Fast Shortest Path Distance Estimation in Networks using Community Detection with Landmarks based approach

Social Network Analysis for Computer Scientists — Course paper

Athul Sreejith
a.sreejith@umail.leidenuniv.nl
LIACS, Leiden University
Leiden, Netherlands

Manasvi Kumar
m.kumar.6@umail.leidenuniv.nl
LIACS, Leiden University
Leiden, Netherlands

Abstract

In this paper, we are addressing the task of efficiently computing shortest-path distances in large graphs, emphasizing the application of landmarks for distance estimations. The central challenge addressed is the need to find a balance between the precision of distance estimates and computational efficiency. To overcome this challenge, we introduce a methodology in community detection. By using community detection for landmark selection in networks we can identify key nodes that serve as critical connectors within communities, significantly enhancing our understanding of network dynamics. Initially, the graph is partitioned into communities using algorithms such as Louvain algorithm. Landmarks are strategically selected within these communities based on degree centrality. Finally, these landmarks are used to find shortest distances between two nodes in the graph. This methodology not only addresses the challenge of distance computation in large graphs but also provides a unique contribution to the optimization of graph analysis, opening new paths for further exploration in the field.

Keywords

shortest-paths, landmarks methods, social network analysis, community detection

ACM Reference Format:
Athul Sreejith and Manasvi Kumar. 2022. Fast Shortest Path Distance Estimation in Networks using Community Detection with Landmarks based approach: Social Network Analysis for Computer Scientists — Course paper. In *Proceedings of Social Network Analysis for Computer Scientists Course 2023 (SNACS '23).* ACM, New York, NY, USA, 7 pages.

1 Introduction

In the recent years, the increasing number of users on various social media platforms or the figure of citations between academic papers, patents and the number of collaboration between individuals or organizations, such as co-authorship networks in Collaboration Networks have led to a constantly increasing availability of very large networks. For better relatability let's consider the case of social media platforms. These platforms handle millions of users on a regular basis. For such platforms, computing the metrics such as providing good recommendations based on user's interests could highlight important characteristics of the network. However for networks of such enormous size, computing these measures can be a challenging task. Hence, to develop an algorithm that approximates such values can help in understanding the flow of information, identifying influential users within the network.

A key application behind these approximation can be understood by taking the example of a social media platform instagram. consider a scenario when a person A and person B have a mutual friend person C. Now if A were to find and follow person B, the search model could use the shortest path distance as a primitive ranking function for recommending other users, meaning that the results should be more closer to person B and both of their mutual friend circle. One another example can be seen as finding the connections in a coauthorship network, where we want to find the authors who have collaborated with other authors quickly.

The shortest distance path problem is already a well studied problem and many methods such as methods using random landmark selection, degree based exist, as can be seen from studying the works of [4] where the authors have analysed various methods for selecting the landmarks, and provided their results.

Our Contribution Our research aims at applying community detection before selecting the landmarks with which we strive to improve the calculation of the distance of the shortest path by narrowing down the search space and decreasing computing time for a path. In this paper we will discuss about our approach and results on using community detection in combination with landmark for finding shortest path distance efficiently and will find if we are able to improve on the existing work or not.

1) Finding optimal set of landmarks is computationally challenging and is known as **NP** hard.

2) We apply community detection using different landmark based shortest paths estimation methods and then compare their performance on five real world datasets.

2 Related work

Pruned Landmark Labeling. Our approach also involves computing BFS of the graph, however we precompute the distances between each vertex in the graph and each landmark offline. The cost of using this approach is O(nm). Akiba,Iwata and Yoshida observe that precomputing distance labels for vertices can answer the correct distance for any pair of vertices from the labels. Moreover they are able to perform 32 or 64 breadth-first searches simultaneously exploiting bitwise operations.[1]

Survey of Shortest-Path Algorithms. Since there exist a huge number of algorithms for approximating shortest paths, the authors

SNACS '23, Master CS, Fall 2023, Leiden, the Netherlands

use the approach of creating a *spanner* which is a sub-graph created from the main graph. This paper aims to present a breakdown of these shortest-path algorithms through an appropriate taxonomy. With help of this we how each shortest-path algorithm works. Using this we can identify the best shortest path algorithm is best suited for our needs.[3]

Goal-Directed Shortest-Paths. This particular algorithm has various version, but we will look at the simple Goal-Directed Search algorithm. So this algorithm uses a heuristic approach in finding the shortest-path. The main reason for understanding the use of this algorithm is that unlike Dijkstra's algorithm, A^* is an informed algorithm. It always finds the shortest path if an heuristic function is used. [2]

Application. We aim to apply the method of community detection algorithm on top these methods and observe and analyse the variations that this will give. In the previous works, landmarks have been selected based on degree centrality, betweeness centrality and partition based methods. Our understanding after referring these papers is that they all have started directly with the landmarks based approach before checking if those landmarks are actually acting as the connectors for different sections of the graph, hence we aim to explore that connection aspect.

