Comparative Analysis of parallelized TopoMap with novel TopoMap

Aniket Bote*
NYU Tandon School of Engineering

Hrituja Khatavkar†
NYU Tandon School of Engineering

ABSTRACT

Topomap introduces a novel way for projecting multidimensional data. There are many visual projection techniques designed map the data. Many measures of similarity and measures of distances are considered while designing these techniques. TopoMap maps high-dimensional data to a visual space while preserving 0-homology (Betti-0) topological persistence. This is defined by Rips filtration over a set of points. Topomap uses the Euclidean distance as a measure to map the data from high dimensional space. However the compuations involved in Topomap over the exact Euclidean distance can be time consuming.
Hence our goal is to improve upon the time complexity involved during the formation of minimum spanning tree in Topomap using parallelized Euclidean Minimum Spanning Tree. Considering the work-depth approach, where work can be defined as computation of instructions and depth as total computation in a single sequence, parallelizing can be done in $W/p+D$. This approach is based on generating well separated pairs using well separated pair decomposition and then computing the minimum spanning tree using Kruskal's algorithm and bichromatic closest pairs.

1 INTRODUCTION

In TopoMap paper, while projecting points, we see that the distance measure involved is the Euclidean distance. Furthermore, the EMST is computed using Dual-Tree Boruvka. This project aims to improve upon the primary bottleneck of computing EMST. The motive is to improve the efficiency of the algorthim by using parallelized EMST in place of sequential EMST, and, thus compare the results between the two variants.We study how parallelizing the computation of EMST affects the Topomap algorithm, with respect to time and space complexity. There has been significant work done in the past over fast and sequential computation of EMST. However, parallelizing the computation is a fairly new arena. It's effects with respect to the visual projection that might be produced in TopoMap is also a result that will be studied in this project. We also study how various the results produced from other distance metrics measures like the Manhattan distance and Minkowski distance used in place of the Euclidean distance affect the projections produced by Topomap.

2 RELATED WORK

Considering the state-of-the-art projection techniques, as mentioned in Topomap there are several methods developed. Since the applications of the visualisations produced from these visualisations are in a vast mutlitude, the need to develop better techniques, that work efficiently has been a motivation of the studies that are being conducted continuously. Techniques like t-SNE UMAP, Isomap are a few dimensionality reduction techniques that also help us develop projections while preserving the geometric properties. Laurens van der Maaten presented, t-SNE [15] and has been a very eminent in the domain of dimensionality reduction. It is a neighborhood preserving technique. t-SNE is an improvement over Stochastic

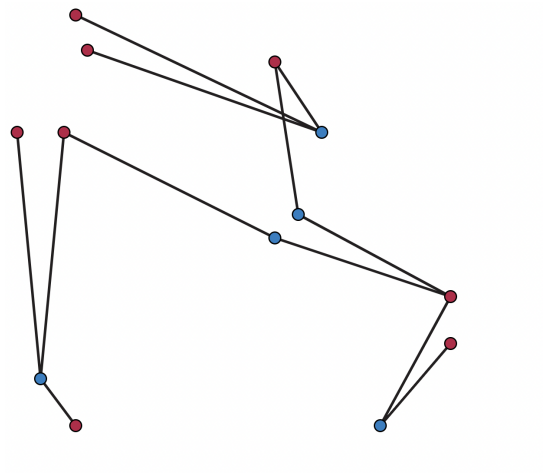

Figure 1: Euclidean Minimum Spanning Tree formed from Bichromatic Closest Pairs

. [10]

Embedding Networks. It improves on generating a single map that shows many structures in a single visualisation. Hence gives a very intricate overview of the data points and their projections. UMAP ,a technique implemented by McInnes et al. [12] , being another competitive technique, which is widely being used for creating visualisation in the domain of machine learning. UMAP again is very useful for visualising neighborhood graphs. It is based on Riemannian geometry and algebraic topology. To understand topological data analysis better we need to understand what simplical complexes is.
Simplical complexes basically are a means to construct topological spaces using combinatorial components. In geometry, a simplex is just a way to build k-dimensional object using the convex hull of k+1 independent objects. For example, a tetrahedron is a 3-simplex, i.e a simplex with 3 faces. [1] We have also seen the comparison of TopoMap with Isomap, which is a non linear dimensionality reduction technique. It tries to preserve the geodesic distances in the lower dimension. Here approximate geodesic distance is calculated between all pairs of points. Then through eigenvalue decomposition, it find low dimensional projections of the points in a given data.

