An efficient implementation for solving the all pairs minimax path problem in an undirected dense graph

Anonymous Author(s)
Affiliation
Address
email

Abstract

We provide an efficient $O(n^2)$ implementation for solving the all pairs minimax path problem or widest path problem in an undirected dense graph. The distance matrix is also called the all points path distance (APPD). We conducted experiments to test the implementation and algorithm, compared it with several other algorithms for solving the APPD matrix. Result shows Algorithm 4 works good for solving the widest path or minimax path APPD matrix. It can drastically improve the efficiency for computing the APPD matrix. There are several theoretical outcomes which claim the APPD matrix can be solved accurately in $O(n^2)$. However, they are impractical because there is no code implementation of these algorithms. Algorithm 4 is the first algorithm that has an actual code implementation for solving the APPD matrix of minimax path or widest path problem in $O(n^2)$, in an undirected dense graph.

1 Introduction

The minimax path problem is a classic problem in graph theory and optimization. It involves finding a path between two nodes in a weighted graph such that the maximum weight of the edges in the path is minimized. [1]

Given a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges, each edge $e \in E$ has a weight e_w. For an undirected graph with n vertices, the maximum number of edges is $\frac{n(n-1)}{2}$. A dense graph has close to $\frac{n(n-1)}{2}$ edges. We can say a dense graph has $O(n^2)$ edges. In an undirected graph, each edge is bidirectional, meaning it connects two vertices in both directions.

The objective of the minimax path problem is to find a path P from a starting node i to a destination node j such that the maximum weight of the edges in the path P is minimized. A minimax path distance between a pair of points is the maximum weight in a minimax path between the points (Equation 2).

$$\Phi = \{max\_weight(p) \mid p \in \Theta_{(i,j,G)}\} \tag{1}$$

$$M(i, j \mid G) = min(\Phi) \tag{2}$$

where G is the undirected dense graph. $\Theta_{(i,j,G)}$ is the set of all paths from node i to node j. p is a path from node i to node j, $max\_weight(p)$ is the maximum weight in path p. Φ is the set of all maximum weights. $min(\Phi)$ is the minimum of Set Φ [16].

[1] https://en.wikipedia.org/wiki/Widest_path_problem

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

Algorithm 4 MMJ distance by Calculation and Copy

Input: Ω
Output: $\mathbb{M}_\Omega$

function MMJ_Calculation_and_Copy(Ω)
    Initialize $\mathbb{M}_\Omega$ with zeros
    Construct a MST of Ω, noted T
    Sort edges of T from large to small, generate a list, noted L
    for e in L **do**
        Remove e from T. It will result in two connected sub-trees, T_1 and T_2;
        For all pair of nodes (p, q), where $p \in T_1$, $q \in T_2$. Fill in $\mathbb{M}_\Omega[p, q]$ and $\mathbb{M}_\Omega[q, p]$ with e.
    end for
    return $\mathbb{M}_\Omega$
end function

(a) Algorithm 4

```python
import networkx as nx

def cal_all_pairs_minimax_path_matrix_by_algo_4(distance_matrix):

    N = len(distance_matrix)
    all_pairs_minimax_matrix = np.zeros((N,N))

    MST = construct_MST_from_graph(distance_matrix)

    MST_edge_list = list(MST.edges(data='weight'))


    edge_node_list = [(edge[0],edge[1]) for edge in MST_edge_list]
    edge_weight_list = [edge[2] for edge in MST_edge_list]

    edge_large_to_small_arg = np.argsort(edge_weight_list)[::-1]

    edge_weight_large_to_small = np.sort(edge_weight_list)[::-1]
    edge_nodes_large_to_small = [edge_node_list[i] for i in edge_large_to_small_arg]

    for i, edge_nodes in enumerate(edge_nodes_large_to_small):
        edge_weight = edge_weight_large_to_small[i]
        MST.remove_edge(*edge_nodes)

        tree1_nodes = list(nx.dfs_preorder_nodes(MST, source=edge_nodes[0]))
        tree2_nodes = list(nx.dfs_preorder_nodes(MST, source=edge_nodes[1]))

        for p1 in tree1_nodes:
            for p2 in tree2_nodes:
                all_pairs_minimax_matrix[p1, p2] = edge_weight
                all_pairs_minimax_matrix[p2, p1] = edge_weight

    return all_pairs_minimax_matrix
```

(b) Python implementation of Algorithm 4

Figure 1: Algorithm 4 and its Python implementation. The three embedded for-loops make it look like an $O(n^3)$ algorithm, but it is actually an $O(n^2)$ algorithm.

