INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

AMST: Accelerated Minimum Spanning Tree for Scalable Data

Chittaranjan Mohapatra[1,2*], Nibedita Adhikari[1], B N Bhramar Ray[1], Biswojit Nayak[1], Akshaya Kumar Dash[2,1]

1 Silicon Institute of Technology, Patia, Bhubaneswar, Odisha, India
2 Utkal University, Vani Vihar, Bhubaneswar, Odisha, India

OPEN ACCESS

Received: 31-08-2023
Accepted: 05-09-2023
Published: 03-10-2023

Citation: Mohapatra C, Adhikari N, Ray BNB, Nayak B, Dash AK (2023) AMST: Accelerated Minimum Spanning Tree for Scalable Data. Indian Journal of Science and Technology 16(37): 3110-3120. https://doi.org/10.17485/IJST/v16i37.1420

* **Corresponding author**.

chitta.mohapatra@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: A traditional Minimum Spanning Tree (MST) is quite complex for large datasets concerning time and space complexities. So an Accelerated MST (AMST) is proposed for scalable data. **Methods:** AMST uses the k-means clustering to divide large data into a bunch of small-size data. The Delaunay Triangulation is applied to find the relationship among the data in $O(nlogn)$ time for n points as compared to earlier approaches. A mid-point heuristic is designed to combine all clustered MSTs to get the MST for the original data. A refinement process is used for any mislaid of minimum weight edges. **Findings:** Several desirable properties of the AMST have been developed and compared with the results of existing literature. AMST is quite competitive in terms of time complexities, least weight error rate and accuracy for both large and small-size datasets. The mathematical analysis of the time complexity $O(nlogn)$ proves that the proposed AMST has a 95% faster execution time than others. The proposed AMST has the least Weight Error Rate than other approaches which indicates that the cost is close to the exact Algorithms. The accuracy of the AMST is an average of 93% in different clustering accuracy indices which is similar to other approaches. **Novelty:** The execution time and accuracy prove that a faster MST can be developed to construct pins wire length routing in the promising field of VLSI (Very Large-Scale Integration) physical design.

Keywords: MST; Clustering; Delaunay Triangulation; Minimax Distance; Spectral Clustering; MidPoint Heuristic

1 Introduction

In classical Graph theory, the construction of a Minimal Spanning Tree (MST) is a very important aspect and attracts the attention of many researchers. It has been studied widely in various fields of computer science and engineering such as computer networks, routing, social networking, etc. In a plane for 'n' collected data points, the MST can be constructed with the following steps [(1)]. A complete graph is formed using n data points, and [(2)] the Kruskals or Prims algorithm is used to find the MST from the complete graph. This approach is feasible computationally when the data points are small in number. However, for a large number of data points, this approach becomes expensive as the run time is very high. The idea for this study originated from the way

MST is used as an input to the VLSI pins wire length routing process. Thus to deal with the MST problem for large data points an obvious approach is to find an approximate MST with near near-optimal solution.

Nowadays, the advent of big data and internet connectivity of all applications has thrown a lot of challenges in handling data efficiently as far as their requirements and customizations are concerned. For efficient handling of data, it needs to be clustered or partitioned into manageable units.

To find the MST for large-scale data points various divide-and-conquer approaches were used in the existing literature. The advantage of this approach is that a sub-problem can be handled independently without the intervention of other data sets. It indicates that the MST problem can be parallelizable and be solved efficiently by exploiting the power of multicore systems.

A bi-means clustering algorithm was used by Jothi et al.[(1)] to partition the datasets. The Euclidian distance was used between all pairs of points in the dataset to partition it into different clusters. The adjacent cluster pair was determined by the distance between all cluster center pairs, and edges between clusters were decided by a centroid-based nearest neighbor rule. Then the MST was generated by an algorithm like Prim or Kruskal's. The size of each partition being $\sqrt{n}$, the upper bound on the final height of the recursion tree was $log\sqrt{n} = O(logn)$. Similarly, the count of inter and intra-partition edges were both $O\left(n^{\frac{3}{2}}\right)$. Finally, the time to construct this approximate MST was $O\left(logn \times n^{\frac{3}{2}}\right)$.

Sandhu et al.[(2)] used k-means++ for the partitioning of the dataset. Then Prim's algorithm was applied to each of the k-clusters to find k-independent MSTs. Then these MSTs were joined to get the whole MST. Further, this MST was refined to get the optimal MST. As the clustering process was faster and the time complexity was near about $O\left(n^{\frac{3}{2}}\right)$.

The disjoint set data structures were used to determine the minimum spanning tree of a graph by Khan et al.[(3)]. A weight was assigned to each node for finding the candidate edge which was initially set to ∞. If the weight of a node was greater than the edge weight, the node's weight value was updated to the edge weight. Then the nodes were traversed as per the cheapest edge connected to it which was considered as the candidate edge. This process was used to find subgraphs of a given graph by connecting the nodes with a candidate edge without cycles. These subgraphs were considered as a single node and connected using the rest of the edges. These subgraphs were considered as a single node. This process was repeated until all subgraphs were connected. The time complexity of this algorithm was the same as Boruvka's Algorithm, Prim's Algorithm and Kruskal's Algorithm but it fails for a dense graph that is required for large datasets.

