \times \binom{\sqrt{n}}{2} = O(n^{3/2})$$

In case of BNNG algorithm, the value of k is not preset. Bi-means algorithm will recursively partition the dataset so that during every iteration the two centers of the partitions are maximum spaced apart. Any node in the Bi-means tree is divided recursively, if its size is greater than $\sqrt{n}$. Otherwise it will become a leaf node which corresponds to a partition S_i. The height of the Bi-means tree is $\log \sqrt{n} = O(\log n)$. Using the property that the number of leaves in a binary tree of height h is at most 2^h, the total number of partitions (k) is calculated as $O(\sqrt{n})$. In practice, $k = c\sqrt{n}$, where c is a constant. Thus, the size of each of the partitions from Bi-means $S_i = \sqrt{n}/c$.

$$E_{intra} = c\sqrt{n} \times \binom{\frac{n}{c\sqrt{n}}}{2} = O(n^{3/2})$$

Inter-Partition Edges Determination of inter-partition edges in the neighborhood graph requires us to compute (i) the number of neighboring partitions among all pairs of partitions; (ii) the number of points which lie on the boundary of two neighboring partitions.

According to definition 1, two partitions are said to be neighbors if the distance between their centers is less than the length of the longest edge in T_μ. The edges in the center MST T_μ can be used to specify the adjacent nature of partitions. If two partitions have an edge in T_μ, then they may be regarded as neighbors. Thus the total number of neighboring partitions is at most $\sqrt{n} - 1$.

As T_μ is constructed considering only the centers, we may miss out the neighboring partitions which are not connected in T_μ but yet they are likely to be

Algorithm 2. *Bi-means Algorithm.*

Input: *Dataset X.*

Output: *Binary partition tree BT of X.*

1 $n' = |X|$.
2 If $n' > \sqrt{n}$
  2.1 Find the center μ *of the node representing 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 node X into two subtrees X_p *and* X_q *according to centers p and q.*
  2.6 Bi-means(X_p).
  2.7 Bi-means(X_q).
3 else mark X as a leaf node.
4 Return Binary partition tree BT.

neighbors by the points in the partitions. In order to cover all those cases, we find all-pair distances between the centers. As there are $\sqrt{n}$ partitions, the total neighboring pairs of partitions is $O(n)$. On an average, the number of neighboring partitions among all pairs of partitions is at most $k/2$. If S_i and S_j are neighboring partitions then the number of points which lie on the boundary of two partitions is at most $(|S_i| + |S_j|)/2$.

For KNNG algorithm,

$$E_{inter} = \frac{k}{2} \times \binom{(\frac{|S_i|}{2} + \frac{|S_j|}{2})}{2} = \frac{\sqrt{n}}{2} \times \binom{\sqrt{n}}{2} = O(n^{3/2})$$

For BNNG algorithm, as the centers of the partitions are maximum separated, the number of boundary points that lie between adjacent partitions will be a constant time smaller as compared to KNNG.

$$E_{inter} = \frac{k}{2} \times \binom{(\frac{|S_i|}{2} + \frac{|S_j|}{2})}{2} = c\sqrt{n} \times \binom{\frac{n}{c\sqrt{n}}}{2} = O(n^{3/2})$$

To summarize, the number of edges in the local neighborhood graph $G_{LN} = (V, E)$ resulting from both KNNG and BNNG is bounded by the following relation:

$$|E| \leq O(n^{3/2})$$

4.2. Time complexity of generating MST

The overall time complexity of generating MST from $G_{LN} = (V, E)$ as follows:

$$T(n) = T_p + T_{intra} + T_{inter} + T_{MST}$$

where T_p is the time required to partition the data; T_{intra} is the time required to compute intra-partition edges; T_{inter} is the time needed to find the inter-partition edges; T_{MST} is the time required to construct MST from G_{LN}.

For KNNG algorithm,

$$\begin{aligned} T(n) &= O(n^{3/2}) + O(n^{3/2}) + O(n^{3/2}) + O(n^{3/2} \lg n) \\ &= O(n^{3/2} \lg n) \end{aligned}$$

For BNNG algorithm, Bi-means partitioning is carried out in $O(n \log n)$. The Bi-means algorithm benefits us in generating partitions which are maximum spaced, as a result the local neighborhood graph will have reduced number of edges.

$$\begin{aligned} T(n) &= O(n \lg n) + O(n^{3/2}) + O(n^{3/2}) + O(n^{3/2} \lg n) \\ &= O(n^{3/2} \lg n) \end{aligned}$$

5. Experimental Analysis

Performance of the proposed algorithm is compared with exact MST algorithms (CG) such as Prim's as well as Kruskal's algorithms. We also compare our results with three recent fast approximate-MST based clustering approaches namely (i) MST-inspired clustering algorithm proposed by Wang et al. [12] (referred as Wang-MST), (ii) Clustering based on near neighbor graph (CNNG) proposed by Chen [13] (referred as Chen-MST) and (iii) fast approximate-MST based on K-means (FMST) proposed by Zhong et al. [14] (referred as Zhong-MST). All the above algorithms aim to improve the computational efficiency of MST-based clustering method by means of reducing nearest neighbor cost associated with the construction of MST. Hence, the proposed algorithm is compared against these three algorithms.

As discussed earlier, the core of MST-based clustering is to identify and remove inconsistent edges (e.g., longest edges) of the MST to get desired clusters. With this basis, Wang-MST algorithm [12] tries to quickly identify potential long edge candidates of an MST before computing the exact distance values of most of the shorter ones. They used cut and cycle property to identify the long edges at an early stage. Given a list of data

points, the algorithm starts with a sequential initialization procedure, wherein each point in the dataset is assigned with a distance between itself and its immediate predecessor or immediate successor. These distances give an upper bound on the long edges for each point. Based on these distances, a spanning tree is formed. Then, the tree is refined from the brute-force nearest neighbors obtained using a divisive hierarchical clustering algorithm (DHCA).

Chen-MST [13] algorithm improves the computational performance of MST-based clustering method by incorporating $K-\delta$ near neighbor graph and grid-cell partitioning during MST construction phase. First it constructs $K-\delta$ near neighbor undirected graph $G_{K-\delta}$ based on K nearest neighbors that lie within the distance threshold δ. If $G_{K-\delta}$ is connected, then the MST is obtained from this graph using Prim's algorithm. If $G_{K-\delta}$ is unconnected, then Kruskal's algorithm is applied on the graph to obtain a set of minimum spanning forests (MSF). Next, the minimum connecting bracket edges are identified to link the disjoint components of the MSF in order to get final MST. Due to the brute-force nearest neighbor search used in $G_{K-\delta}$ graph construction phase, time complexity of this algorithm turns out to be $O(n^2)$. In order to speed up $K-\delta$ near neighbor search phase, the author has used multidimensional grid partitioning technique. Grid partitioning divides the given attribute-space into a number of regular regions called as grid-cells basing on interval length r_i, $i = 1, 2, \cdots, d$, where d is the number of dimensions. If a grid cell covers atleast ε number of points, then it is regarded as an effective grid cell. A set of all such effective grid cells are identified and the nearest neighbor search is applied on this set directly rather on the given feature space. Major drawback of the Chen-MST algorithm is its reliance to the number of user defined parameters such as number of nearest neighbors K, distance threshold δ and interval length in each dimension r_i.