The distance can also be called the longest-leg path distance (LLPD) [15] or Min-Max-Jump distance (MMJ distance) [16]. The all pairs minimax path distances calculate the distance between each pair of points in a dataset X or graph G . It is also called all points path distance (APPD) [15]. It is a matrix of shape $n \times n$. A dataset X can be straightforwardly converted to a complete graph.

We can use a modified version of the Floyd–Warshall algorithm to solve the APPD in both directed and undirected dense graphs [21], or use the Algorithm 1 (MMJ distance by recursion) in [16], both of them take $O(n^3)$ time. However, in an undirected dense graph, we have a better choice. We may use an $O(n^2)$ algorithm to calculate the APPD matrix. There are several theoretical outcomes which claim the APPD matrix can be solved accurately in $O(n^2)$ [20, 8, 9, 2]. However, there is no code implementation of these algorithms, which implies they are impractical.

Code implementation is the process of translating a design or algorithm into a programming language. It is critical in algorithm design where ideas are turned into practical, executable code that performs specific tasks.

In section 4.3 (MMJ distance by calculation and copy) of [16], Liu proposes an algorithm which also claims to solve the APPD matrix accurately in $O(n^2)$, in an undirected dense graph. The algorithm is referred to as Algorithm 4 (MMJ distance by Calculation and Copy). In the paper, the algorithm is left unimplemented and untested. In this paper, we introduce a code implementation of Algorithm 4, and test it.

The widest path problem is a closely related topic to minimax path problem. In contrary, The objective of the widest path problem is to find a path P from a starting node s to a destination node t such that the minimum weight of the edges in the path P is maximized. Any algorithm for the widest path problem can be easily transformed into an algorithm for solving the minimax path problem, or vice versa, by reversing the sense of all the weight comparisons performed by the algorithm. Therefore, we can roughly say that the widest path problem and the minimax path problem are equivalent.

2 RELATED WORK

Numerous distance measures have been proposed in the literature, including Euclidean distance, Manhattan Distance, Chebyshev Distance, Minkowski Distance, Hamming Distance, and cosine similarity. These measures are frequently used in algorithms like k-NN, UMAP, and HDBSCAN. Euclidean distance is the most commonly used metric, while cosine similarity is often employed to address Euclidean distance's issues in high-dimensional spaces. Although Euclidean distance is widely used and universal, it does not adapt to the geometry of the data, as it is data-independent. Consequently, various data-dependent metrics have been developed, such as diffusion distances [6, 7], which arise from diffusion processes within a dataset, and path-based distances [10, 4].

Minimax path distance has been used in various machine learning models, such as unsupervised clustering analysis [15, 12, 11, 10], and supervised classification [5, 16]. The distance typically performs well with non-convex and highly elongated clusters, even when noise is present [15].

```python
# G is an undirected dense graph, which has N vertices.
# adj_matrix is its adjacency_matrix.

def variant_of_Floyd_Warshall(adj_matrix):
    p = adj_matrix.copy()
    N = len(adj_matrix)

    for i in range(N):
        for j in range(N):
            if i != j:
                for k in range(N):
                    if i != k and j != k:
                        p[j,k] = min (p[j,k], max (p[j,i], p[i,k]))
    return p
```

