Kashf Journal of Multidisciplinary Research
Vol: 02 - Issue 11 (2025)
P-ISSN: 3007-1992 E-ISSN: 3007-200X
https://kjmr.com.pk

TIME COMPLEXITY ANALYSIS OF SINGLE SOURCE SHORTEST PATH (SSSP) ALGORITHMS

[1]Khalid Nooruddin Charan*, [2]Saad Akbar, [3]Raza Hussain Shah, [4]Mohammad Ayub Latif, [5]Abdul Razaque

[1, 2, 5]Department of Computing, Faculty of Engineering, Science & Technology (FEST), Hamdard University, Karachi, Pakistan.

[3]Department of Computer Science, SZABIST University, Hyderabad, Pakistan.

[4]College of Computing and Information Sciences, Karachi Institute of Economics and Technology (KIET), Karachi, Pakistan.

****Corresponding Author:*** khalid.chaaran@hamdard.edu.pk

Article Info

Abstract

There are real-world complex networks formed by people, roads, communication network nodes, genes, file Servers and financial transactions etc based on their interdependent associations. Often, there exist multiple paths to connect one of the network nodes to another in the networks. Various situations demand shortest path to reach from one node to another. To find shortest path between a pair of the nodes is challenging in terms of computational cost, therefore, competing algorithms for Shortest Path problem are analyzed to predict their greediness for the computing resources such as memory, bandwidth, or hardware but most often its computational time. The scope of this work is to provide comprehensive knowledge on single-source shortest path problem and analyze time complexity of the prominent single-source shortest path algorithms.

Keywords: *Single-Source Shortest Path (SSSP), Computational Complexity Analysis, Graph Algorithms, Network Optimization*

1. INTRODUCTION

There exists countless real-life phenomenon where a set of independent entities form a complex network of relationships based upon any of their interdependency. Often, there exist multiple paths connecting the networks nodes among each other. In various real-world situations, there arises a dire need to ascertain a shortest link between two members of the network. Ascertained shortest link is termed as Shortest Path that is the path with minimum total cost among all existing alternate paths. A well-defined computational procedure called algorithm, a sequence of computational instructions that transform the input into the output [1], is required to solve Shortest Path problem.

All scientific activities including computation depends on mathematical modeling of real-world phenomena, even though they are only approximate idealizations [2]. A graph is an abstract representation of mathematical model that can be used to express complex computation phenomenon with clarity and precision in a concise pictorial language [3]. A graph contains a set of vertices and a set of edges joining the vertices; therefore, it can be used to transform a phenomenon of objects as vertices and associations / relationships as edges to determine a shortest path between these objects.

The Shortest Path problems are graph-searching algorithms that identify the optimal route with minimum cost from a given starting vertex to the target vertex. Design of an efficient algorithm to find out shortest path in a graph is an ongoing research challenge [4]. Graph algorithm works on graphs to provide solution to a problem represented by the graphs [5]. The solution can be identified based on various settings of the graph. For example, the graph can be static, where the vertices and the edges do not change over time, and in contrast, a graph can be dynamic, where vertices and edges can be introduced, updated, or deleted over time. The graph contains either directed or undirected edges and the edges may carry weights, moreover, the weights can either be non-negative or negative and their values can be real or integer numbers. These are the different constraints posed by the problem being studied [6].

Shortest Path algorithms implement various searching approaches to find the minimum distance (cost) and each approach generates different performance aspects, therefore, an algorithm is analyzed to discover its performance characteristics to evaluate its suitability for various applications or to compare it with other algorithms for same application. Analysis of algorithm means to predict its greediness for resources such as memory, bandwidth, or hardware but most often the computational time. The terms “theory of algorithms” and “scientific approach” are used to describe analysis of algorithms and to classify the algorithms as per their performance characteristics [1, 2].

To find out shortest path is computationally costly in terms of time and space requirements. Prediction of the greediness of an algorithm for computational time is termed as Time Complexity of an algorithm. Scope of this research is to review time complexity of the single source shortest Path algorithms [8]. Although this review might show that there is only a slight time complexity variations, however, these small differences become crucial in bigger graphs.

The rest of this paper is organized in the fashion such that section II briefly presents Shortest Path applications followed by the basics of graph theory in Section III. The section IV of paper explains the analysis of algorithms and covers Shortest Path algorithms. The section V & VI provide time complexity

analysis of Dijkstra's and Bellman-Ford algorithms, respectively. Finally, section VII provides conclusions.