3 Preliminaries

In this section we introduce the notations and definitions that we describe throughout the paper. We then describe how to index distances efficiently using landmarks. We formally define the landmark-selection strategies that we consider in this paper.
In addition, we outline the specific methodologies used for community detection, explaining our approach and revealing meaningful structures within the network.

Landmark Selection: Landmarks act as reference points in the network which helps in finding shortest-path distances efficiently. We use degree centrality here as the basic strategy for landmark selection in the network.

Here, we provide a brief overview of the terminology we are going to use:
A graph with n vertices and m edges. Each vertex represents a point or node in the network, and edges connect pairs of vertices, representing relationships or connections between them in the network. Let $G(V, E)$ be a graph representing the network, V be the set of nodes $\{v_1, v_2, v_3, \ldots, v_n\}$, and E be the set of edges $\{e_1, e_2, e_3, \ldots, e_m\}$, where n is the number of nodes, m is the number of edges and d be the number of landmarks to be selected, where landmark itself are the nodes in the graph we selected using degree centrality.

In our methodology, we first apply the Louvain community detection algorithm to a graph $G(V, E)$. A community detection algorithm is a method used in network analysis to identify groups or clusters of nodes (vertices) in a network that are more densely connected to each other than to the rest of the network. The initial assignment of each node to its own community is denoted as $C_i = i$, for each node i in the graph. The Louvain iteration involves optimizing modularity (ΔQ) by iteratively moving nodes to communities that maximize the modularity gain. Modularity is a measure used to evaluate the quality of a partition of a network into communities or modules. It quantifies the extent to which the number of edges within communities is higher than what would be expected by random chance. Modularity assesses the strength of the division of a network into cohesive groups. The final output of the community detection algorithm is a partition of nodes into communities, denoted as c_i for each node i.
In the second step, landmark selection, we identify important nodes within communities using degree (L_d). This set of landmarks is denoted as L_k. Shortest distance paths from each node to these landmarks and from landmarks to nodes are calculated using Dijkstra's algorithm, We therefore use two distance matrices Node to landmark distance matrix (Mn) in the form n x l, where n as rows as the number of nodes in the graph and l as columns as the number of landmark we choose and for Landmark to node distance matrix (Ml) in the form l x n where l as rows as the number of landmarks and n as columns as the number of nodes in the graph respectively. To estimate the approximate shortest path distance between source node x and destination node y, we use Node to landmark distance matrix (Mn) and Landmark to node distance matrix (Ml).
This methodology provides a practical approach to community-based graph analysis and distance estimation, demonstrating its effectiveness through the selected notations and procedures.

4 Approach

Step 1: Applying Community Detection
Initialize each node in the graph to its own community.
Assign community as $C_i = i$ for each node i in the graph G.
Consider Graph $G(V, E)$ with V vertices and E edges. We apply Louvain community detection algorithm to this graph.

`Louvain Iteration Steps:` Define the Louvain iteration steps as shown below until no big increase in modularity is achieved:
`Modularity Calculation for Single Node:` For each node i, calculate the modularity gain ΔQ by moving i to the community of each of its neighbors. Define $\Delta Q(i \to j)$ as the modularity gain of moving i to the community of j. Modularity calculation is given by:

$$\Delta Q(i \to j) = \left[\frac{\left(\sum_{in} - k_i^{in} \cdot k_j^{in} \right) / 2m}{2m} \right] - \left[\frac{\left(\sum_{\text{tot}} - k_i \cdot k_j \right) / 2m}{2m} \right]$$

where

Σ_{in} : Sum of weights of edges inside the community of j.

k_i^{in} : Sum of weights of edges from i to nodes in the community of j.

Σ_{tot} : Sum of weights of all edges in the graph.

k_i : Sum of weights of edges connected to i.

k_j : Sum of weights of edges connected to nodes in the community of j.

m : Total edge weight in the graph.