Figure 2: A variant of the Floyd-Warshall algorithm for solving the minimax path problem

```python
# G is an undirected dense graph, which has N vertices.
import networkx as nx
def MST_shortest_path(G):

    MST = nx.minimum_spanning_tree(G)
    minimax_matrix = np.zeros((N, N))

    for i in range(N):
        for j in range(N):
            if j > i:
                max_weight = -1
                path = nx.shortest_path(MST, source=i, target=j)
                for k in range(len(path)-1):
                    if( MST.edges[path[k],path[k+1]]['weight'] > max_weight):
                        max_weight = MST.edges[path[k],path[k+1]]['weight']
                minimax_matrix[i,j] = minimax_matrix[j,i] = max_weight

    return minimax_matrix
```

Figure 3: Python implementation of *MST_shortest_path*, see Table 1

2.1 Calculation of minimax path distance

The challenge of computing the minimax path distance is known by several names in the literature, such as the maximum capacity path problem, the widest path problem, the bottleneck edge query problem [18, 14, 3, 13], the longest-leg path distance (LLPD) [15], and the Min-Max-Jump distance (MMJ distance) [16].

A straightforward computation of minimax path distance is computationally expensive due to the large search space [15]. However, for a fixed pair of points x and y connected in a graph $G = G(V, E)$, the distance can be calculated in $O(|E|)$ time [19].

A well-known fact about minimax path distance is: "the path between any two nodes in a minimum spanning tree (MST) is a minimax path."[14] With this conclusion, we can simplify an undirected dense graph into a minimum spanning tree, when calculating the minimax path distance.

2.2 Computing the all points path distance

Computing minimax path distance for all points is known as the all points path distance (APPD) problem. Applying the bottleneck spanning tree construction to each point results in an APPD runtime of $O(\min\{n^2 \log(n) + n|E|, n|E| \log(n)\})$ [15, 3, 13]. The resulting APPD may not be accurate when calculating with bottleneck spanning tree, because a MST (minimum spanning tree) is necessarily a MBST (minimum bottleneck spanning tree), but a MBST is not necessarily a MST. A variant of the Floyd-Warshall algorithm can calculate the APPD accurately in $O(n^3)$ [1]. Several