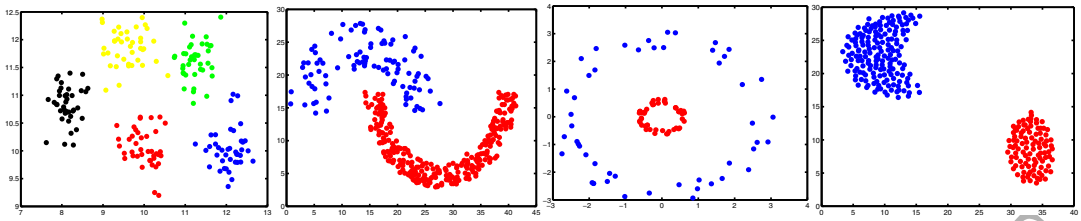
In case of Chen-MST algorithm, distance threshold δ has a crucial role in determining the nearest neighbors. For instance, a small value of δ leads to more number of disjoint components and many of which are isolated nodes. This increases the time spent in identifying minimal connecting edges for linking the disjoint components. Thus computational overhead increases with the increase in number of components of the graph. On the other hand, large value of δ ensures connectivity of the graph. But then, size of the graph poses a bottleneck and accordingly the cost of $K-\delta$ nearest neighbor search also increases. Thus choosing an optimal value of δ that result in connected sparse graph is utmost important for Chen-MST algorithm. We have run the algorithm for varying values of δ. Empirically we analyzed that setting δ to the average length of $\sqrt{n}$ longest edges in the exact MST leads to reduced runtime of Chen-MST algorithm. Hence we follow this heuristic for all the datasets.

Similar to the previous two approaches, Zhong-MST [14] also tries to get approximate nearest neighbors through divide and conquer approach. It uses K-means clustering to get $\sqrt{n}$ subsets, and then applies Prim's algorithm on each of these individual subsets. Next, a primary approximate MST is obtained by connecting MSTs of the subsets via nearest points between two neighboring subsets. In order to handle boundary points misclassification, same divie-and conquer approach is applied by giving focus on the boundary points and a second approximate MST is obtained. Finally, the two approximate MSTs are merged as a graph G and Prim's algorithm is applied on G to get final approximate MST.

We analyze efficiency of the algorithms in terms of size of the graph they generate ($|E|$), time required to construct MST ($T(n)$) and validity of clusters obtained from the approximate MSTs. For proposed algorithms, $T(n)$ is the sum of partitioning time (T_p), time taken for graph generation ($T_{intra} + T_{inter}$) and time taken for constructing the approximate-MST (T_{MST}) either by Kruskal's or Prim's algorithm as is the case. Similarly, we measure the above parameters for other algorithms also. All the algorithms have been implemented in C++. Experiments were conducted on a computer with an Intel Core2 Duo Processor 2.83GHz CPU and 4GB memory running Ubuntu operating sysem. All the algorithms are run for 10 times and the average value of $T(n)$ is observed. In case of K-means based partitioning, we just need only approximate closest points. Hence, the maximum number of K-means iterations is set to 10 for all the experiments. We have used both synthetic as well as real world datasets for experimental study.

5.1. Results on synthetic datasets

The synthetic datasets are chosen such that each dataset represents a different kind of clustering problem. We consider Gaussian clusters, half-moon clusters, cluster-inside-cluster and well-separated clusters. For each of the above categories, we choose two datasets: one with smaller number of points for the ease of visual illustration and other dataset with large number of points. Data11 and Data12 represent Gaussian clusters. Data21 and Data22 contain two half-moon clusters. Data31 and Data32 comprise of a cluster inside another cluster. Data41 and Data42 contain well-separated clusters. Details of the synthetic datasets are given in Table 1.



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

Fig. 5. Synthetic datasets used in the experiments. (a) Gaussian dataset (Data1). (b) Half-moon dataset (Data2). (c) Cluster-inside-cluster dataset (Data3). (d) Well-separated clusters (Data41).

.

Table 1. Details of synthetic 2-dimensional datasets: No. of instances (n), No. of clusters (k).

Dataset	n	k	Description
Data11	200	5	Gaussian clusters
Data12	5000	15	
Data21	373	2	Half-moon clusters
Data22	11522	2	
Data31	100	2	Cluster inside clusters
Data32	10033	2	
Data41	272	2	Well-separated clusters
Data42	10000	5	

First we illustrate through examples how approximate-MST of the proposed algorithms preserve the structure of an dataset using sparse local neighborhood graph. Consider the dataset Data11 containing 5 Gaussian clusters as shown in Fig. 6. Wang-MST algorithm has shown much divergence from exact MST due to their sensitivity to the sequential initialization. Despite applying brute-force nearest neighbor search, Chen-MST also misses some true nearest neighbors and thus it has slight deviation from the exact MST. Comparing the approximate-MSTs from proposed KNNG and BNNG algorithms shown in Fig. 6i and Fig. 6k with exact MST shown in Fig. 6a, it is understood that structure of the dataset depicted by the approximate-MST does not deviate from that of exact MST. This is the reason that clustering pattern obtained by removing the longest edges of approximate-MST is almost similar to that of exact MST. It is also observed that KNNG algorithm is able to achieve better cluster separation than other algorithms. Similarly Zhong-MST algorithm is also able to get approximate-MST closer to that of exact MST.

Comparison of algorithms on half-moon dataset is illustrated in Fig. 7. This dataset contains two half-spirals shaped like a crescent with different densities. While the lower spiral is highly dense and compact, the upper spiral is sparsely connected. While approximate MSTs of BNNG and Zhong-MST algorithms closely match with the exact MST, KNNG slightly deviates from the exact one. Due to the local optimum partitioning of K-means in KNNG algorithm, centers of the preliminary partitions are not well separated. As a result, local neighborhood graph tends to include long edges between farthest points, while excluding the edges between actual nearest neighbors. Considering quality of clusters, the longest edge of the MST is not actually between upper and lower spirals, but between two points of upper spirals. This is the reason eaxct MST is not able to separate the spirals. But in BNNG, the longest edge goes between upper and lower spirals and thus it produces approximately similar results with that of actual clusters.

Consider the case where a cluster is inside another cluster as shown in Fig. 8. Here, Wang-MST has shown large deviation from the exact MST. Chen-MST also shows variation in maintaining the long-edge which separates the inner cluster from outer one. It is also observed that MST of BNNG algorithm is almost similar to exact MST and thus cluster structure is obtained by CG and BNNG algorithms are same. But MST of KNNG and Zhong-MST algorithms are quite different from exact one due to their imbalanced preliminary partitioning.