Calculation of Minimum Spanning Tree is a highly sought area in the study of algorithms. There have been several ways formulated to calculate minimum spanning tree. Over years, various methods have been developed to reduce the time and space complexity it takes to compute the MST of a graph. Shamos and Hoey [14] gave an algorithm which showed that the Delaunay's triangulation is the EMST of the subgraph. Hence the time complexity being $O(n\log n)$. Yao [17] further implemented an improvement that could be solved in subquadratic time, famously being called as the Yao's Graph. Another improvement over Delaunay's triangulation was implemented by Agarwal [3] which runs at a time complexity of $O(n^{2-2/(d/2)+1}$. Arya and Mount gave an algorithm [7] to calculate the ε approximate of the Euclidean minimum spanning tree. The main aim of this approach was to get a sublinear time to compute the MST. This

*e-mail:ab9114@nyu.edu
†e-mail: hk3219@nyu.edu

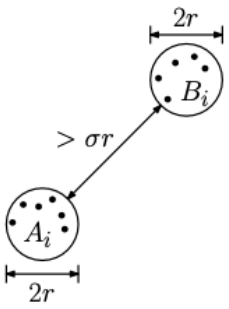

Figure 2: Well Separated Pair Decomposition
[7]

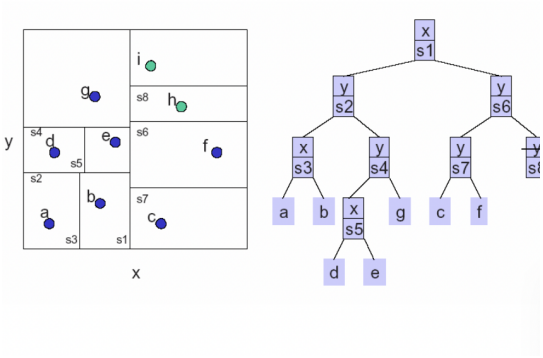

Figure 3: Kd-tree
[5]

approach is also based on well separated pair decomposition of the representative nodes of a compressed quad-tree. There also have been many randomized algorithms. Arya and Chan developed a randomized algorithm by improving the Voronoi Diagram [6]. The main motive here was to reduce the dimensional dependence and hence further improve the expected runtime. Another algorithm for ε approximation was given by Chazelle et al. where the algorithm assumes orthogonal range of empty queries and approximate nearest neighbors. [9] Some form of parallelisation is involved in the implementation given by Chatterjee et al. [8]
The computation of minimum spanning tree that we do involves the use of Euclidean distance. However there are several other distances and measures of similarity that can be considered while calculating this MST. In Manhattan distance we calculate the sum of the lengths of the projections of the line segment between the points onto the coordinate axes. This distance metric is calculated between two points in a N dimensional vector space. Minkowski distance is a generalisation of Manhattan distance and Euclidean distance and is given by $\sqrt[p]{(x1-y1)^p + (x2-y2)^p + ... + (xN-yN)^p}$ [11] for two n dimensional points. The Mahalonobis distance is a multidimensional generalization of deriving standard deviation.

3 Preliminaries and Relevant Techniques

EMST: The Euclidean minimum spanning tree is a minimum spanning tree, over a set of points **P** = $\{P_1, P_2, ...P_n\}$, where the weight of the edge between each pair of points is the Euclidean distance between those two points. It is computed over an acyclic graph with no cycles..
There are many MST algorithms that have been developed. Boruvka's algorithm, Prim's algorithm, Kruskal's algorithm are a few to name. In our use case Kruskal's suits the best, since it initially finds a minimum spanning forest before forming the minimum spanning tree.
Well Separated Pairs and Well Separated Pair Decomposition: Callahan and Kosaraju introduced the concept of WSPD. Consider a separation faction *s*, and two sets A and B. The two sets will then be called well separated such that the distance between any two points in the nodes is $s \geq 2*r$. Formally, for any two points $p, q \in S(P)$, and p lies in one set and a point q in another set.