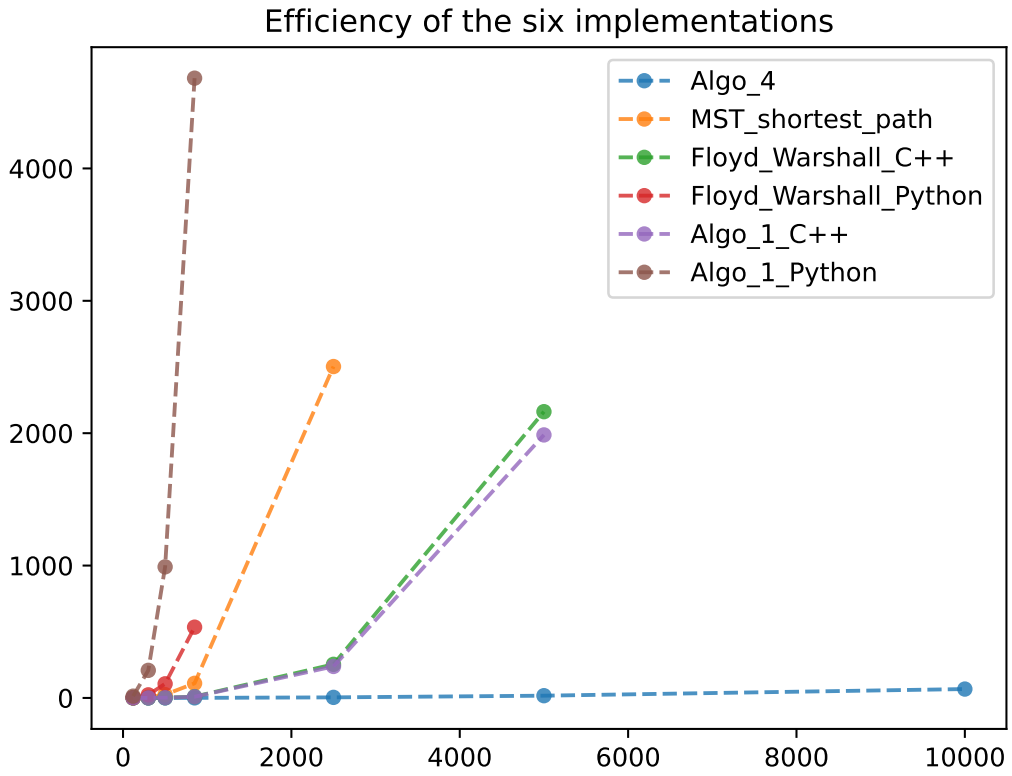

Figure 4: Performance of the algorithms (implementations)

Implementation ID	Implementation name	Complexity	Coding language	Notes
0	Algo_1_Python	$O(n^3)$	Python	Algorithm 1 (MMJ distance by recursion)
1	Algo_1_C++	$O(n^3)$	C++	Algorithm 1 (MMJ distance by recursion)
2	Floyd_Warshall_Python	$O(n^3)$	Python	A variant of Floyd-Warshall Algorithm
3	Floyd_Warshall_C++	$O(n^3)$	C++	A variant of Floyd-Warshall Algorithm
4	MST_shortest_path	$O(n^3log(n))$	Python	Calculate the shortest path in a MST
5	Algo_4	$O(n^2)$	Python	Algorithm 4 (MMJ distance by Calculation and Copy)

Table 1: Profiles of the four algorithms. Two of them are implemented with different programming languages, Python and C++

theoretical results suggest that the APPD matrix can be accurately solved in $O(n^2)$ time [20, 8, 9, 2]. However, the absence of code implementations for these algorithms indicates their impracticality.

3 Implementation of the algorithm

As described in Section 1, the Algorithm 4 (MMJ distance by Calculation and Copy) in [16] also claims to solve the APPD matrix accurately in $O(n^2)$, in an undirected dense graph. But it is left unimplemented and untested. Figure 1a is Algorithm 4 (MMJ distance by Calculation and Copy) in [16], for convenience of reading, we re-post it here. Figure 1b is its python implementation.

	data 139 (N = 120)	data 109 (N = 300)	data 18 (N = 500)	data 19 (N = 850)	data 16 (N = 2500)	data 35 (N = 5000)	data 136 (N = 10000)
Algo_1_Python	13.451s	208.363s	990.308s	4681.911s	>7200s	>7200s	>7200s
Algo_1_C++	0.033s	0.414s	1.794s	9.032s	237.961s	1986.928s	>7200s
Floyd_Warshall_Python	1.489s	23.353s	106.745s	534.683s	>7200s	>7200s	>7200s
Floyd_Warshall_C++	0.033s	0.436s	2.324s	10.035s	253.909s	2162.514s	>7200s
MST_shortest_path	0.399s	4.229s	24.926s	110.449s	2503.483s	>7200s	>7200s
Algo_4	0.02s	0.073s	0.191s	0.511s	4.311s	17.015s	67.048s

Table 2: Performance of the four algorithms. N is the number of points in the datasets.

Note the three embedded for-loops make it look like an $O(n^3)$ algorithm, but it is actually an $O(n^2)$ algorithm. Because when the variable i in $Line$ 21 is small, both *tree1* and *tree2* are of size $O(n)$; but when the variable i is large, both *tree1* and *tree2* are of size $O(1)$. The final net effect is that the three embedded for-loops only access each cell of the APPD matrix only once. Therefore, it is an $O(n^2)$ algorithm.

In the implementation, we first construct a minimum spanning tree (MST) of the undirected dense graph. The complexity of constructing a MST with prim's algorithm is $O(n^2)$. Then, we sort the edges of the MST in descending order. It is critical to remove the edges from the MST one-by-one, from large to small. Only by this we can get the two sub-trees, *tree1* and *tree2*. By traversing each sub-tree, nodes of the two sub-trees can be obtained, respectively.