In case of well-separated clusters as shown in Fig. 9, all the algorithms are able to attain similar partitioning as that of actual clusters. But, real life datasets are not always well-separated and thus a fast approximate-MST solution that deals with all types of cluster problems is required. Compared to other approximate-MST algorithms, our proposed algorithm BNNG is able to retain structure of the dataset through partition based

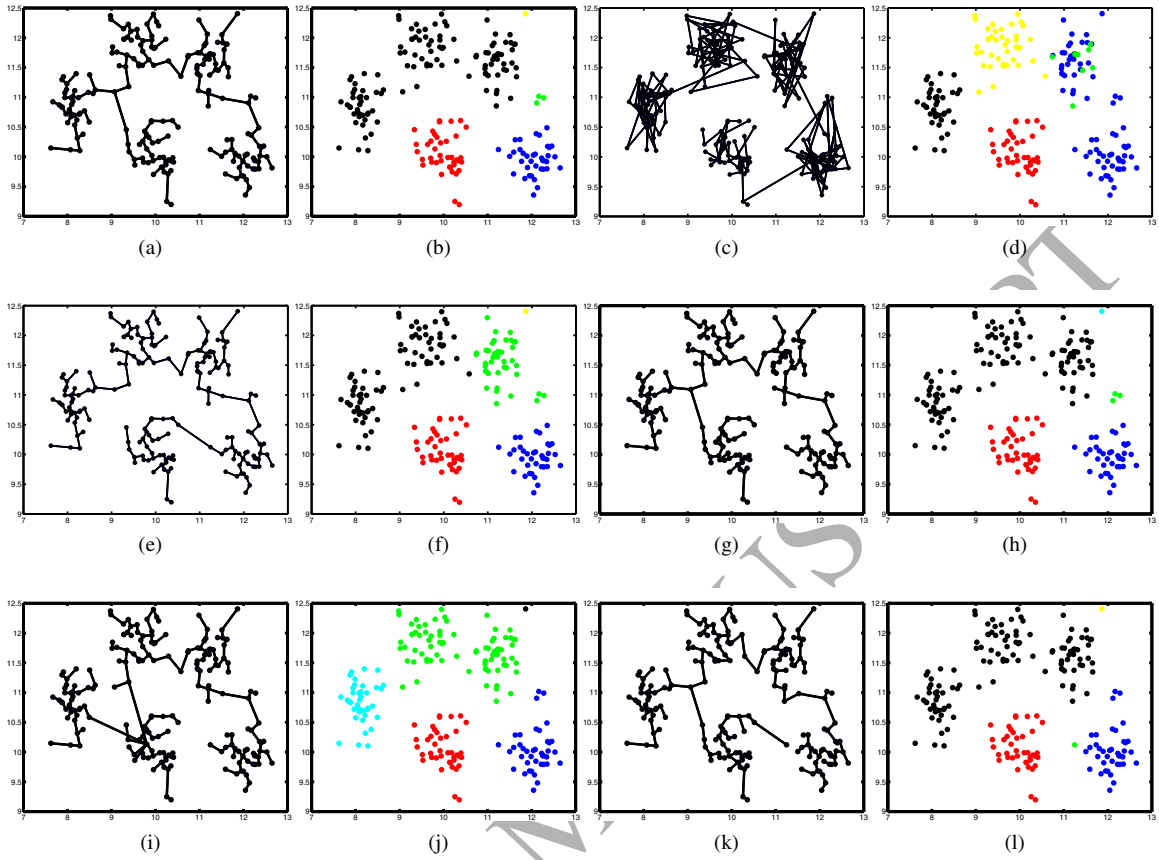


Fig. 6. Comparison of algorithms on Gaussian dataset with number of clusters $k = 5$: MST and the clusters identified by removing $k - 1$ longest edges from the MST are compared. (a) Exact MST constructed from complete graph. (b) Clusters from exact MST. (c) Approximate MST from Wang-MST algorithm. (d) Clusters from Wang-MST. (e) Approximate MST from Chen-MST algorithm. (f) Clusters from Chen-MST. (g) Approximate MST from Zhong-MST algorithm. (h) Clusters from Zhong-MST. (i) Approximate MST from the proposed KNNG algorithm. (j) Clusters from KNNG. (k) Approximate MST from the proposed BNNG algorithm. (l) Clusters from BNNG.

.

nearest neighbor graph and hence the approximate-MST of BNNG does not see much fluctuation irrespective of cluster problem.

Similar experiments were carried out on larger instance of each type of clustering problem. As number of points increases, size of the graph $|E|$ poses a major constraint on the nearest neighbor search especially in sorting phase of Kruskal's algorithm. The proposed algorithm tackles this by keeping $|E|$ as smaller as possible using Centroid based nearest neighbor rule. Table 2 shows size of the graph $|E|$ generated by different algorithms on synthetic datasets Data12, Data22, Data32 and Data42. Although Wang's algorithm does not generate graph, we measure $|E|$ as the total number of distance comparisons taken by the algorithm. Wang-MST starts with the spanning tree ST obtained from sequential initialization and further refines it using DHCA process. So here $|E|$ denotes the sum of $|ST|$ (which is $n - 1$) and K-nearest neighbors obtained from DHCA and marked-DHCA procedures. It is worthwhile to note here that $|E|$ of both Wang-MST and Chen-MST algorithms depend on the user-defined parameters such as number of nearest neighbors and distance threshold. Table 3 shows the time taken by different algorithms on synthetic datasets. As size of the graph increases, amount of time spent in sorting the huge list of edges also increases tremendously. This is the reason that Kruskal's version of the algorithms CG, BNNG and