The definition given by Callahan and Kosaraju [2] is as follows: Two sets of points set A and set B are well separated if A and B can be contained in spheres of radius r and the the minimum distance between the two sphere is at least s=2*r where for a separation constant s. We will be using separation constant as 2 for our project. In figure 4 we have set of 28 points that is forming a

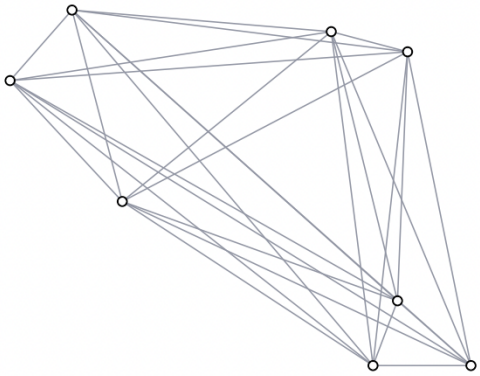

Figure 4: 28 points in a complete graph

complete graph. Further 6 shows the well separated pairs formed after well separated pair decomposition.
kdtree
One of the most common data structures used in nearest neighbor searches is kdtree. It is just like any other binary tree, but constructed recusively. Here,the nodes represent a set of points. Further, they are split into two children. This process of partitioning the node is carried further recursively until the leaf node is reached. Figure 3, shows the construction of a kd-tree. The computation can be parallelized by processing every child in parallel

Bichromatic Closest Pairs (BCCP)
Consider,two sets of points, $\{A\}$ and $\{B\}$. A BCCP returns the pair of points $p1$ and $p2$ with minimum distance between them, where $p_1 \in set(A)$ and $p_2 \in (B)$.
Figure 3 represents a well constructed euclidean spanning tree out of bichromatic closest pairs. If there are collections *A* and *B* of red and blue sets, respectively. Bichromatic Closest Pair is a method to generate a pair from $\{A\} \times \{B\}$ that has similarity higher than a given threshold according to some similarity measure. [13] The similarity measures can be Jaccard similarity, Cosine similarity, Adar index etc.

4 Methodology

Since our baseline algorithm is Topomap, in our work, we are focusing on Line 2 of the Topomap algorithm. The novel Topomap algorithm uses sequential Euclidean Minimum Spanning Tree in

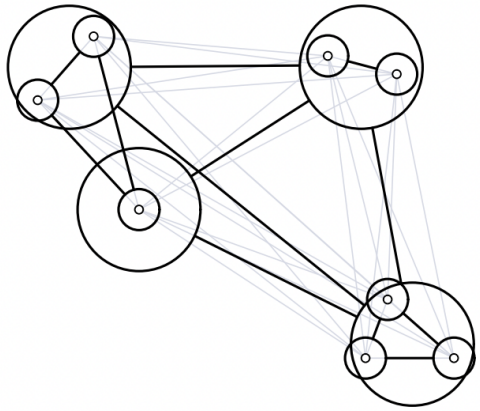

Figure 5: 12 well separated pairs created from the 28 points of the complete graph in the 4. Here the separation factor again is 2 making it 2-WSPD

Algorithm 1 Parallelized Topomap

Require: High dimensional points $P = \{p_0, p_1, p_2....\}$

1. Compute parallelized E_{mst} over P using parallel Kruskal's algorithm

2. Let $E_{mst} = e_1, e_2, e_3....e_{n-1}$ be the edges ordered on length

3. Let $P^{'} = \{p_0^{'}, p_1^{'}, p_2^{'}, ..., p_n^{'}\}$

4. Let $C_i = \{p_i^{'}\}$ be the initial set of components

for some condition **do**
Let (p_a, p_b) be the end points of edge e_i
Let C_a be the component containing $p_a^{'}$ and C_b be the component containing $p_b^{'}$
Place C_a and C_b in R^2 s.t. $min_{d(p_j^{'} \in C_a, p_k{'} \in C_b)}$ = length(e_i)
Let $C^{'} = C_a \cup C_b$
Remove C_a and C_b from the set of components, and add $C^{'}$ into this set
end forreturn $P^{'}$