`Modularity Maximization:` Node Movement: Move the node i to the community j that maximizes $\Delta Q(i \rightarrow j)$.
Node movement in Modularity Maximization is given by:

$$c_i = \arg\max_j \Delta Q(i \rightarrow j)$$

For each community in the network, calculate how much the modularity would increase if the node i were moved to that community and then we select the community where moving node i results in the maximum increase in modularity. This step aims to iteratively move nodes to different communities to maximize the overall modularity of the network partition. The goal of this step is to find a configuration where nodes are grouped into communities in a way that maximizes the density of connections within communities while minimizing connections between communities.

`Aggregate Communities:` Form a new graph where nodes represent communities from the previous step. Edge weights between communities are the sum of weights of edges between their constituent nodes. Community Aggregation: Nodes become communities, and edges represent inter-community connections.
`Repeat or Stop:` If modularity increases after the community aggregation step, repeat Louvain Iteration Steps again. Otherwise, stop the iteration. The final output is a partition of nodes into communities. Final Output: ci, for each node i.

Step 2: Landmark Selection
After finding communities, we need to find important nodes of those communities to select as landmarks. Here, we apply degree centrality in each community. Let L_d be the set of all nodes with high degree centrality.

$$L_d = \{Ld_1, Ld_2, Ld_3, ...Ld_n\}$$

After finding these landmarks, we calculate the approximate shortest distance path from each node in the graph to these landmarks using Dijkstra's algorithm. We store these distances in the distance matrix for further computations.

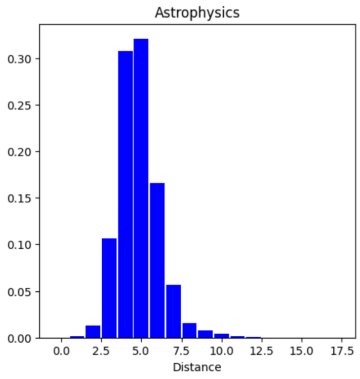
Now, let x be the source node and y be the destination node to find its shortest path distance using the matrix.Le t $L_k = \{lk_1, lk_2, lk_3\}$ be the three selected landmarks near the source node x, these landmark are selected from the distance matrix Mn, Now with these three landmark we select the distance from selected landmarks to the destination node from the distance matrix (Ml) and then we find the minimum of these three landmarks distances and choose the final landmark, after choosing the final landmark we apply approximate distance estimation, this is achieved by taking the sum of distance from source node to selected final landmark and distance from that selected landmark to the destination node and finds the approximate shortest distance path between source and destination nodes.