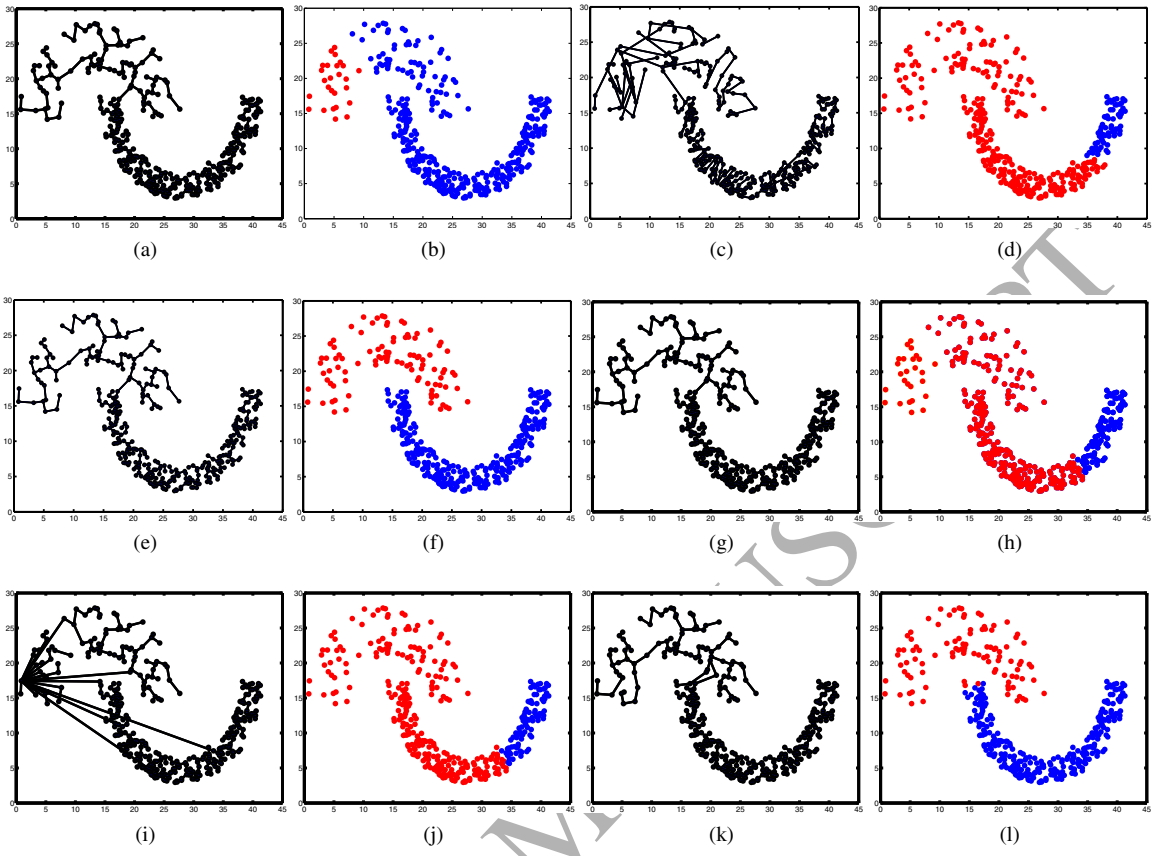


Fig. 7. Comparison of algorithms on Half-moon dataset with $k = 2$: MST and the clusters identified by removing $k - 1$ longest edges from the MST are compared. (a) Exact MST constructed from complete graph. (b) Clusters from exact MST. (c) Approximate MST from Wang-MST algorithm. (d) Clusters from Wang-MST. (e) Approximate MST from Chen-MST algorithm. (f) Clusters from Chen-MST. (g) Approximate MST from Zhong-MST algorithm. (h) Clusters from Zhong-MST. (i) Approximate MST from the proposed KNNG algorithm. (j) Clusters from KNNG. (k) Approximate MST from the proposed BNNG algorithm. (l) Clusters from BNNG.

.

KNNG are much slower as compared to their Prim's version. The results indicate that the proposed algorithm BNNG-Prims outperforms rest of the algorithms on all the datasets.

We can observe from the above results that both the proposed algorithms BNNG and KNNG outperform the exact MST algorithms by a larger margin. Chen-MST has poor performance as compared to other algorithms on all the datasets. Although Wang-MST algorithm has obtained considerable performance improvement over exact MST and Chen's algorithm, overhead caused by multiple DHCA-runs degrade its performance. In case of Zhong-MST algorithm, efficiency depends on the quality of partition resulting from K-means phase. If K-means does not result in balanced partitions, then obtaining exact MST of each subset may introduce substantial delay, which in turn increases the overall run time of Zhong-MST algorithm. The same reasoning applies to our proposed algorithm KNNG which also uses K-means for obtaining preliminary partitions. However, the CBR nearest neighbor process saves distance computation of many shorter edges and hence KNNG performs relatively better than Zhong-MST algorithm. Using a Bi-means based partitioning, BNNG algorithm overcomes the effect of imbalanced partitions and also captures most of the true nearest neighbors of the points quickly. This is the reason, BNNG is able to outperform almost on all the datasets.

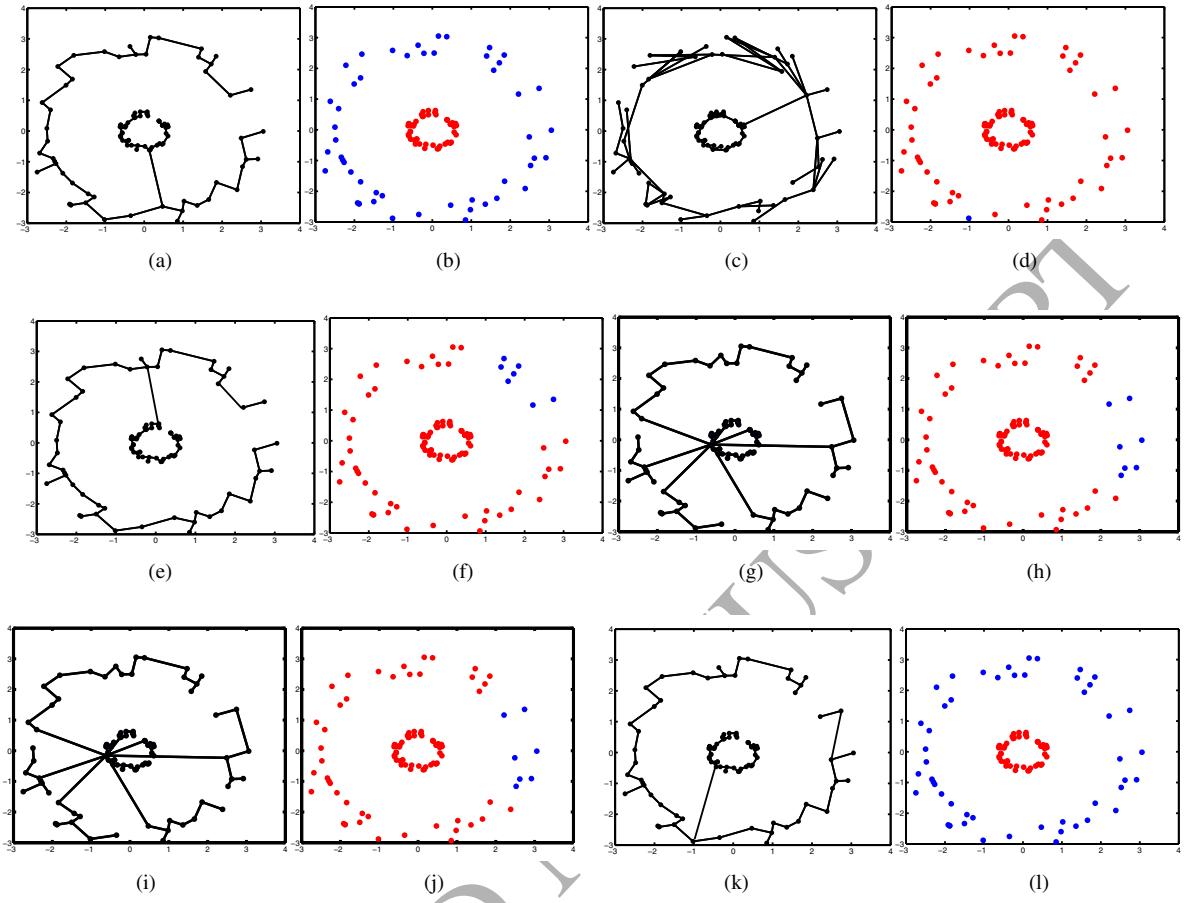


Fig. 8. Comparison of algorithms on cluster-inside-another cluster dataset with $k = 2$: MST and the clusters identified by removing $k - 1$ longest edges from the MST are compared. (a) Exact MST constructed from complete graph. (b) Clusters from exact MST. (c) Approximate MST from Wang-MST algorithm. (d) Clusters from Wang-MST. (e) Approximate MST from Chen-MST algorithm. (f) Clusters from Chen-MST. (g) Approximate MST from Zhong-MST algorithm. (h) Clusters from Zhong-MST. (i) Approximate MST from the proposed KNNG algorithm. (j) Clusters from KNNG. (k) Approximate MST from the proposed BNNG algorithm. (l) Clusters from BNNG.

.

Table 2. Comparison of $|E|$ obtained by different algorithms on synthetic datasets.

Dataset	CG	Wang-MST [12]	Chen-MST [13]	Zhong-MST [14]	KNNG	BNNG
Data12	12497500	4200568	210154	9998	911669	160378
Data22	66372481	3655408	42396	23042	3187984	500372
Data32	50325528	671803	422854	20064	1462062	413894
Data42	49995000	842388	1960090	19998	10317631	9081024

Table 3. Comparison of $T(n)$ (in seconds) obtained by different algorithms on synthetic datasets.