the line two of the algorithm. We improve upon that by using the implemenation of parallelized Kruskal's algorithm. The rest of the steps in topomap remain the same.
However, as calculation of the MST is a significant step, improving on that step affect the algorithm in terms of time and space. Our algorithm is based on calculation of well separated pairs using well separated pair decomposition. This algorithm was used by Wang et al to calculate Hierarchical Spatial Density Clusters in their work: "Fast Parallel Algorithms for Euclidean Minimum Spanning Tree and Hierarchical Spatial Clustering" [16]. The computation of well separated pairs is based on the algorithm provided by Callahan and Kosaraju [2] According to the algorithm, the first step is the construction construction of well separated pairs, using well separated pair decomposition in parallel. If the algorithm was to be sequential, an edge is created between the BCCP using WSPD, with the distance between them equal to the weight. Then we run the Minimum Spanning Tree algorithm, i.e Kruskal's algorithm. However , in case of this parallel implementation, we parallelize the construction of WSPD and also use a parallel Kruskal's algorithm that is introduced by Wang et al [16]. An overview of the improvised algorithm is provided in Algorithm 1.

4.1 Construction of WSPD in Parallel

We need a data structure to get the well separated pairs from. Since we need a tree data structure which will generate these pairs with relatively better time complexity, kd-tree seems to be an appropriate choice. Here we are constructing a kd-tree using spatial median. The tree will be constructed in parallel with each leaf node consisting of one point.
Once constructed we use this to generate the well separated pairs. We run the WSPD algorithm in parallel on a pair of children in order to find the pairs.
The function FindPairs(A,B) initially will check if the nodes are well separated, if they are well separated, then the pair is added to the list of well separated pairs, otherwise the set is partitioned into two to run the WSPD() in parallel and recursively over all the nodes of the two sets.

Algorithm 2 WSPD in Parallel

procedure WSPDPARALLEL(A)
if $A > 1$ **then**
do in parallel
WSPD(A_l)
WSPD(A_r)
end if
FINDPAIR(A_l,A_r)
end procedure

Algorithm 3 FINDPAIR in Parallel

procedure FINDPAIR(A,B)
if $A_diam > B_diam$ **then**
Swap(A,B)
end if
if WELLSEPARATED(A,B) **then**
Record(A,B)
do in parallel
FINDPAIR(A_l,B)
FINDPAIR(A_r,B)
end if
end procedure

4.2 Parallel Kruskal's Algorithm

As proposed in the algorithm by Callahan and Kosaraju, their algorithm takes all well separated pairs to form a complete graph. This complete graph is further used to generate the MST. However, as seen in few other works like the work of Chatterjee et al, it is costly to compute all the well separated pairs. We can only consider the BCCP in a pair if there is no edge between them. This reduces the recursive calls to be made and hence reduces the time and space complexity. The parallel variant of Kruskal's is defined by Wiang et al. The original algorithm, intially takes the sorted edges, processes the edges in the order of non decreasing weights and then uses union-find to append the components generated. The algorithm we use in our project, GeoFilterKruskal i.e the parallelized Kruskal's algorithm prevents trivial computation of BCCP. Hence, the algorithm makes sure that the BCCPs will smaller weights i.e which will be less costlier to compute will be prioritized and hence reduce the time and space complexity. Each parallel procedure of Kruskal's takes edges with weights. Inside each Kruskal's subroutine the union-find is shared. The parallelKruskals() takes the following as an input
a) S, pairs of well separated pairs
b) UF, a union find structure
c) E_{out}, array to store edges of MST

Considering a constant β for each round, we only compute BCCPs for WSPDs with total sum of at most β. This as discussed above is done to reduce the cost of the computation. Further the well separated pairs S is then split into two
1. *Sl*: that has edges that are light weight and will be used in compuation of BCCP
2. *Su*: the edges that are remaining.
3. We then compute the BCCP from *Sl* and again split *Sl* into *Sl1* and *Sl2* by setting ϕ equal to distance between the sets {A} and set {B}.
4. The edges obtained from*Sl1* are then passed to Kruskal's algorithm.The algorithm suggests to double the value of β at every round to get logarithimic number of rounds.