Mishra et al. divided the dataset into subsets based on the dot product of the dispersion property and position vector[(4)]. A cluster with a high variance to its centroid was considered to be the best dispersion property. The MST of each cluster was found and merged as per the maximum cohesion and intra-similarity of data points within adjacent clusters. To find the adjacent cluster pairs, a minimal spanning tree of the cluster center was generated. This algorithm's time complexity was $O\left(n^{\frac{3}{2}}\right)$.

The works in[(1–4)] obtained approximate MSTs of n data points with the complexity $O\left(n^{\frac{3}{2}}\right)$. This literature created a complete graph to get the relationship among cluster points and applied MST to each cluster thereby consuming the run time overhead of the dense graph for each cluster. Due to this reason the run time overhead of their algorithm is $O\left(n^{\frac{3}{2}}\right)$. In short, the contribution of the proposed work in this paper is as follows.

- A novel heuristic is proposed to find a suitable value of k for the k-means algorithm.
- Delaunay Triangulation is used to form a graph for each cluster as opposed to a complete graph.
- The AMST algorithm is proposed with complexity $O(n\ logn)$ as opposed to the time complexity $O\left(n^{\frac{3}{2}}\right)$.
- A new heuristic is proposed to find the second MST capturing boundary points in order to optimize the initial MST formed from earlier clusters.

The paper is organized into five Sections. Section 2 discusses some preliminary requirements for the development of AMST and mentions an overview of the proposed algorithms followed by proofs of the supporting theorems. Section 3 shows the experimental results and their comparison with existing works. Section 4 describes the concluding remark. All references are mentioned in the Section 5.

2 Methodology

This Section discusses Delaunay triangulation and the k-means clustering algorithms that are necessary for the development of proposed Accelerated MST as preliminaries. The proposed method is then discussed, followed by the mathematical justifications behind it.

2.1 Preliminaries

2.1.1 Delaunay Triangulation

A Delaunay Triangulation is made up of triangles for a set of points P, none of which are inside the circumcircle of another triangle[5]. One of the most important geometric structures is the Voronoi diagram, which is made up of Voronoi cells. The Voronoi diagram's vertices are the circumcenters of Delaunay triangles. As a result, the Voronoi diagram's Dual is a Delaunay Triangulation. In the case of a Voronoi diagram, a point can be considered as a neighbor of another if it is located within the same Voronoi circle boundary or adjacent Voronoi cell. As a result, the triangles of Delaunay Triangulations are formed using the closest points.

Let P be a maximal planner subdivision of a Delaunay Triangulation of n points in a plane. The exact number of triangles and edges in P is determined by the number of boundary points m in the Voronoi diagram's unbounded region. A Delaunay Triangulation has $2n-2-m$ triangles and $3n-3-m$ edges.

Funke et al.[6] created a divide-and-conquer algorithm for computing triangulations in two dimensions, which is improved by Nguyen et al.[7], Dinas et al.[8] and Mikhailov et al.[9]. In this algorithm, a line recursively partitions the nodes into two distinct subsets. The Delaunay triangulation is applied to each set, and then all sets are merged along the splitting line. In[10] an approach is proposed to compute the merge operation in $O(n)$ time. Hence, the final time complexity becomes $O(logn)$.

Computationally, the Delaunay triangulation can be obtained more easily from a set of points in logarithmic time. In the proposed approach Delaunay triangulation is used to get the reduced set of edges from a set of connected points, as a result, the graph formed here has a smaller number of edges compared to a complete graph.

2.1.2 K-means Algorithm

A popular optimization problem is solved using the k-means algorithm[11]. The goal is to group the data by minimizing the squared error between all points and the centroid of a cluster[12]. If μ_k is the mean of the cluster C_k and x_i is a point with it for $i = 1,2,...,n$ and k is the number of clusters, the object function is as follows.

$$\min \sum_{j=1}^{k} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2 \tag{1}$$

The observed time complexity of the k-means algorithm is $O(n^{dk+1})$ for a d-dimensional dataset[13]. Most algorithms, including Lloyd's, demonstrate that the time complexity of the k-means algorithm is linear and is equal to $O(nkdi)$, where i is the number of iterations[14]. Hence, it is useful for the clustering of large datasets.

2.2 AMST

The AMST is created in three steps for generating an approximate MST for a big dataset. In the first stage, the data is divided into different clusters, and an initial MST is constructed, comprising all MSTs of these clusters. The loss of short-distance edges during the connection phase of MSTs necessitates a refinement stage. Finally, AMST is created by combining the results of both the stages. The following subsections briefly provide more information about the intermediate stages.

1. **AMST Overview**

- **Divide-and-Conquer Stage**

a) Divide Step. Partition of the n data points into k clusters using the k-mean algorithm.

b) Conquer Step. Construct a Delaunay Triangulation graph for k cluster points. Then find MST for individual clusters.

c) Combine Step. Connecting all MSTs of k cluster to form the first approximate MST, denoted as MST_1 by using a connection criteria.

- **Refinement Stage**

a) Re-Partition the dataset over the border of the cluster produced in the previous stage using a heuristics called Mid-Point-Heuristics.

b) Generate the second approximate MST denoted as MST_2 by using the Conquer and combine step of the Divide-and-Conquer Step.