Dataset	CG-Kruskal	CG-Prims	Wang-MST [12]	Chen-MST [13]	Zhong-MST [14]	KNNG-Kruskal	KNNG-Prims	BNNG-Kruskal	BNNG-Prims
Data12	20.14	1.56	1.19	2.27	0.66	1.03	0.53	0.12	**0.06**
Data22	145.8	10.94	4.95	5.62	2.9	4.47	1.43	0.48	**0.21**
Data32	81.59	7.24	3.58	29.12	2.44	2.00	1.17	0.28	**0.15**
Data42	102.11	7.26	15.44	36.48	1.77	14.3	3.65	0.43	**0.24**

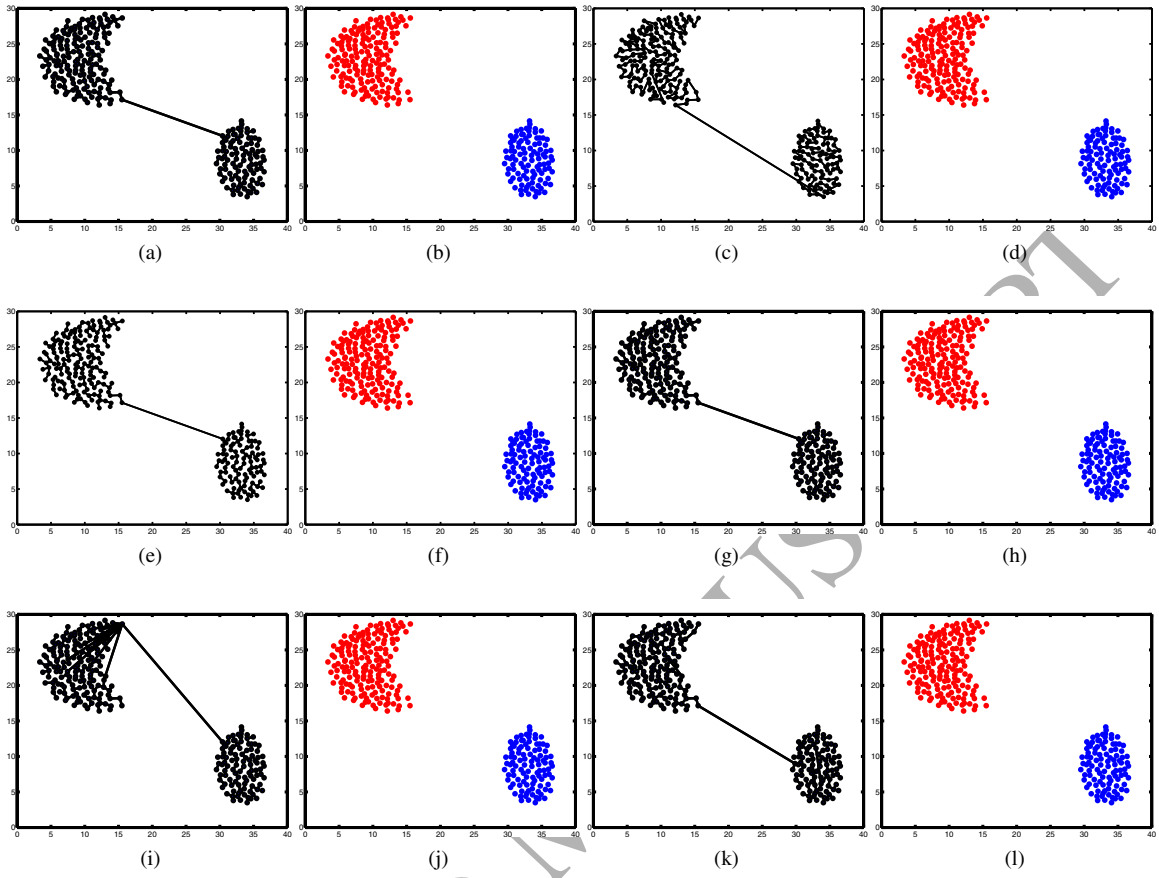


Fig. 9. Comparison of algorithms on well-separated clusters dataset with $k = 2$: MST and the clusters identified by removing $k - 1$ longest edges from the MST are compared. (a) Exact MST constructed from complete graph. (b) Clusters from exact MST. (c) Approximate MST from Wang-MST algorithm. (d) Clusters from Wang-MST. (e) Approximate MST from Chen-MST algorithm. (f) Clusters from Chen-MST. (g) Approximate MST from Zhong-MST algorithm. (h) Clusters from Zhong-MST. (i) Approximate MST from the proposed KNNG algorithm. (j) Clusters from KNNG. (k) Approximate MST from the proposed BNNG algorithm. (l) Clusters from BNNG.

.

Table 4. Details of the real datasets: No. of genes (n), Time points (d) and No. of clusters (k).

Dataset Accession number	n	d	k
GDS608	6303	10	26
GDS1550	9275	6	21
GDS759	6350	24	25
GDS1013	9275	24	18
GDS2347	6228	13	18

5.2. *Results on real datasets*

We have tested our algorithm also on publicly available 6 yeast microarray time series gene expression data sets which are downloaded from Gene Expression Omnibus (http://www.ncbi.nlm.nih.gov/geo/). Details of the datasets are given in Table 4. Table 5 shows the size of the graph generated by different algorithms on real datasets. It is clear from the results that $|E|$ of proposed local neighborhood graph is much smaller as compared to the complete graph. Table 6 shows the time taken by different algorithms on real datasets. Chen-MST has taken more run time as compared to other approximate MST algorithms. Moreover, grid partitioning used in nearest neighbor search incurs additional overhead when number of dimension increases. This is clearly evident in the datasets GDS759 and GDS1013, and thus their runtime is approximately same as that of

exact MST. The results shown in Table 6 demonstrate that the proposed algorithm BNNG-Prims has shown considerable performance improvement than other algorithms almost on all the datasets.

5.3. Effect of partitioning time on the overall efficiency

Table 7. Comparison of partitioning time (T_p) (in seconds) obtained by different algorithms on real datasets.

Dataset	Wang-MST [12]	Chen-MST [13]	Zhong-MST [14]	KNNG	BNNG
GDS608	2.11	2.72	0.82	0.83	**0.12**
GDS1550	3.18	3.33	0.87	0.92	**0.16**
GDS759	4.89	6.03	2.06	1.9	**0.40**
GDS1013	8.45	13.48	4.68	4.68	**1.07**
GDS2347	2.77	3.40	1.29	1.48	**0.14**

As the proposed algorithm relies on preliminary partitions, the amount of time spent on partitioning phase also make an impact on the performance of the algorithm. Table 7 reports the measures of time taken for obtaining the preliminary partitions (T_p) of different algorithms on various datasets. As the proposed algorithm KNNG and Zhong-MST [14] use K-means based partitioning, T_p of both these algorithms are approximately same. The proposed algorithm BNNG uses Bi-means based partitioning which is much faster than K-means.