The algorithm is further enhanced by improving the space required for the computation of parallel Kruskal's algorithm.As in the GeoFilterGFK the union find structure is shared between multipe parallel calls of Kruskal's algorithm. However here it is not so. This algorithm, i.e the MemoGFK or the memory-optimized GFK uses the basic idea to traverse the kd-tree partially and hence reducing the space in the first step itself. In each iteration we consider β and only find the BCCPs with a lower than β. In this upper(ρ_{hi}) and lower(ρ_{lo} bounds on the weighted edges are mainted. Based on the values of β, ρ_{hi} is initially calculated. This is done during the first traversal of kd-tree. Then we use the ρ_{lo} from the last iteration to get BCCP distance in range $[\rho_{lo}, \rho_{hi}]$ In the next traversal of kd-tree, we then pass the edges corresponding to this pair. Since we get the pairs on-demand, that is as and when we need , it improves the memory usage and performance as well. An elaborate algorithm as given by Wiang et al is show above in Algorithm 5 for MemoGFK.

To further understand the implementation of MemoGFK, we will need to understand the details of the terms that have been used. GETRHO calculates ρ_{hi} using the value of ρ_{lo} from previous iteration. GETPAIR returns all pairs who aren't yet connected in union find structure and have BCCP in range [ρ_{lo},ρ_{hi}]. GETEDGES finally returns the edges over which we run the Kruskal's algorithm in parallel.

5 Results

To implement our algorithm, we used datasets with various shapes to check for the correctness of our algorithm. A few of the datasets used are similar to Tomomap to get a fair idea of the comparison

Algorithm 4 Kruskal's Algorithm in Parallel

procedure PARALLELGFK($Edges: E_{out}, WSPD: S, UnionFind: UF$)
  $\beta \leftarrow 2$
  while $(A) \leq n-1$ **do**
    (S_l, S_u) = SPLIT(S, f_b) →For a pair (A,B), f_b checks if $A+B < \beta$
    $\rho = min_{A,B} d(A,B), where$ A,B $\in S_u$
    (S_{l1}, S_{l2}) = SPLIT$(S_l, f_{\rho_{hi}})$
  end while
  $E_{l1} = GETEDGES(S_{l1})$
  $PARALLELKRUSKAL(E_{l1}, E_out, UF)$
  $S = FILTER(S_{l2} \cup S_u, f_{diff})$, where f_{diff} checks points in A are different from points in B in UnionFind
  $\beta = \beta \times 2$
end procedure

Algorithm 5 Kruskal's Algorithm in Parallel

procedure MEMOGFK($Edges: E_{out}, kdtree: R, UnionFind: UF$)
  $\beta \leftarrow 2, \rho_{lo} \leftarrow 0$
  while $(E_{out}) \leq n-1$ **do**
    $\rho_{hi} = GETRHO(R, \beta)$
    $S_{l1} = GETPAIRS(R, \beta, \rho_{lo}, \rho_{hi}, UF)$
    $E_{l1} = GETEDGES(S_{l1})$
    $PARALLELKRUSKAL(E_{l1}, E_{out}, UF)$
    $\beta = \beta \times 2$

with our implementation. So, we used the following datasets:

- Fico dataset
- Seeds dataset
- mfeat dataset
- Cancer data

As a comparative study conducted over various distances like distance metrics like the euclidean distance, the manhattan distance and minkowski distance. We observed that due to the difference in the properties of how these distances are calculated, the visualisations obtained for TopoMap differ with respect to how the clustering is happening

We can also see that , greater the size of the data, greater is the tendency to probe out more. Features can clearly be seen with greater data. Also, the more the number of parameters for clusters to form , the variations in the visualisations produced by these metrics can clearly be seen. As Fico is a huge dataset, it takes highest time for computation which is about 372.32 seconds for Euclidean MST. However, no significant change is observed if the metric is changed to Manhattan or Minkowski. The least time for the computation for all datasets except seeds is in Manhattan distance.