- **Merging Stage**

a) Create a graph $G_{MST} = MST_1 \cup MST_2$

b) Get the AMST by applying Kruskals algorithm to G_{MST}

2.2.1 Divide-and-Conquer Stage

- **Divide Step**

At this stage, the given dataset of size n is partitioned into $k = logn$ clusters using the k-means algorithm. This choice of $k = logn$ is different than $k = \sqrt{n}$ taken in Zhong et al.[15]. The proof of taking $k = logn$ is discussed in Theorem 2 of Section 3. As discussed in Section 2.1.1, Delaunay triangulation has less number of edges than a complete graph. Hence, a Delaunay graph is formed to get the Delaunay edges of the clustered points.

- **Conquer Step**

In this step for each cluster, a graph is formed using Delaunay Triangulation. An MST is constructed using Kruskal's algorithm for each graph of a cluster. The motivation behind the use of Kruskal's algorithm is to minimize the runtime of MST as the graph by Delaunay triangulation has a $3n-3-m$ number of edges [refer Section 2.1.1]. Thus, a small graph is used where $(E| < n^2$ as opposed to the complete graph used in earlier approaches[15], where E is the number of edges.

Table 1. Algorithm 1: Get MST_1

Algorithm 1: Get MST_1
Input: Data Set X comprising n points
Output: MSTs of clusters from X
1. Taking $k \approx log\ n$, apply the $k-$means algorithm to X to obtain k clusters $C = C_1, C_2, \ldots, C_k$choosing k seeds randomly from X
2. Construct a graph of each cluster using the Delaunay Triangulation
3. Apply Kruskal's Algorithm to the graph of each cluster to get k MSTs: $MST\ (C_i)\ \ for\ 1 \leq i \leq k$.
4. return k MSTs: MST (C_1), MST (C_2),... MST (C_k)

- **Combine Step**

A Delaunay Triangulation graph is formed using the centroids of each cluster and then an MST of the graph is obtained to find the neighboring relationship between the clusters. A pair of adjacent MSTs are connected by an edge joining two vertices belonging to respective MSTs sitting nearest to their neighbor cluster's centroids. The MST generated by this step is called the first approximate MST denoted by MST_1. The detailed discussion of the generation of MST_1 is described in Algorithms 1, 2 and 3 (Tables 1, 2 and 3).

Table 2. Algorithm 2: Joining MSTs

Algorithm 2: Joining MSTs
Input: k MSTs: MST , MST , ...MST
Output: Approximate MST of obtained from MST(C_i) for $1 \leq$ i $\leq$ k
1. Find the centroid of each cluster Ci, $1 \leq$ i $\leq$ k.
2. Construct a graph of all centroids μ = , , ..., μk using the Delaunay Triangulation.
3. Find an MSTμ of the graph obtained from step 2 using Kruskal's Algorithm to determine the neighboring relationship between clusters.
4. For each pair of clusters , that their center and are connected by e $\in$ MS Tμ, discovered the edge by Algorithm 3 (Get Connecter Edges), that connects MST and MS T.
5. Join the set of edges found in step 4 to k MSTs: MST , MST , ..., MST to get the First approximate MST: MST1.
return {MST1}

- **Refinement**

The MST_1 obtained using the divide-and-conquer approach is suboptimal. Because some edges between the two adjacent MSTs lead to optimality being missed while combining the MSTs. So a refinement of the dataset over the boundary is required to capture those edges. Hence, a repartitioning of the dataset over the boundary is required. A mid-point heuristics is proposed for repartitioning in the following subsection that obtains the second approximate MST called MST_2.

- **Re-Partition**

Table 3. Algorithm 3: Get Connecter Edges

Algorithm 3: Get Connecter Edges
Input: A neighbor pair MSTs,
Output: Connecter edge between
1. Find the nearest vertex a $\in$ from the centroid of cluster .
2. Find the nearest vertex b $\in$ MST from the centroid of cluster .
3. return } // connecter edge.

To get a refined MST, the dataset is partitioned again by targeting all points that lie across the boundaries between a pair of clusters. The initial centroid of this clustering process is the mid-points between two neighbour centroid μ_i and μ_j of cluster C_i and C_j respectively which lies across the boundary line of a pair of clusters.

A heuristic is used to calculate the mid-points. This is called Mid-Point heuristics and is defined in Algorithm 5. The mid-points are calculated for some specific edges only. If an edge cost is less than or equal to the average cost of all edges in the graph of all centroids, then the k-means clustering algorithm is applied to these mid-points. The clustering converges in one iteration only for identifying missing edges between two adjacent clusters.

- **Building of Secondary MST**

The conquer and combine steps are similar to the Conquer Step and Combine Step of the divide-and-conquer Stage. The MST generated in this stage is called the second approximate MST, termed MST_2. The step-by-step procedures are described in Algorithm 4 (Table 4).

- **Merging**

The first and second approximate MSTs are merged to produce a graph having $2(n-1)$ edges. The Kruskals Algorithm is applied on this graph to get the Accelerated MST (AMST) whose cost is pretty close to the exact MST.