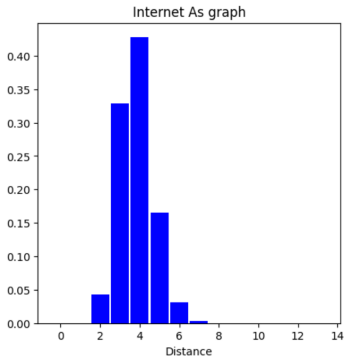
5 Data

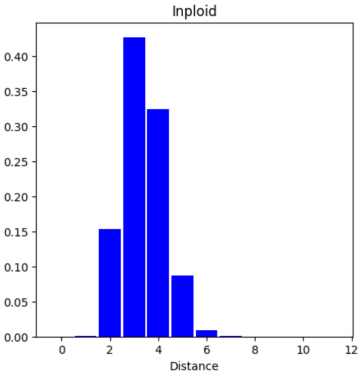
To check the robustness of our methods and to observe their performance, we test it on 5 real world datasets. Three of which are new datasets that were not used in the original paper namely, Scientific collaborations in physics, Inploid network, and Internet AS graph. Then we use two datasets that are similar to the original paper which are Wikipedia and DBLP authors. Now we provide the stastics and details of the datasets.
Scientific collaborations in physics The first graph is an undirected graph. The network coauthorships between scientists posting preprints on the Condensed Matter E-Print Archive, and is extracted from he Los Alamos e-Print arXiv (physics). Each node in the graph represents a scientist and the edges represent if they have coauthored a pre-print. This graph consists of 16,706 nodes and 121,251 edges.
Internet AS graph. This graph is a symmetrized snapshot of the structure of the Internet at the level of Autonomous Systems (ASs), reconstructed from BGP tables posted by the University of Oregon Route Views Project. The nodes are autonomous systems (AS), i.e. collections of connected IP routing prefixes controlled by independent network operators. Edges are connections between autonomous systems. This graph consists of 22,963 nodes and 48,436 edges.
Inploid network. Inploid is a social question and answer website in Turkish. Users can follow others and see their questions and answers on the main page. Each user is associated with a reputability score which is influenced by feedback of others about questions and answers of the user. Each user can also specify interest in topics. The data is crawled in June 2017 and consist of 39,749 nodes and 57,276 directed links between them. In addition, for each user, reputability scores and top five topics are included in the dataset. Usernames and topics are anonymized.
Wikipedia link dynamics. This network consists evolving hyperlink structure among wikipedia articles. Nodes represent the articles, and the edges represent whether an article hyperlinks/references another. This graph consists of 100,312 nodes and 1,627,472 edges. This network is similar to the one used in the original paper[4].
DBLP authors. This network is a A coauthorship network extracted from the DBLP computer science manuscript database, in 2012. This network is a one-mode projection from the bipartite graph of computer scientists and their publications. This network is a co-authorship network where two authors are connected if they publish at least one paper together, that means each node is an author and the edges represent the published paper. This graph consists of 425,957 nodes and 1,049,866 edges. This network is similar to the one used in the original paper.
We have obtained all the datasets used in our project from:
https://networks.skewed.de/
In the below table we present the statistics about the datasets like the number of nodes |V|, the number of edges |E| and clustering coefficient c.

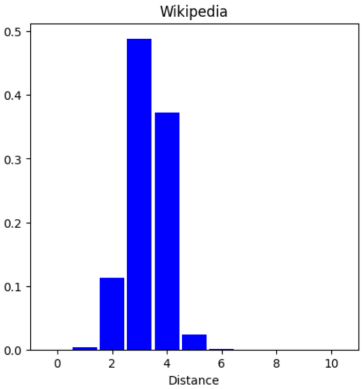
Dataset	|V|	|E|	Diameter	c
Scientific collaborations in physics	167,06	121,251	14	0.89
Internet AS graph	22,963	48,436	11	0.01
Inploid network	39,749	57,276	8	0.02
Wikipedia link dynamics	100,312	1,627,472	12	0.06
DBLP authors	425,957	1,049,866	23	0.31

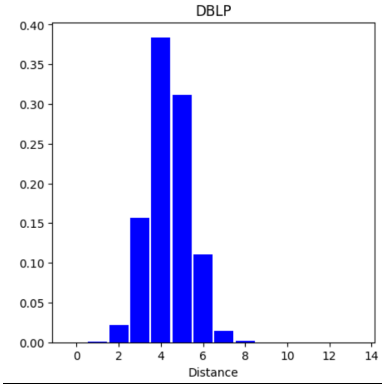
Figures below represent the distance distribution plots of 5 different datasets











6 Experiments

We present experimental results in terms of efficiency, accuracy and comparison to existing work for five dataset. Also to lower the computation load we use a subset of the original graphs of DBLP and wikipedia , which have 12,590 nodes and 49,759 edges for DBLP and 11,381 nodes and 198,275 edges for wikipedia.

6.1 Distance Approximation

We start our experimental process by loading the graphs which are present in an edgelist format. In the first step we apply the popular Louvain community detection algorithm. Louvain Community Detection Algorithm is a simple method to extract the community structure of a network. This is a heuristic method based on modularity optimization. The algorithm works in 2 steps. On the first step it assigns every node to be in its own community and then for each node it tries to find the maximum positive modularity gain by moving each node to all of its neighbor communities. If no positive gain is achieved the node remains in its original community. The first phase continues until no individual move can improve the modularity.

The second phase consists in building a new network whose nodes are now the communities found in the first phase. To do so, the weights of the links between the new nodes are given by the sum of the weight of the links between nodes in the corresponding two communities. Once this phase is complete it is possible to reapply the first phase creating bigger communities with increased modularity.

The above two phases are executed until no modularity gain is achieved (or is less than the threshold).

6.1.1 ***Landmark Selection*** After we have the communities we begin the process of landmark selection. First we define the parameter of selecting the number of landmarks for every community, since the majority of the graphs that we are working with have about 10000 to 40000 nodes, we define this parameter to be equal

to 3. The reason for choosing this parameter to be equal to 3 is because after testing our code on different values, 3 gives the quickest results without increasing the computational load too much and also the results are closer to the actual distances.