T_p is negligible as compared to the time required for graph and MST construction. While comparing BNNG and KNNG algorithms, T_p of BNNG is much smaller than that of KNNG. This difference is clearly visible as the size of the dataset increases. In order to show the effect of dataset size on T_p, we consider randomly chosen subsets from the given dataset and measure the run time of T_p. We start from the sample size of 1000 and at each run, we increase size of the sample by 1000. Fig. 10 illustrates the time taken for partitioning T_p on GDS1550 and GDS1013 datasets. The figure also shows the partitioning time for Wang-MST, Chen-MST and Zhong-MST algorithms. For Wang-MST, T_p denotes the time spent on DHCA process. In case of Chen MST algorithm, T_p represents the time required to obtain $K-\delta$ nearest neighbors. BNNG algorithm takes relatively lesser time T_p as compared to other algorithms.

5.4. Accuracy analysis of MST

The quality of clusters generated from the approximate-MST depends on the accuracy of edges in the MST. Accuracy of the approximate-MST is determined based on edge-error and weight-error [14]. Edge-error is computed as the number of edges in the approximate-MST that do not match with the exact MST. Weight-error is computed as the weight difference between weights of the exact and approximate-MST. Let T and T' be the exact and approximate-MST of the dataset X respectively. Let W and W' be the weights of T and T' respectively. As defined in [14], edge-error and weigh-error are given as follows.

$$Edge-error = 1 - \frac{|E_{approx}|}{n-1}$$

where $|E_{approx}|$ is the number of edges in the approximate-MST that matches with exact MST.

$$Weight-error = \frac{W'-W}{W}$$

The edge-error and weight-error of all the algorithms on synthetic datasets are shown in Fig. 11a and Fig. 11b respectively. Similarly, the edge-error and weight-error of all the algorithms on real datasets are shown in Fig. 12a and Fig. 12b respectively. The error analysis on synthetic datasets shows that the Chen-MST algorithm has obtained maximum error rate with average edge-error of 11.29% and average weight-error of 5.88%. Wang-MST also has obtained average error rate closer to Chen-MST. The error rate of Zhong-MST and proposed algorithm KNNG are almost same with average edge-error of 5.5% and 5.7% respectively for Zhong-MST and KNNG, and with average weight-error of 2.6 % and 2.7% respectively for Zhong-MST and KNNG. It is also observed that our BNNG algorithm has attained lowest error rate with average edge-error of 2.6% and weight-error 1.7%. These measures indicate that the proposed algorithm BNNG produces approximate-MST much faster than other algorithms with reduced error rate.

5.5. Effect of distance threshold on determining neighboring partitions

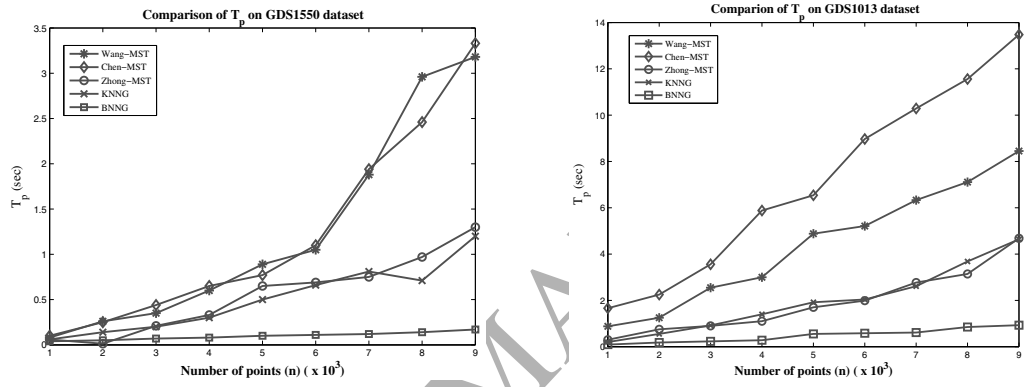
In case of proposed algorithms KNNG and BNNG, neighboring relation of preliminary partitions has an impact in both time taken for constructing approximate-MST and accuracy of the MST. KNNG and BNNG algorithms use center MST T_μ as given in Definition 1 for determining neighboring nature of preliminary partitions. In our previous work [21], we have used the average of all pair distances between centers of the partitions in order to rule out their closeness. This section compares both these approaches as distance threshold in determining the neighboring partitions. Table 8 reports the comparison of BNNG and BMGClust [21] algorithms according to distance threshold ω, runtime T_{MST} and $Edge-error$. ω of BNNG is much smaller than that of BMGClust and hence it reduces the radius for nearest partitions search. Thus BNNG is much faster

Table 5. Comparison of $|E|$ obtained by different algorithms on real datasets.

Dataset	CG	Wang-MST [12]	Chen-MST [13]	Zhong-MST [14]	KNNG	BNNG
GDS608	19860753	2168988	796793	12604	835902	863337
GDS1550	46679667	30883181	619368	18668	770002	809534
GDS759	20158075	7347339	2468671	12698	1492333	2279889
GDS1013	46679667	40014518	1142256	18668	1310180	1602138
GDS2347	19390878	17857536	5195283	12454	1283959	4608655

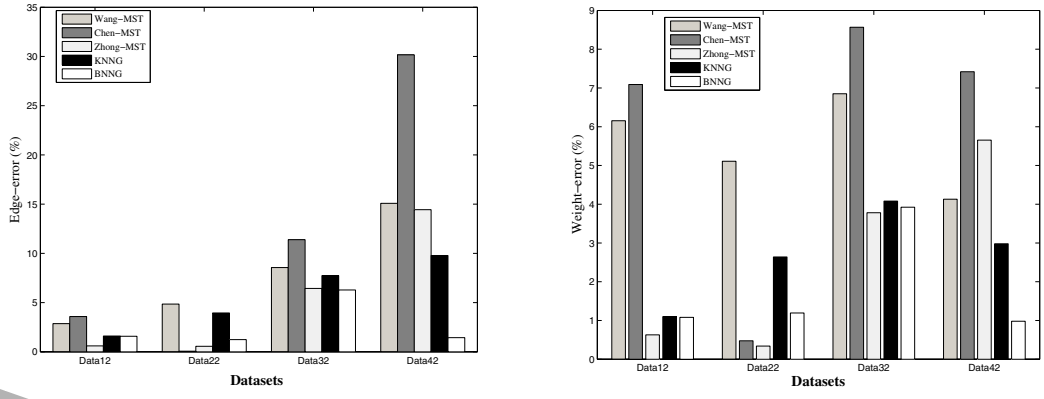
Table 6. Comparison of $T(n)$ (in seconds) obtained by different algorithms on real datasets.

Dataset	CG-Kruskal	CG-Prims	Wang-MST [12]	Chen-MST [13]	Zhong-MST [14]	KNNG-Kruskal	KNNG-Prims	BNNG-Kruskal	BNNG-Prims
GDS608	37.35	5.29	2.66	4.73	1.5	1.55	1.28	1.09	**0.66**
GDS1550	93.93	10.71	5.51	7.58	2.14	1.67	1.73	1.5	**0.71**
GDS759	45.79	10.64	9.48	13.33	3.22	3.77	3.11	4.42	**2.3**
GDS1013	107.97	28.23	14.55	101.25	7.21	6.42	5.76	5.39	**3.12**
GDS2347	38.18	5.81	36.85	7.63	1.98	3.34	2.5	4.9	**2.15**



(a) (b)

Fig. 10. Comparison of time for obtaining preliminary partitions (T_p) from different algorithms.