Table 4. Algorithm 4: Get MST2

Algorithm 4: Get MST2
Input: Centroid Set of Clusters and Dataset having n points
Output: Second Approximate MST of : MST2
1. Construct a graph of all Centroids using Delaunay Triangulation
2. Get the mid-point using Algorithm 5 (Mid-Point-Heuristics)
3. Apply k-means clustering over mid-point as seeds of clusters and run it for a single iteration
4. Construct a graph of each cluster using the Delaunay Triangulation
5. Apply Kruskal's Algorithm to get MSTs for each cluster.
6. Join all MSTs by Algorithm 2 (Joining MSTs) to get the Second approximate MST: MST2.

Table 5. Algorithm 5: Mid-Point Heuristic

Algorithm 5: Mid-Point Heuristic
Input: A graph of Centroids μ of Clusters C
Output: A set of Mid-points
1. Find the Average Cost of all Edges in the given graph
2. Select edges having Cost $\leq$ Average Cost of all edges in the graph
3. Compute the mid-point of Selected edges
4. return {A set of Mid-points}

2.3 Complexity Analysis

Before the complexity calculation of the AMST algorithm, it has been shown that the cost of Euclidian MST obtained from a complete graph of n-points in a plane is the same as the cost of MST obtained from the Delaunay Triangulation of these n-points.

Theorem 1: Given a set of n points in a plane. Let $EMST$ (Euclidian MST) be computed in $O(n^2)$ time from the complete graph and $DMST$ (Delaunay Triangulation MST) of these n-points is provided as follows.

(i) $\text{Cost}(EMST) = \text{Cost}(DMST)$

(ii) DMST can be computed in $O(nlogn)$ time.

(i) Proof: Let $e(u,v) \in EMST$ be arbitrary. On the contrary, let us assume $\notin DMST$. Then two cases arise here.

(a) In the Delaunay Triangulation there exists $e'(u,v) \in EMST$ such that $Cost(e) = Cost(e')$. This implies Delaunay Triangulation of m point has a parallel edge which contradicts the fact that the Delaunay Triangulation is a simple graph.

(b) If $e(u,v) \notin DMST$ then u is connected to v by any intermediate vertex $w \in DMST \subset Delaunay\ Trangulation$ Graph. So the path u to v must be shortest in DMST and also in Delaunay Triangulation Graph $\subset$ Complete Graph.

As a Delaunay Graph is a subgraph of the complete graph of m points the shortest path from u to v is achieved through w. This implies $d(u,w) + d(w,v) = d(u,v)$ but this contradicts the triangle inequality $d(u,w) + d(w,u) > d(u,v)$. Thus $e \in DMST$. Since $e \in EMST$ is arbitrary. This proves that every edge $e \in EMST$ and also $e \in DMST$. Hence $Cost\ (DMST)$ and $Cost(EMST)$ are equal.

(ii) Proof : By Fortunes Algorithm, construction of Voronoi Diagram takes $O(mlogm)$. Then Delaunay Triangulation takes $O(m\ logm)$. These two procedures take $O(mlogm)$.

The Delaunay Triangulation is a subgraph of a Complete Graph. So running time of the Kruskals algorithm on Delaunay Triangulation will take time $O(mlogm) + O(E_\triangle)$, where $E_\triangle$ is the no of edges of Delaunay Triangulation. The no of edges of $E_\triangle 3m-2-A$ where A is the no of points on the boundary of the Delaunay Triangulation [5]. Then running time DMST is approximated to $O(mlogm + 3m - 2 - A) \approx O(m\ logm)$.

Theorem 2: Let us Assume a dataset D_k is equally partitioned by the k-means algorithm by taking $k = log\ n$, constructing Delaunay triangulation and then constructing k MSTs of each cluster by Prim's algorithm. Then the overall runtime of constructing MSTs is T = O(n $logn$).

Proof: Let $F(k)$ be the total running time of the proposed AMST algorithm where k is the total number of clusters. Then $F(k)$ can be written as follows: $F(k)$ =cost for creating k clusters + cost for creating k graphs using Delaunay triangulation + cost for running $Kruskal's$ algorithm on k graphs.

F(k) = $T(Clustering) + k*(T(Delaunay) + T(Kruskal's)]$

$F(k) = nkdi + k\left(\frac{n}{k}log\frac{n}{k} + 3nk - 3 - A + \frac{n}{k}log\frac{n}{k}\right)$

Where

$$T(Clustering) = O(\ nkdi) \tag{2}$$

is the time complexity for k-means clustering as per FMST [1].

T(Delaunay) is the time to compute the Voronoi Diagram for $\frac{n}{k}$ points of a cluster T_v and the time to form Delaunay triangulation for $\frac{n}{k}$ points T_D. Referring to Zhong et al. [15],

$$T_v = O\left(\frac{n}{k}log\frac{n}{k}\right), T_D = O\left(\frac{n}{k}log\frac{n}{k}\right) \tag{3}$$

The time complexity of Kruskal's algorithm is $E\ logE$. As Delaunay triangulation graph is applied to a cluster to get a linear set of edges of $\left(\frac{3n}{k} - 3 - A\right)$ for $\frac{n}{k}$ vertex edges, where A is the number of points on the boundary of the Delaunay triangulation graph.

$$T\left(\text{ Kruskal's }'\right) = O\left(\frac{n}{k}\log\frac{n}{k}\right) \tag{4}$$

Then

$$F(k) = nkdi + k\left(\frac{n}{k}\log\frac{n}{k}\right) \tag{5}$$

To get the optimal value of k, let us differentiate $F(k)$ with respect to k and $\frac{dF}{dk} = 0$, $F'(k) \approx ndi + k\left(\frac{-n}{k^2}\right)$

Taking $di = 1$ as stated in Zhong et al. [15],

$$F'(k) \approx n + k\left(\frac{-n}{k^2}\right) = 0 \tag{6}$$

On solving the above Equation 6 it is found that $k = 1$. To prove $k = 1$, is the optimal of $F(k)$, Let us calculate.