6.1.2 ***Calculating Distance Matrices*** Once we get the top nodes based on degree centrality for all the communities we define a function **shortest_path_distance_matrix** for calculating the distances between our nodes and landmarks. Now we start with our matrix formations. The function starts by determining the number of nodes in the graph and the total number of landmarks, ensuring a proper initialization of matrices. We use two matrices to represent our nodes and landmarks. The first matrix is the **distance_node_to_landmark_matrix** and the second one is **distance_landmark_to_node_matrix**.

A nested loop iterates over each node in the graph and each landmark, calculating the shortest path lengths using NetworkX's shortest_path_length function. If a valid path exists between the node and landmark, the distance is recorded in the distance_node_to_landmark_matrix.

A similar process is employed to calculate the distance matrix from landmarks to nodes. The function iterates over each landmark and each node, determining the shortest path lengths and populating the distance_landmark_to_node_matrix accordingly. Finally the function's execution terminates after returning these two matrices.

6.1.3 ***Estimating the shortest Distance*** Finally we start with calculating the shortest path length between any two nodes. We do this by defining a function **estimate_shortest_distance**. At the outset, the function initializes a list to store landmark indices within the community of the source node, which is required for subsequent calculations.

Now we find the community to which the source node belongs and retrieve the landmarks associated with that community. After this we obtain their indices, a crucial step given that distance matrices use indices for landmarks rather than their actual node values. Followed by this, we determine the distance from the source node to the closest landmark.

Having identified the closest landmark, we calculate the distance from this landmark to the destination node.

In the final step, we compute the total estimated shortest distance by summing the distance from the source node to the closest landmark and the distance from that landmark to the destination node.

6.1.4 ***Software and Hardware Used*** We implemented our approach completely on **python** and used it's **NetworkX** package for finding the communities and the shortest path lengths. Our entire code was run on a laptop with AMD Ryzen 5 4500U processor with 7.39 GB of memory.

7 Results

In The following tables we provide the shortest path distance calculations of three Methods:
Actual S.P.D (Actual Shortest path distance between two node pairs, Method1 S.P.D (Method used in the main paper to find the approximate shortest path, Comm detec(The method we used to find the approximate shortest path distance using Community detection)

Node Pairs	Shortest Path Distances (S.P.D)		
	Actual S.P.D	Method1 S.P.D	Comm detec
8289-7565	6	6	9
8551-9119	6	6	6
9104-9412	7	7	10
9351-9901	7	7	8
9394-11913	3	3	4
10039-859	5	5	5
11378-1794	6	7	10
12688-2413	6	7	9
14607-2414	7	7	9
15009-3053	7	7	10

Table 1: Shortest distance path calculation for node pairs in Astrophysics Dataset

Node Pairs	Shortest Path Distances (S.P.D)		
	Actual S.P.D	Method1 S.P.D	Comm detec
5427-500	6	6	7
1833-681	5	5	6
1373-1275	4	4	6
1607-6654	3	4	4
1254-7955	4	5	5
1351-4433	4	5	5
1000-4822	5	5	6
1240-4421	6	7	6
793-7659	5	6	6
1405-8212	4	4	4

Table 2: Shortest distance path calculation for node pairs in DBLP Dataset

Node Pairs	Shortest Path Distances (S.P.D)		
	Actual S.P.D	Method1 S.P.D	Comm detec
14552-3264	3	3	5
245-3380	2	2	4
397-3406	2	2	3
365-3579	3	3	3
62-3746	1	1	2
5716-3786	2	2	3
97-4466	2	2	2
9916-4553	3	3	4
484-4658	3	3	3
123-4788	2	2	2

Table 3: Shortest distance path calculation for node pairs in Inploid Dataset

Node Pairs	Shortest Path Distances (S.P.D)		
	Actual S.P.D	Method1 S.P.D	Comm detec
5427-500	4	4	4
1833-681	4	4	4
1373-1275	3	3	3
1607-6654	3	3	3
1254-7955	3	3	3
1351-4433	3	3	3
1000-4822	2	2	2
1240-4421	3	3	3
793-7659	3	3	3
1405-8212	4	4	4

Table 4: Shortest distance path calculation for node pairs in Internet as graph Dataset