The experiment was executed on 2-core machine instead of 48-core machine as suggested in by Y. Wang et al [16] because of compute restrictions. The parallel implementations out-performs the current implementations of EMST computations majority of the time.

Dataset	Parallel	Non-Parallel
fico	353.833s	372.32s
mFeat	14.851s	16.27s
cancer	0.049s	0.28s
seeds	0.023s	0.28s
iris	0.058s	0.1239s

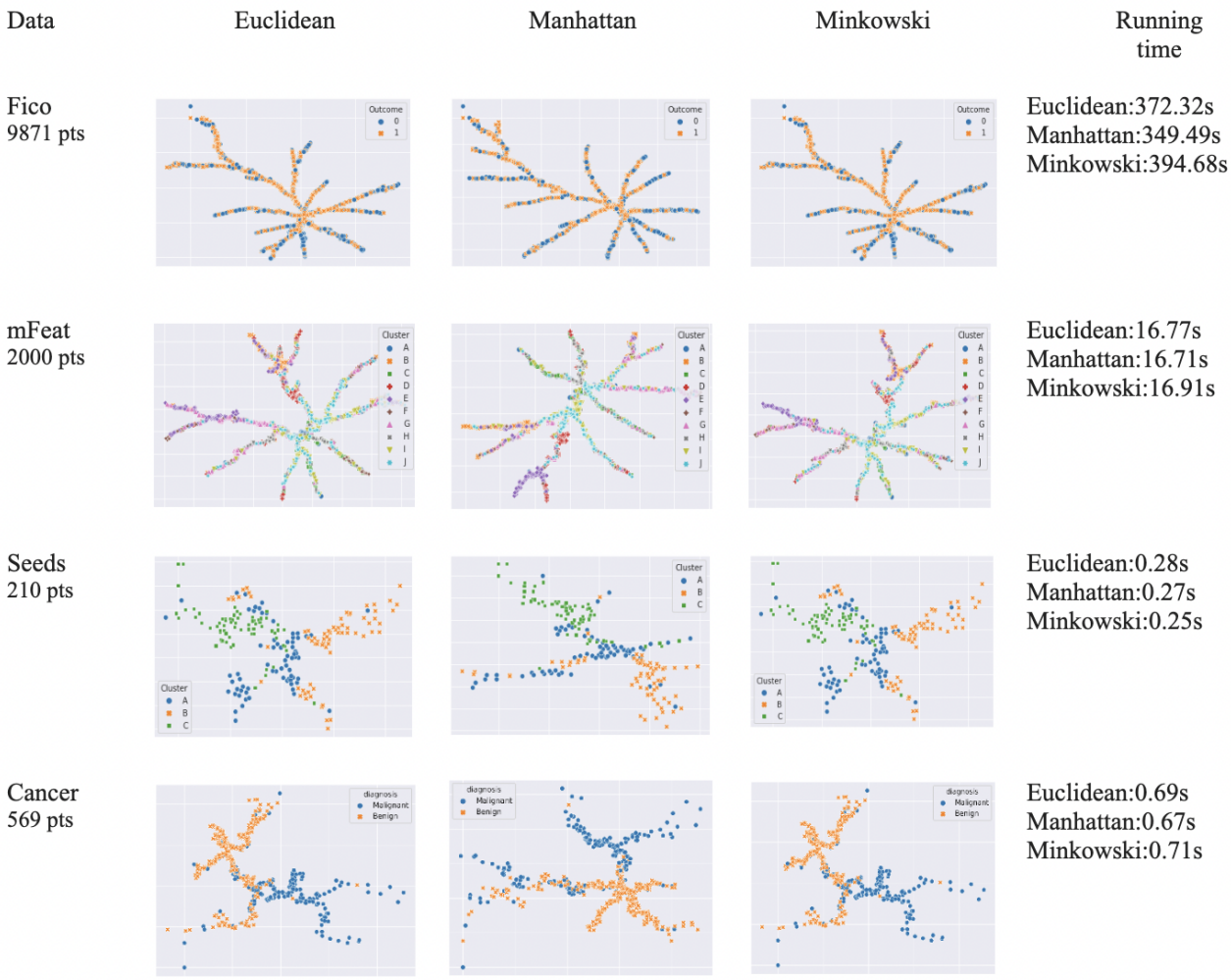

Figure 6: Comparison for difference distance metrics over 4 datasets with varying sizes