$F''(k) = -n\left(\frac{-1}{k^2}\right) = \frac{n}{k^2} > 0$

Since $F''(k) = \frac{n}{k^2}|_{k=1} n > 0$

where $k = 1$ is the point of minimum for $F(k)$. Since the number of clusters is assumed to be greater than 1, there could be many choices for k for which $F''(k) = \frac{n}{k^2} > 0$.

The choice could be $k = 1, k = \sqrt{n}, k = logn$ and $k = n^{\frac{3}{2}}$. Among these choices, the value $k = logn$ is selected to make the running time of AMST as $O(nlogn)$. Interestingly this choice of $k = logn$ suites the requirements of this work and it is cluster optimal with the widely used Elbow method for k-mean algorithms.

Next combining the results of Equations 2, 3 and 4 it is found that If $k = logn$ then the total running time (RT) is

$$F(k) = nkdi + k * O\left(\left(\frac{n}{k}\log\frac{n}{k}\right) + \left(\frac{n}{k}\log\frac{n}{k}\right) + \frac{n}{k}\log\frac{n}{k}\right)$$

$$\Rightarrow F(k) = nkdi + O\left(n\,log\left(\tfrac{n}{k}\right) + 3n - 3k - Ak + n\,log\left(\tfrac{n}{k}\right)\right)$$

$$\Rightarrow F(k) \approx nkdi + n\log\left(\frac{n}{k}\right) \tag{7}$$

Substituting $k = logn$ and taking $di = 1$ in Equation 5 results in the following

$F(k) = \left(O\left(nkdi + n\,log\frac{n}{k}\right) \right. = 1, \quad k = logn$

$$\Rightarrow F(k) = O\left(n\,logn + n\,log\frac{n}{logn}\right)$$

$\Rightarrow F(k) = O(n\,logn)$

This proves the theorem 2.

Theorem 3: Let us assume a dataset D_k is linearly partitioned by taking k = log n, then the proposed algorithm´s time complexity is T = O(n log n).

Proof: Let the total points in each cluster form an arithmetic series $n_1, n_2, \ldots, n_k$ with common difference $n_i - n_{i-1} = c$ for a constant c. Now taking $n_1, = 0$ as used in [15] the sum of the arithmetic series $\frac{k*n_k}{2} = n$.

That is $n_k = \frac{2n}{k}$ and $c = \frac{2n}{k(k-1)}$.

Since all graphs are constructed using Delaunay Triangulation for each cluster i with $n_i$data points instead of the complete graph used in [15], the complexity of forming the MST for each cluster is $O(n_i logn_i) where (1 \le i \le k)$. Thus the total complexity of constructing k-MSTs for k-clusters is:

$$T = n_1 logn_1 + n_2 logn_2 + \ldots + n_k logn_k$$

$$\Rightarrow T = clogc + 2clog2c + \ldots (k-1)\,c\;log\;(k-1)c$$

$$\Rightarrow T = c\;(logc + 2\;log2c + \ldots (k-1)\;log\;(k-1)c]$$

$$T = c\sum_{i=1}^{k-1} i\,log(ic) \tag{8}$$

By Wallis formula [16]

$$c\sum_{i=1}^{k-1} i\,log(ic) \approx c\int_1^{k-1} x\,log(x-c)dx$$

$$\approx \frac{c}{2}(k-1)^2 log\,(k-1)\,c \tag{9}$$

Putting $c = \frac{2n}{k(k-1)}$ in Equation 8, is results into,

$$c\sum_{i=1}^{k-1} i\, log(ic) \approx \frac{n(k-1)}{k} log \frac{2n}{k} \tag{10}$$

Further taking $k = logn$ in Equation 10 gives,

$$T \approx \frac{n(logn-1)}{logn} log \frac{2n}{logn} \leq \frac{nlogn}{logn}(log(2n) - loglogn) \leq nlog2 + nlogn = O(nlogn).$$

This ensures the proof of Theorem 3.

It is interesting to note that the time complexity of the proposed AMST is $O(nlogn)$, given by Theorem 3 is more computationally efficient than the time complexity $O\left(n^{\frac{3}{2}}\right)$ of others [1–4].

3 Result and Discussion

To test the efficacy of the proposed AMST against existing algorithms, the experiments are conducted on clustering datasets namely Compound, Pathbased, S1, Joensuu, A3, T4.8k, Birch1 and Europe [17] as detailed below in Table 6. Algorithms of the current work are implemented against algorithms of [1,2,15] in Python on an intel I3 machine with a processor speed of 3.70 GHz, Windows 10 Operating System having 8 GB RAM.