4 Testing of the algorithm

In an experiment, we tested the Algorithm 4 (MMJ distance by Calculation and Copy) on seven datasets with different number of data points, note a dataset can be easily converted to a complete graph. The performance of Algorithm 4 is compared with three other algorithms that can calculate the APPD matrix.

Table 1 lists the profiles of the four algorithms. *Algo_1* is the Algorithm 1 (MMJ distance by recursion) in [16], it has complexity of $O(n^3)$; *Floyd_Warshall* is a variant of the Floyd-Warshall algorithm. Figure 2 is its python implementation. It has complexity of $O(n^3)$; *MST_shortest_path* firstly construct a minimum spanning tree (MST) of the undirected dense graph, then calculate the shortest path between each pair of nodes, then compute the maximum weight on the shortest path. Its complexity is $O(n^3 log(n))$. Figure 3 is its python implementation. The implementation is based on Madhav-99's code [2]; *Algo_4* is Algorithm 4 (MMJ distance by Calculation and Copy) in [16], it has complexity of $O(n^2)$. Both *Algo_1* and *Floyd_Warshall* are implemented with C++ and python, respectively, to test the difference between different programming languages.

4.1 Performance

Table 2 is performance of the algorithms (implementations). We test each algorithm with seven datasets which have different number of data points. The data sources corresponding to the data IDs can be found at the official code of Liu's paper [16]. [3] The values are the time of calculating the minimax path APPD by each algorithm, on a desktop computer with "3.3 GHz Quad-Core Intel Core i5" CPU and 16 GB RAM.

To save time, we stop the execution of an algorithm if it cannot obtain the APPD matrix in 7200s (two hours). The computing time is recorded only once for each dataset and algorithm. Figure 4 converts the values in Table 2 into a figure. It can be seen that Algorithm 4 has achieved a good performance than other algorithms. It can calculate the APPD matrix of 10,000 points in about 67 seconds, while other algorithms cannot finish it in two hours.

Reasonably, the C++ implementations of *Algo_1* and *Floyd_Warshall* are much faster than their python edition. Interestingly, when implemented in python, *Algo_1* is much slower than *Floyd_Warshall*, but a little faster than *Floyd_Warshall* in C++.

4.2 Solving the widest path problem

As stated in Section 7 (Solving the widest path problem) of [16], Algorithm 4 (MMJ distance by Calculation and Copy) can be revised to solve the widest path problem APPD in undirected graphs, by constructing a maximum spanning tree and sort the edges in ascending order. In another experiment, we tested using Algorithm 4 to compute the widest path APPD. Result shows Algorithm 4 works good for solving the widest path problem.

[2] `https://github.com/Madhav-99/Minimax-Distance`

[3] `https://github.com/mike-liuliu/Min-Max-Jump-distance`

5 Proof of the algorithm

A good question is why Algorithm 4 (MMJ distance by Calculation and Copy) works. Here is a theoretical proof of the correctness of the algorithm.

Whenever we are about to remove an edge e from the MST, e must belong to a connected sub-tree of MST T. The sub-tree is noted S_t. A sub-tree is a tree wholly contained in another. Note the MST T can be considered as a sub-tree of itself. We can conclude edge e is the largest edge in sub-tree S_t. Since the edges have been sorted in descending order, and edges larger than e have been removed in previous steps. It does not matter if there are other edges in S_t which are as large as e.

After removing edge e from S_t, we get two smaller connected sub-trees, $tree1$ and $tree2$. For any pair of nodes (p, q), where $p \in tree1$, $q \in tree2$, the minimax path distance between p and q must be the weight of edge e. Because "the path between any two nodes in a minimum spanning tree (MST) is a minimax path" [14], and edge e is the largest edge in sub-tree S_t. A path between p and q must pass through edge e, and edge e is the largest edge in the path. It does not matter if there are other edges in the path which are as large as e. Note a sub-tree that has only one node is considered as a valid sub-tree.

Therefore, the minimax path distance between p and q must be the weight of edge e. The correctness of Algorithm 4 (MMJ distance by Calculation and Copy) is proved.