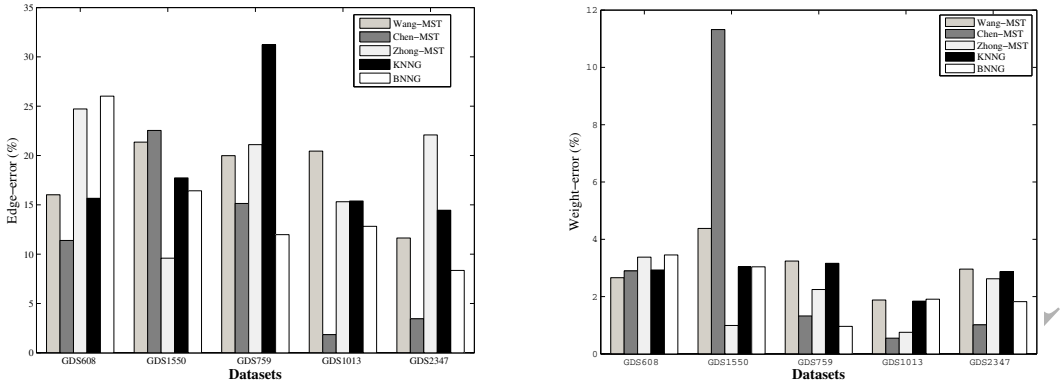
(a) (b)

Fig. 11. Edge and weight error analysis on synthetic datasets.

than BMGClust algorithm [21] as can be seen from the table.

5.6. Cluster quality analysis

In order to demonstrate that the local neighborhood graphs generated by the proposed algorithm do not miss any information that is significant for clustering, we test



(a) (b)

Fig. 12. Edge and weight error analysis on real datasets.

Table 8. Comparison BNNG and BMGClust [21] algorithms according to distance threshold ω, runtime T_{MST} and $Edge - error$.

Dataset	BMGClust [21]			BNNG		
	ω	T_{MST}	$Edge - error$	ω	T_{MST}	$Edge - error$
GDS608	302.67	4.65	19.45	16.45	0.66	26.02
GDS1550	1890.55	6.11	17.69	144.86	0.71	16.42
GDS759	257.86	4.98	13.00	32.84	2.3	11.98
GDS1013	708.97	11.64	14.88	112.92	3.12	12.82
GDS2347	167.74	5.36	6.99	44.17	2.15	8.35

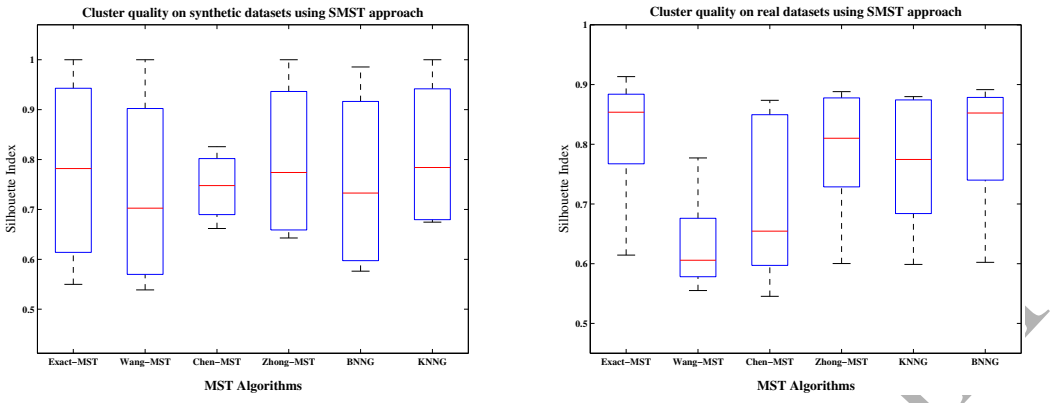
quality of the clusters generated using the approximate-MST obtained by the proposed algorithms BNNG and KNNG. Two MST-based clustering approaches are considered: (i) Zhan's standard MST-based clustering algorithm (SMST) [11] and (ii) MST-based clustering using betweenness heuristics (B-MST) [9]. SMST simply removes the $k-1$ inconsistent longest edges from the MST to get k clusters. Performance of this algorithm is affected by size of the neighborhood and presence of outliers [25]. B-MST algorithm is based on the concept of edge betweenness, which is defined as number of times a given edge appears on the shortest path between any pairs of nodes [9]. B-MST computes betweeness scores of all the edges in the MST and removes the highest betweenness edge. This iterative process continues until desired cluster structure is obtained. For the present study, both Prim's and our proposed approximate-MST algorithms are used for constructing MST. Then clusters are determined by applying SMST and B-MST approaches.

The clusters are validated using the quality measure Silhouette index, which represents the degree of cohesion and separation of clusters obtained by a clustering algorithm [26]. Let $C = \{C_1, C_2, \cdots, C_k\}$ be the clustering solution obtained by an algorithm. Consider a point i which has been assigned to the cluster C_i. Let a_i denotes the average distance between the point i to the rest of the points in the same cluster C_i. Compute the average distance between the point i and the points in other clusters $C_j, C_i \neq C_j$, let b_i denotes the minimum of average distances. For the i^{th} point, Silhouette index of a point is defined as follows [26].

$$s_i = \frac{(b_i - a_i)}{\max(a_i, b_i)}$$

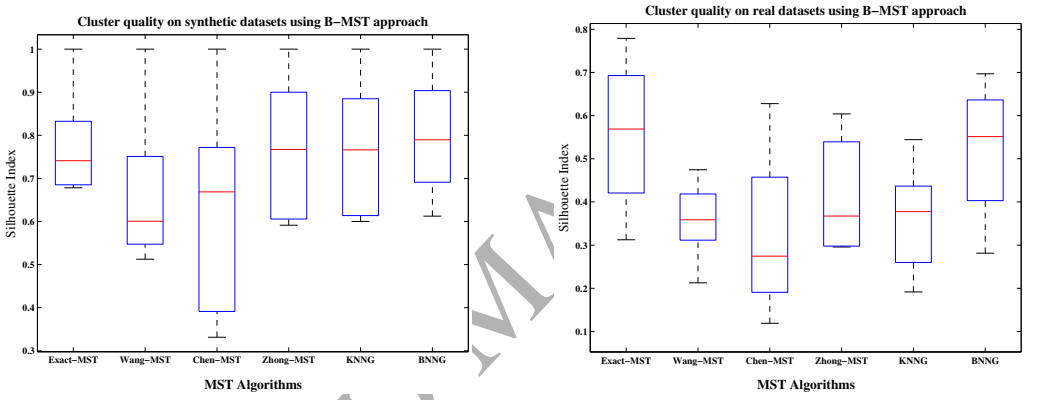
An overall measure of goodness of a clustering is obtained by computing the average silhouette coefficient of all the points in the dataset. The Silhouette index takes values in the range -1 to 1, where the value 1 indicates all the points are appropriately clustered.