Table 6. Description of Datasets

	Pathbased	Compound	S1	Joensuu	A3	T4.8k	Birch1	Europe
Data size	300	399	5000	6014	7500	8000	100000	169308
Dimension	2	2	2	2	2	2	2	2

A Weight Error Rate (ERweight) metric is used to test AMST and [1,2,14] against Kruskal's algorithm. It is calculated using Equation 11. In this equation, Wexact and Wappr are the cost of Kruskal's algorithm and AMST respectively.

$$ER_{weight} = \frac{W_{appr} - W_{exact}}{W_{exact}} \tag{11}$$

A large dataset requires a partition approach that depends on the number of clusters k. A suitable value of k enables an algorithm to produce an MST whose exactness is close to Kruskal's MST. The AMST promotes choosing $k = logn$ to justify the overall running time of AMST as $k = n\, logn$. Interestingly the choice of $k = logn$ produces less ERweight for large datasets as shown in Figure 1.

In the current work, the proposed algorithm chosen $k = logn$ to justify the overall running time of AMST as $k = nlogn$. Interestingly, the value of $k = logn$ produces less ER-weight for large datasets as shown in Figure 1.

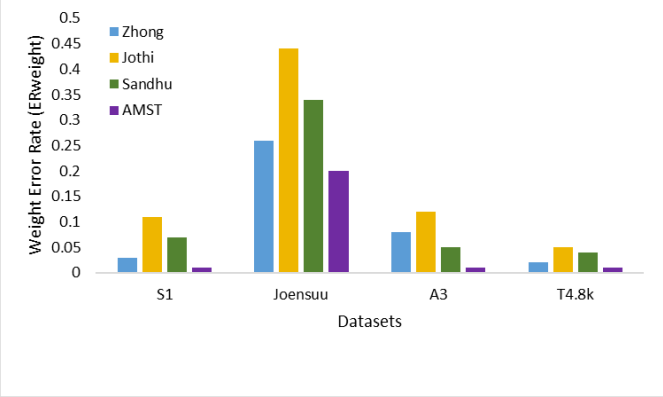

Fig 1. Weight Error Rate Comparison

The weight error rate versus datasets for various algorithms[1,2,15] is shown as a bar graph in Figure 1. The dataset is taken along the X-axis and the weight error rate is represented along the Y-axis. In the graph the columns in blue, yellow, green and violet color represent Zhong et al., Joithi et al., Sandhu et al. and the proposed AMST approach respectively. It is interesting to note that for all datasets the column shown in violet has the least height or error rate. It means the proposed AMST edges match Kruskal's algorithm edges in a difference less percentage than existing algorithms discussed in[1,2,15]. This close approximation may be attributed to the Mid-Point Heuristic and $\lceil logn \rceil$ number of clusters used in algorithms of current work. It may be mentioned that the proposed Mid-point Heuristic and the choice of the number of clusters are quite different from their counterparts used by the algorithms published earlier[1,2,15]. It is seen that the Joensuu dataset has a higher weight error rate. This is due to the layout of the data in the dataset.

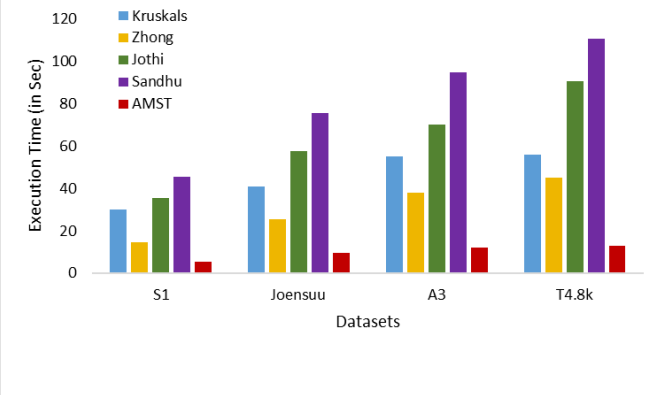

Fig 2. Execution Time Comparison

The execution time of the proposed AMST is compared with earlier works[1,2,15] in Figure 2. The Datasets are taken along the X-axis and the runtime statistics of the MST algorithms are taken along the Y-axis. The columns in blue, yellow, green, violet and red color represent the runtime of Kruskal, Zhong et al., Jothi et al., Sandhu et al and AMST approach respectively. It is deduced from the graph that the red column corresponding to AMST has the least height for all the datasets as compared to other columns. Hence AMST outperforms other procedures.

Kruskal's algorithms and[1,2,15] works are suitable for small-size datasets that are used in the above execution time comparison. From the result, it is found that the speed-up of AMST and Zhong et al.[15] is better than the speed-up of existing approximate MSTs[1,2,15]. Thus Zhong et al[15] (FMST) is selected to compare with the run time of the AMST algorithm for some scaled-up datasets in Table 7. The speed-up of FMST and AMST is broken down into the runtimes of Clustering, MST_1, MST_2 and merging as the phases are quite similar.