6 Discussion

6.1 Merit of Algorithm 1

Algorithm 1 (MMJ distance by recursion) has a merit of warm-start. Suppose we have calculated the APPD matrix M_G of a large graph G, then we got a new point (or node) p, where $p \notin G$. The new graph is noted $G + p$. To calculate the APPD matrix of graph $G + p$, if we use other algorithms, we may need to start from zero. Algorithm 1 has the merit of utilizing the calculated M_G for computing the new APPD matrix, with the conclusions of Theorem 3.3., 3.5., 6.1., and Corollary 3.4. in [16]. This is especially useful when the graph is a directed dense graph, where starting from zero needs $O(n^3)$ complexity, but a warm-start of Algorithm 1 (MMJ distance by recursion) only needs $O(n^2)$ complexity. We can say Algorithm 1 supports online machine learning[4], in which data becomes available in a sequential order.

6.2 Using parallel programming

If speed is the main concern of calculating the APPD matrix, we can use parallel programming to accelerate Algorithm 4. Firstly, we can use different processors for traversing the *tree1* and *tree2* in $Line\ 25\ and\ 26$ of Figure 1b. Secondly, we can copy the minimum spanning tree (MST) to many processors. For the nth processor, we just remove the n largest edges, obtaining the nth *tree1* and *tree2*, traversing them, then fill in the corresponding positions of the APPD matrix that are decided by the nth *tree1* and *tree2*. A possible limitation of Algorithm 4 is that it needs some effort to balance the workload of each processor, when using parallel computing to accelerate it.

7 Conclusion

We implemented the Algorithm 4 (MMJ distance by Calculation and Copy), then tested the implementation and compared it with several other algorithms that can calculate the all pairs minimax path distances, or also called the all points path distance (APPD). Experiment shows Algorithm 4 works good for solving the widest path or minimax path APPD matrix. As an algorithm of $O(n^2)$ complexity, it can drastically improve the efficiency of calculating the APPD matrix. Note algorithms for solving the APPD matrix are at least in $O(n^2)$ complexity, because the matrix is an $n \times n$ matrix.

In Section 2.3.3. of the paper "Path-Based Spectral Clustering: Guarantees, Robustness to Outliers, and Fast Algorithms," [15] Dr. Murphy and his collaborators write:

[4] `https://en.wikipedia.org/wiki/Online_machine_learning`

"Naively applying the bottleneck spanning tree construction to each point gives an APPD runtime of $O(min\{n^2log(n) + n|E|, n|E|log(n)\})$. However the APPD distance matrix can be computed in $O(n^2)$, for example with a modified SLINK algorithm (Sibson, 1973), or with Cartesian trees (Alon and Schieber, 1987; Demaine et al., 2009, 2014). "

The author sent an email for further clarity about this statement.

The author:

"You indicated the APPD distance matrix can be computed in $O(n^2)$. However, I searched the Internet and github, I have not found any code implementation that can accurately calculate the APPD distance matrix in $O(n^2)$. Do you know any code implementation of that? Please indicate it to me. "

Dr. Murphy:

"If you can find an implementation of SLINK to do single linkage clustering in $O(n^2)$, then you can do APPD by reading off the distances from the resulting dendrogram. I don't know any implementations of SLINK, and it may be easier to prove things about than to implement practically. "

"Regarding tree structures, these are certainly more of theoretical interest, and I would not be surprised if there were no practical implementations of them at all. So, achieving $O(n^2)$ via those methods may be impractical. "

It is worth noting that although Dr. Murphy indicated the SLINK algorithm can be revised to solve the APPD matrix in $O(n^2)$ time, there is no code implementation showing how the SLINK algorithm can be revised to do so.

The contributions of the paper can be summarized as following:

- It provides the first code implementation for solving the all pairs minimax path problem or widest path problem in an undirected dense graph, in $O(n^2)$ time.
- It provides the fastest code implementation for solving the all pairs minimax path problem or widest path problem in an undirected dense graph.
- We provide a theoretical proof of the correctness of Algorithm 4 (MMJ distance by Calculation and Copy) .
- It indicated and verified the warm-start merit of Algorithm 1 (MMJ distance by recursion) , which is a key merit of Algorithm 1. This merit makes Algorithm 1 can calculate the all pairs shortest paths (APSP) efficiently in dynamic graphs [17].
- It explores how Algorithm 4 (MMJ distance by Calculation and Copy) can be accelerated by parallel computing, which is not straight-forward.