Node Pairs	Shortest Path Distances (S.P.D)		
	Actual S.P.D	Method1 S.P.D	Comm detec
1216-5819	2	3	3
1224-5821	3	3	4
1231-5827	3	4	4
220-8371	3	3	3
1262-4422	3	3	4
323-507	2	3	4
926-508	3	4	5
1344-511	3	3	5
401-512	3	3	5
1371-515	3	5	5

Table 5: Shortest distance path calculation for node pairs in Wikipedia

Datasets	MAE	RMSE
Astrophysics	2.0	2.40
DBLP	0.9	1.04
Inploid	0.8	1.01
Internet-graph	0.1	0.31
Wikipedia	1.4	1.54

Table 6: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) of Actual and Approximate (Community detection method) Shortest path distances

Based on the Errors, Astrophysics dataset has the highest error meaning our method was not able to find the shortest path distance as effectively and Internet-graph dataset has the least error

Datasets	MAE	RMSE
Astrophysics	0.2	0.45
DBLP	0.5	0.70
Inploid	0.0	0.0
Internet-graph	0.0	0.0
Wikipedia	0.6	0.89

Table 7: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) of Actual and Method1 (Method in the orginal paper)

8 Conclusion

Our paper addresses the challenge of efficiently computing shortest-path distances in large graphs, with a focus on the application of landmarks for distance estimations. The central problem revolves around achieving a balance between precision and computational efficiency. To overcome this, we incorporate the method of community detection before landmark selection is introduced.

Experimental results across diverse datasets validate the effectiveness of the proposed approach. The measured Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) for the Inploid and Internet-graph datasets indicate promising outcomes in terms of accuracy and efficiency.

The results showcase the potential of community-based landmark selection in improving the computation of shortest paths in large graphs.

Graphs with a lower clustering coefficient have given better results, Communities in such graphs are more clearly separated making it easier for our community detection algorithm to identify distinct groups of nodes. This separation allows for more accurate landmark selection within communities. Also graphs with lower clustering coefficients have a random or simpler overall structure, This simplicity leads to more accurate landmark selection and, consequently, better approximate shortest path distance calculations. In these graphs communities are less entangled or interwoven, reducing the computational overhead associated with landmark selection within communities. This efficiency contributed faster and more accurate distance estimations during evaluation.

However, on further evaluation of the results it is evident that the distance estimates obtained from our method are quite bigger than the actual distance for the majority of cases. This shows that there are still a lot of different aspects of community detection that we need to examine to further optimize our approach so that we can get much better distance estimates.

In conclusion, these observations reveal that our distance estimates tend to be larger than the actual distances in many cases. This underscores the need for further exploration of various aspects of community detection to optimize our approach. Notably, graphs with lower clustering coefficients yield better results, showcasing the significance of graph structure in our method's performance. Such graphs exhibit clearer community separation, facilitating more accurate landmark selection and, consequently, improved approximate shortest path distance calculations. The simplicity of their overall structure contributes to more efficient landmark selection

and reduced computational overhead, leading to faster and more accurate distance estimations.
Finally, while our approach exhibits promising outcomes, future research should focus on refining community detection methods to further optimize distance estimates, especially in graphs with more complex structures.

8.0.1 ***Future Work*** For our future work, we intend to explore other community detection methods that yield more robust and stabilized community structures. Furthermore we aim to find a much optimized way of selecting the number of landmarks per community which should also help in giving better estimates

Acknowledgments

We would like to express our sincerest gratitude to all the teaching assistants for constantly guiding us and clarifying all our doubts, also for providing us with valuable suggestions to improve our code and report content.

References

[1] Takuya Akiba, Yoichi Iwata, Ken-ichi Kawarabayashi, and Yuki Kawata. 2014. Fast shortest-path distance queries on road networks by pruned highway labeling. (2014), 147–154.
[2] Reinhard Bauer, Daniel Delling, Peter Sanders, Dennis Schieferdecker, Dominik Schultes, and Dorothea Wagner. 2008. Combining hierarchical and goal-directed speed-up techniques for Dijkstra's algorithm. (2008), 303–318.
[3] Amgad Madkour, Walid G Aref, Faizan Ur Rehman, Mohamed Abdur Rahman, and Saleh Basalamah. 2017. A survey of shortest-path algorithms. *arXiv preprint arXiv:1705.02044* (2017).
[4] Michalis Potamias, Francesco Bonchi, Carlos Castillo, and Aristides Gionis. 2009. Fast shortest path distance estimation in large networks. (2009), 867–876.