Table 7. Comparison of Execution Time (In Sec.) of Different phases on

Datasets	Approaches	Clustering (In Sec.)	MST_1 (In Sec.)	MST_2 (In Sec.)	Merging (in Sec)	Total (In Sec.)
1	**2**	**3**	**4**	**5**	**6**	**7**
T4.8k	FMST	1.61	31.01	41.28	0.31	74.21
	AMST	1.19	**0.77**	**1.08**	0.31	**3.35**
Birch1	FMST	33.17	89.91	95.96	0.93	219.97
	AMST	30.69	**4.52**	**5.96**	0.85	**42.02**
Europe	FMST	136.28	614.1	545.63	1.42	1297.43
	AMST	11.93	**7.18**	**7.36**	1.33	**27.77**

In Table 7, columns 1 and 2 show some large datasets and the corresponding approaches respectively. Columns 3 to column 7 contain the runtime measurement of the clustering, MST_1, MST_2, merging and total time in seconds respectively. It shows that AMST has far better runtime statistics than FMST for the MST_1 and MST_2 phases which has measurable impacts on the reduction of total runtime by 95%.

In the following the accuracy of MST as regards clustering for the proposed AMST are discussed against cluster-based MST including FMST of Zhong et al using three indices such as Rand Index[18], Adjusted Rand Index[19] and Fowlkes Mallows

Index [20]. For fare comparison purposes a similarity matrix is generated using the minimax distance approach used by Kim et al. [21] and Chehreghani [22]. The generated similarity matrix is fed into Spectral clustering to get the clustering labels using the approach of Chang and Yeung [23].

Table 8. Accuracy Test using Rand Index

Datasets	Kruskals	Zhong	Jothi	Sandhu	AMST
Pathbased	0.84	0.93	0.82	0.75	0.94
Compound	0.98	0.98	0.98	0.98	0.98

Table 9. Accuracy Test using Adjusted Rand Index

Datasets	Kruskals	Zhong	Jothi	Sandhu	AMST
Pathbased	0.643	0.86	0.614	0.509	0.86
Compound	0.962	0.962	0.96	0.973	0.96

Table 10. Accuracy Test using Fowlkes Mallows Index

Datasets	Kruskals	Zhong	Jothi	Sandhu	AMST
Pathbased	0.764	0.907	0.745	0.715	0.907
Compound	0.971	0.971	0.97	0.979	0.971

The experimental results for the accuracy of MST based on clustering using the Rand Index, Adjusted Rand Index and Fowlkes Mallows Index are presented in Tables 8, 9 and 10 respectively. For all these Table columns 1 includes datasets, and column 2, 3, 4 and 5 shows the result based on the Kruskals algorithm, Zhong et al., Jothi et al. and Sandhu et al. respectively. Whereas column 6 is the result of the proposed AMST. For the Rand Index from Table 8 it is clear that for the Pathbased dataset, the proposed AMST clustering accuracy is 94% which is superior to the methods due to Kruskals (84%), Jothi et al. (82%) and Sandhu et al. (75%) but comparable to the Zhong et al. method (93%). For the Compound dataset, Sandhu et al give 99% accuracy whereas all other methods give 98% accuracy. The 1% accuracy for the Sandhu et al. method may be attributed to the nature of the Compound dataset and the k-means++ algorithm used in the work.

The Adjusted Rand Index in Table 9 shows that for the Pathbased dataset, the AMST obtains 86% accuracy which is as the Zhong et al. method and better than others approach. For the compound dataset, the AMST has a 96% adjusted rand index value as clustering accuracy, which is almost close to or the same as other approaches. Table 10 reflects a similar scenario to the other table. The Fowlkes Mallows Index for Pathbased data AMST and Zhong et al. have 90% accuracy whereas Kruskal's, Jothi et al. and Sandhu et al. have 76%, 74%, and 71% respectively. For the Compound dataset, all method's MST accuracy is the same that is 97%. It is observed from Tables 8, 9 and 10 that the quality of the Minimum Spanning Tree resulting from proposed AMST is enhanced in comparison to the approaches of [1,2,15] and Kruskal's algorithm.

A run-time advantage of AMST for various small and large datasets is found in Figures 1 and 2 and Table 7. Thus considering the trade off between the quality of MST and run time is 95% superior to most of the state-of-the-art algorithms discussed in the paper [1,2,15].

4 Conclusion

The proposed AMST algorithm obtains an optimal MST for scalable datasets considering a Delaunay Triangulation approach. The proposed algorithm has a promising speed-up of 95% as compared to the existing algorithms and the quality of the MST is also close to the exact MST. The salient feature of the proposed AMST is time complexity is of the order $O(nlogn)$less than a quadratic time. Hence, it can be used to find the adjacent points of a graph easily for a set of points in a planer graph. The idea suggests that AMST can be utilized to construct pin routing in VLSI physical design with minimum interconnect wire length. This will lead to minimum delay and thereby it will enhance the chip performance. This approach can be easily applied to construct the optimal MST for any other graph theoretical approaches.

References

1) Jothi R, Mohanty SK, Ojha A. Fast approximate minimum spanning tree based clustering algorithm. *Neurocomputing*. 2018;272:542–557. Available from: https://doi.org/10.1016/j.neucom.2017.07.038.