References

[1] Aho, A. V. and Hopcroft, J. E. (1974). *The design and analysis of computer algorithms*. Pearson Education India.

[2] Alon, N. and Schieber, B. (2024). Optimal preprocessing for answering on-line product queries. *arXiv preprint arXiv:2406.06321*.

[3] Camerini, P. M. (1978). The min-max spanning tree problem and some extensions. *Information Processing Letters*, 7(1):10–14.

[4] Chang, H. and Yeung, D.-Y. (2008). Robust path-based spectral clustering. *Pattern Recognition*, 41(1):191–203.

[5] Chehreghani, M. H. (2017). Classification with minimax distance measures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

[6] Coifman, R. R. and Lafon, S. (2006). Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30.

[7] Coifman, R. R., Lafon, S., Lee, A. B., Maggioni, M., Nadler, B., Warner, F., and Zucker, S. W. (2005). Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the national academy of sciences*, 102(21):7426–7431.

[8] Demaine, E. D., Landau, G. M., and Weimann, O. (2009). On cartesian trees and range minimum queries. In *Automata, Languages and Programming: 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I 36*, pages 341–353. Springer.

[9] Demaine, E. D., Landau, G. M., and Weimann, O. (2014). On cartesian trees and range minimum queries. *Algorithmica*, 68:610–625.

[10] Fischer, B. and Buhmann, J. M. (2003). Path-based clustering for grouping of smooth curves and texture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):513–518.

[11] Fischer, B., Roth, V., and Buhmann, J. (2003). Clustering with the connectivity kernel. *Advances in neural information processing systems*, 16.

[12] Fischer, B., Zöller, T., and Buhmann, J. M. (2001). Path based pairwise data clustering with application to texture segmentation. In *Energy Minimization Methods in Computer Vision and Pattern Recognition: Third International Workshop, EMMCVPR 2001 Sophia Antipolis, France, September 3–5, 2001 Proceedings 3*, pages 235–250. Springer.

[13] Gabow, H. N. and Tarjan, R. E. (1988). Algorithms for two bottleneck optimization problems. *Journal of Algorithms*, 9(3):411–417.

[14] Hu, T. (1961). The maximum capacity route problem. *Operations Research*, 9(6):898–900.

[15] Little, A. V., Maggioni, M., and Murphy, J. M. (2020). Path-based spectral clustering: Guarantees, robustness to outliers, and fast algorithms. *J. Mach. Learn. Res.*, 21:6:1–6:66.

[16] Liu, G. (2023). Min-max-jump distance and its applications. *arXiv preprint arXiv:2301.05994*.

[17] Liu, G. (2024). Solving the all pairs shortest path problem after minor update of a large dense graph. *arXiv preprint arXiv:2412.15122*.

[18] Pollack, M. (1960). The maximum capacity through a network. *Operations Research*, 8(5):733–736.

[19] Punnen, A. P. (1991). A linear time algorithm for the maximum capacity path problem. *European Journal of Operational Research*, 53(3):402–404.

[20] Sibson, R. (1973). Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34.

[21] Weisstein, E. W. (2008). Floyd-warshall algorithm. *https://mathworld. wolfram. com/*.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the claims made, including the contributions made in the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitation is discussed in Section 6.2.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Proofs of theoretical results have been provided in the paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We have fully disclosed all the information needed to reproduce the main experimental results of the paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide an URL to data and code of the paper, to reproduce the main experimental results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Full details are provided with the code.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [NA]

Justification: The paper does not contain statistical experimental results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have provided the information about the type of compute workers CPU and RAM.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators or original owners of assets used in the paper are properly credited. The license and terms of use are explicitly mentioned and properly respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.