Fig. 13a presents the boxplot of distribution of silhouette score obtained by SMST [11] approach using different approximate-MST algorithms on synthetic datasets. Similarly, Fig. 13b shows distribution of silhouette score obtained by SMST on real datasets. The results indicate that the proposed algorithms attain quality measures that are approximately equal or closer to that of exact MST algorithm CG. Although there is a notable edge distortion in the approximate-MST, especially in KNNG algorithm, the proposed algorithms do not miss any significant information that is essential for MST based clustering. Similar evaluation using B-MST [9] approach is reported in Fig. 14. B-MST algorithm identifies cluster boundaries using high-betweenness edges so that number of shortest paths that flow between clusters through these bridging edges are higher as compared to that pass through other edges. In case well-separated clusters, it is trivial to preserve such bridging edges in the approximate-MST and thus all



(a) (b)

Fig. 13. Comparison of Silhouette index obtained by SMST [11] approach using different approximate-MST algorithms.



(a) (b)

Fig. 14. Comparison of Silhouette index obtained by B-MST [9] approach using different approximate-MST algorithms.

the algorithms, including proposed algorithms KNNG and BNNG, are able to attain same partitioning as that of exact MST. But for other type of cluster problems, whether high-betweenness edges in the approximate-MST actually flow between different clusters depend on the quality of nearest neighbors. Centroid based nearest neighbor rule (CBR) used by the proposed algorithms effectively identifies the boundary points and bridging edges. Thus the cluster quality of the proposed algorithms are closer to exact MST. As compared to other approximate-MST algorithms, the average silhouette index obtained by the proposed algorithms are relatively higher than that of other algorithms.

6. Conclusion

This paper proposed two efficient algorithms namely partition based near neighbor graph using Bi-means partitioning (BNNG) and partition based near neighbor graph using K-means partitioning (KNNG) for obtaining MST in the context of clustering in a less than quadratic time. The theoretical analysis proved that 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 quality of the clusters. Experimental results on various synthetic and real datasets demonstrated efficiency of the proposed algorithms. Both BNNG and KNNG outperformed the exact MST algorithms by a larger margin. It has also observed that the proposed algorithms have shown considerable performance improvement over three recent

fast approximate-MST algorithms namely Wang-MST [12], Chen-MST [13] and Zhong-MST [14]. 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

References

[1] Jain, A.K., Murty, M.N., Flynn, P.J.. Data clustering: a review. ACM computing surveys (CSUR) 1999;31(3):264–323.

[2] Xu, R., Wunsch, D., et al. Survey of clustering algorithms. IEEE Transactions on Neural Networks 2005;16(3):645–678.

[3] Xu, Y., Olman, V., Xu, D.. Minimum spanning trees for gene expression data clustering. Genome Informatics 2001;12:24–33.

[4] Juszczak, P., Tax, D.M., Pekalska, E., Duin, R.P.. Minimum spanning tree based one-class classifier. Neurocomputing 2009;72(79):1859 – 1869.

[5] Zhong, C., Miao, D., Wang, R.. A graph-theoretical clustering method based on two rounds of minimum spanning trees. Pattern Recognition 2010;43(3):752–766.

[6] Zhong, C., Miao, D., Fränti, P.. Minimum spanning tree based split-and-merge: A hierarchical clustering method. Information Sciences 2011;181(16):3397–3410.

[7] Wang, X., Wang, X.L., Chen, C., Wilkes, D.M.. Enhancing minimum spanning tree-based clustering by removing density-based outliers. Digital Signal Processing 2013;23(5):1523–1538.

[8] Wu, J., Li, X., Jiao, L., Wang, X., Sun, B.. Minimum spanning trees for community detection. Physica A: Statistical Mechanics and its Applications 2013;392(9):2265–2277.

[9] Pirim, H., Ekşioğlu, B., Perkins, A.D.. Clustering high throughput biological data with B-MST, a minimum spanning tree based heuristic. Computers in Biology and Medicine 2015;62:94–102.

[10] Jothi, R., Mohanty, S.K., Ojha, A.. Functional grouping of similar genes using eigenanalysis on minimum spanning tree based neighborhood graph. Computers in Biology and Medicine 2016;71:135 – 148.

[11] Zahn, C.T.. Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Transactions on Computers 1971;100(1):68–86.

[12] 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 2009;21(7):945–958.

[13] Chen, X.. Clustering based on a near neighbor graph and a grid cell graph. Journal of Intelligent Information Systems 2013;40(3):529–554.

[14] Zhong, C., Malinen, M., Miao, D., Fränti, P.. A fast minimum spanning tree algorithm based on k-means. Information Sciences 2015;295:1–17.

[15] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., et al. Introduction to algorithms; vol. 2. MIT press Cambridge; 2001.

[16] Cheriton, D., Tarjan, R.E.. Finding minimum spanning trees. SIAM Journal on Computing 1976;5(4):724–742.

[17] Fredman, M.L., Tarjan, R.E.. Fibonacci heaps and their uses in improved network optimization algorithms. J Assoc Comput Mach 1987;34(3):596–615.

[18] Sankaranarayanan, J., Samet, H., Varshney, A.. A fast all nearest neighbor algorithm for applications involving large point-clouds. Computers & Graphics 2007;31(2):157–174.

[19] Muja, M., Lowe, D.G.. Scalable nearest neighbor algorithms for high dimensional data. IEEE Transactions on Pattern Analysis and Machine Intelligence 2014;36(11):2227–2240.

[20] Esmaeili, M., Ward, R., Fatourechi, M.. A fast approximate nearest neighbor search algorithm in the hamming space. IEEE Transactions on Pattern Analysis and Machine Intelligence 2012;34(12):2481–2488.

[21] Jothi, R., Mohanty, S., Ojha, A.. Fast minimum spanning tree based clustering algorithms on local neighborhood graph. In: Graph-Based Representations in Pattern Recognition; vol. 9069 of *Lecture Notes in Computer Science*. Springer International Publishing. ISBN 978-3-319-18223-0; 2015, p. 292–301.

[22] Bezdek, J., Pal, N.. Some new indexes of cluster validity. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 1998;28(3):301–315.

[23] Jain, A.K., Law, M.H.. Data clustering: A users dilemma. In: Pattern Recognition and Machine Intelligence. Springer; 2005, p. 1–10.

[24] Chavent, M., Lechevallier, Y., Briant, O.. DIVCLUS-T: A monothetic divisive hierarchical clustering method. Computational Statistics and Data Analysis 2007;52(2):687–701.

[25] Grygorash, O., Zhou, Y., Jorgensen, Z.. Minimum spanning tree based clustering algorithms. In: Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on. IEEE; 2006, p. 73–81.

[26] Rousseeuw, P.J.. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 1987;20:53 – 65.

R. Jothi received M.E. degree in software engineering from Anna University, Chennai. Currently, she is pursuing Ph.D. in computer science and engineering from Indian Institute of Information Technology, Design and Manufacturing, Jabalpur. Her research interests include machine learning, bio-informatics and software engineering.

Sraban Kumar Mohanty received Ph.D. degree in computer science and engineering from Indian Institute of Technology Guwahati, Assam, India in 2010. He is currently working as an Assistant Professor in computer

science and engineering discipline of PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, MP, India. His research interests are graph-based clustering techniques and Large matrix computations.

Aparajita Ojha is a professor of computer science and engineering at PDPM Indian Institute of Information Technology, Design and Manufacturing Jabalpur, India. She obtained her Ph.D. in mathematics from R.D. University Jabalpur in 1987. Her research interests include software engineering, visual cryptography and clustering techniques.