2) Sandhu SS, Tripathy BK, Jagga S. KMST+: A K-Means++-Based Minimum Spanning Tree Algorithm. *Smart Innovations in Communication and Computational Sciences*. 2019;p. 113–127. Available from: https://doi.org/10.1007/978-981-10-8968-8_10.
3) Khan A, Aesha AA, Sarker J. A New Algorithmic Approach to Finding Minimum Spanning Tree. *2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT)*. 2018;p. 590–594. Available from: https://doi.org/10.1109/CEEICT.2018.8628095.
4) Mishra G, Mohanty SK. A fast hybrid clustering technique based on local nearest neighbor using minimum spanning tree. *Expert Systems with Applications*. 2019;132:28–43. Available from: https://doi.org/10.1016/j.eswa.2019.04.048.
5) De Berg M, Van Kreveld M, Overmars M, Schwarzkopf O. Computational Geometry: algorithms and applications;vol. 2. and others, editor. 1997. Available from: https://doc.lagout.org/science/0_Computer%20Science/2_Algorithms/Computational%20Geometry%20Algorithms%20and%20Applications%20%282nd%20ed.%29%20%5Bde%20Berg%2C%20van%20Kreveld%2C%20Overmars%20%26%20Schwarzkopf%202000-02-18%5D.pdf.
6) Funke D, Sanders P, Winkler V. Load-Balancing for Parallel Delaunay Triangulations. *Lecture Notes in Computer Science*. 2019;p. 156–169. Available from: https://doi.org/10.1007/978-3-030-29400-7_12.
7) Nguyen CM, Rhodes PJ. Delaunay triangulation of large-scale datasets using two-level parallelism. *Parallel Computing*. 2020;98:102672. Available from: https://doi.org/10.1016/j.parco.2020.102672.
8) Sharp N, Gillespie M, Crane K. Geometry processing with intrinsic triangulations. *ACM SIGGRAPH 2021 Courses*. 2021. Available from: https://doi.org/10.1145/3450508.3464592.
9) Mikhailov V, Pchenitchnyi P, Tagirov R, Khabibrakhmanov R, Shaimukhametov R. Analysis of algorithms for implementing Delaunay triangulation. *2021 International Conference on Information Technology and Nanotechnology (ITNT)*. 2021;p. 1–5. Available from: https://doi.org/10.1109/ITNT52450.2021.9649039.
10) Mathieson L, Moscato P. An Introduction to Proximity Graphs. *Business and Consumer Analytics: New Ideas*. 2019;p. 213–233. Available from: https://doi.org/10.1007/978-3-030-06222-4_4.
11) Ahmed M, Seraj R, Islam SMS. The k-means Algorithm: A Comprehensive Survey and Performance Evaluation. *Electronics*. 2020;9(8):1295. Available from: https://doi.org/10.3390/electronics9081295.
12) Wu G, Xu Y, Wu D, Ragupathy M, Mo YY, Chu C. Flip-flop clustering by weighted K-means algorithm. *Proceedings of the 53rd Annual Design Automation Conference*. 2016;p. 1–6. Available from: https://doi.org/10.1145/2897937.2898025.
13) Inaba M, Katoh N, Imai H. Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering. *Proceedings of the tenth annual symposium on Computational geometry - SCG '94*. 1994;p. 332–339. Available from: https://doi.org/10.1145/177424.178042.
14) Germiniani S, Pravadelli G. Exploiting clustering and decision-tree algorithms to mine LTL assertions containing non-boolean expressions. *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*. 2022;p. 1–6. Available from: https://doi.org/10.1109/VLSI-SoC54400.2022.9939640.
15) Zhong C, Malinen M, Miao D, Fränti P. A fast minimum spanning tree algorithm based on K-means. *Information Sciences*. 2015;295:1–17. Available from: https://doi.org/10.1016/j.ins.2014.10.012.
16) Miller SJ. A probabilistic proof of Wallis's formula for π. 2008. Available from: https://doi.org/10.1080/00029890.2008.11920586.
17) Franti P, Sieranoja S. K-means properties on six clustering benchmark datasets . 2018. Available from: http://cs.uef.fi/sipu/datasets/.
18) Anita R, Subalalitha CN. An Approach to Cluster Tamil Literatures Using Discourse Connectives. *2019 IEEE 1st International Conference on Energy, Systems and Information Processing (ICESIP)*. 2019;p. 1–4. Available from: https://doi.org/10.1109/ICESIP46348.2019.8938315.
19) Radu RG, Radulescu IM, Truica CO, Apostol ES, Mocanu M. Clustering Documents using the Document to Vector Model for Dimensionality Reduction. *2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*. 2020;p. 1–6. Available from: https://doi.org/10.1109/AQTR49680.2020.9129967.
20) Mishra S, Moulik S, Prakash V. Invalid Scenarios of External Cluster Validity Indices: An Analysis Using Bell Polynomial. *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2021;p. 2215–2220. Available from: https://doi.org/10.1109/SMC52423.2021.9659242.
21) Kim KH, Choi S. Neighbor search with global geometry: a minimax message passing algorithm. *Proceedings of the 24th international conference on machine learning*. 2007;p. 401–408. Available from: https://doi.org/10.1145/1273496.1273547.
22) Chehreghani MH. Efficient Computation of Pairwise Minimax Distance Measures. *2017 IEEE International Conference on Data Mining (ICDM)*. 2017;p. 799–804. Available from: https://doi.org/10.1109/ICDM.2017.95.
23) Yang Y, Shen F, Huang Z, Shen HT, Li X. Discrete Nonnegative Spectral Clustering. *IEEE Transactions on Knowledge and Data Engineering*. 2017;29(9):1834–1845. Available from: https://doi.org/10.1109/TKDE.2017.2701825.