6 Discussion

Although parallelised TopoMap works well with respect time and space complexity, the algorithm is designed for upto 7 dimensions itself. Projections of higher dimensions cannot be produced by this algorithm in parallel. This is one future scope that can be worked upon. Since in real world there are huge datasets with a huge number of features (dimensions) that would be required to be projected in lower dimensions. This is on significant improvement to be considered in the algorithm. Also, Parallel Minimum Spanning Tree generated by this algorithm takes only euclidean distance into consideration. Other distance measures cannot be worked on. This is also a major drawback.

7 Conclusion

The computation of parallel minimum spanning tree and then it's implementation in TopoMap, showed how there was an improvement in time complexity. Even after using heavy datasets, the output still in sync with the expected results. As seen from the results, the various metrics i.e Euclidean, Manhattan and Minkowski that have been experimented on in order to compute minimum spanning tree, as a step of TopoMap. A variation in visualistion is observed in the way clusters of different parameters are formed spatially. However there is no much time difference observed.

8 Github Repository

The code along with the datasets used for the project above is added in the Github repository [4].

Acknowledgments

We take this opportunity to express our sincere and heartfelt gratitude to our professor, Prof. Claudio Silva for imparting us the knowledge of Visualization for Machine Learning and all the intricacies and workings of developing projects in the Visualisations and Machine learning in general.
We would also like to acknowledge with much appreciation our teaching assistants, Peter Xenopolous, Jun Yuan Jun, and Joao Rulff for their support and guidance. We will surely keep this experience in mind as we move forward and begin working on similar projects in the industry.

References

[1] How umap works¶.

[2] Well separated pair decomposition for unit-disk graph, 1995; callahan, kosaraju. *SpringerReference*. doi: 10.1007/springerreference_57999

[3] P. K. Agarwal, H. Edelsbrunner, O. Schwarzkopf, and E. Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. *Discrete amp; Computational Geometry*, 6(3):407–422, 1991. doi: 10.1007/bf02574698

[4] Aniketbote. Aniketbote/topomap-project: Visualization for machine learning research project.

[5] J. Anzola, J. Pascual, G. Tarazona, and R. González Crespo. A clustering wsn routing protocol based on k-d tree algorithm. *Sensors*, 18(9), 2018. doi: 10.3390/s18092899

[6] S. Arya and T. M. Chan. Better -dependencies for offline approximate nearest neighbor search, euclidean minimum spanning trees, and -kernels. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG'14, p. 416–425. Association for Computing Machinery, New York, NY, USA, 2014. doi: 10.1145/2582112.2582161

[7] S. Arya and D. M. Mount. A fast and simple algorithm for computing approximate euclidean minimum spanning trees. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, p. 1220–1233. Society for Industrial and Applied Mathematics, USA, 2016.

[8] S. Chatterjee, M. Connor, and P. Kumar. Geometric minimum spanning trees with geofilterkruskal. *Experimental Algorithms*, p. 486–500, 2010. doi: 10.1007/978-3-642-13193-6_41

[9] B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM Journal on Computing*, 34(6):1370–1379, 2005. doi: 10.1137/s0097539702403244

[10] D. Eppstein. Bichromatic euclidean minimum spanning trees.

[11] O. Foundation. Minkowski distance [explained], Jun 2021.

[12] L. McInnes and J. Healy. Umap: Uniform manifold approximation and projection for dimension reduction. 02 2018.

[13] R. Pagh, N. Stausholm, and M. Thorup. Hardness of bichromatic closest pair with jaccard similarity, 07 2019.

[14] M. I. Shamos and D. Hoey. Closest-point problems. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pp. 151–162, 1975. doi: 10.1109/SFCS.1975.8

[15] L. van der Maaten and G. Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

[16] Y. Wang, S. Yu, Y. Gu, and J. Shun. Fast parallel algorithms for euclidean minimum spanning tree and hierarchical spatial clustering, 04 2021.

[17] A. C.-C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982. doi: 10.1137/0211059