2. Applications of Shortest Path Algorithms

Applications of Shortest Path algorithms encompass multiple disciplines such as social sciences, biology, technology, chemistry, neuroscience, software engineering, security, logistics & planning, and finance [9] etc.

2.1 Digital mapping services in Google Maps / GIS:

Distance from one city to another or from a location to another desired location, selection of network path topology, GIS and track planning are every day computational tasks. In such computational tasks, there are multiple routes / paths connecting the entities under certain constraints like obstacles avoidance and boundary control conditions such as navigation mission, maneuverability and geomagnetic navigation adaptability. Shortest path algorithms serve as backbone of these computations. Moreover, in addition to 3D geographic information network analysis shortest path algorithms have a wide range of applications in path analysis, location analysis, and cost analysis. A lot of research has undergone for shortest path visualization on the vector data [10, 11, 12 and 13].

2.2 Social Networking

Social network is a structured collection of dyadic associations between social entities such as persons / groups/ organizations. Social networking has attracted a lot of attention from different fields of study including sociology, informatics, and computer science. Social influence, social groupings, inequality, disease propagation and communication of information are today's research topics. Shortest path is an extremely important tool for social network analysis, for example, it can be used to study the behavior of information spreading, especially fastest information spreading.

People maintain variety of social relationships like family, friends, colleagues, neighbors, and different other associations such as jobseekers and professional these relationships vary from weak to strong in strengths. Weak ties are extremely important in social networks. People generally turn down the invitation of unknown person (weak tie), which result in breaking of a link and loss of information. Boštjan Šumak and Maja Pušnik analyzed the application of Dijkstra and Bellman-Ford shortest path algorithms in social networks as a strategic attempt to speed the search of network nodes [14].

Singh et al. [15], analyzed the collaboration network structure for betweenness and average gap between the relationships of scientists, and recommended that the research work and paper reviewing efficiency can be improved based on the shortest path. LinkedIn or Facebook, exponentially increasing social network models, represent huge entities and their relationship. LinkedIn connects professionals worldwide. The website finds the shortest path to reach a desired person by starting from inquiring person's connections and moving on to friends of the friends [16].

2.3 Telecom Networks / File Servers

In telecom networks / File servers such as IP Routing, Telephone Network and Content Delivery Networks (CDN), routers and servers are considered as entities and factors such as bandwidth, physical distance or similar metrics are treated as relationships between these entities. Cloud computing triggered the rapid expansion of data center networks (DCN). DCN adopted conventional link state routing algorithms. Most traditional Software Defined Network (SDN) controllers rely on the Shortest Path First (SPF) algorithms for routing decisions. The controller obtains current network topology through the Link Layer Discovery Protocol (LLDP)-based topology detection mechanism. Whenever the network topology changes, the controller recalculate shortest path(s) between nodes [17].

2.4 Business and Marketing

Yinbo et al. [18] suggested an efficient shortest path algorithm for a directed weighted graph in which cost of all arcs depends on the source node. The algorithm can be applied to optimal assortment problem for a network good industry with vertical differentiation. The assortment problem was introduced to select what products to be produced or stocked from a large product category when it is not desirable to stock all of them. The research revealed that the algorithm can identify an optimal assortment plan in polynomial time.

2.5 Technology

The routing in VLSI design is very crucial and time consuming. VLSI design instance graphs consist millions of shortest paths in billions of the vertices. Therefore, instance specific algorithm, which heavily exploits the instance structure, may offer an acceptable running time [19]. Bao Zhang et al. [20] claimed a novel method based on shortest path theory for unsupervised object-level video object segmentation and compared it with the state-of-the-art methods. The proposed method demonstrated that it performed well when foreground object is interfered by other fast-moving objects, however, failed when the foreground object misses in many video frames.

2.6 Road Networks

Computing the shortest travel routes between cities is a fundamental problem in road networks. Shortest path based methods are proposed in the literature for efficient shortest path query processing. A shortest path algorithm based on MapReduce proposed in [21] which uses widely existing vehicle data. The Δ-stepping algorithm [22] successfully implemented for distributed memory architectures in which Dijkstra's algorithm is divided into phases and each phase is executed in parallel. Daniele et al. [23] carried performance analysis of Δ-stepping algorithm considering threading, load balancing and synchronization parameters with focus on the Game-Map graphs commonly used to represent the physical environment for trajectory planning of autonomous vehicles.

2.7 Medical Sciences

Ryan et al. [24] concluded that Dijkstra's Shortest path algorithm analyses lead to many new discoveries in structural biology which were not previously possible. Applications include protein shape changes due

to increased thermo-stability, finding, and examining ion channels and pores, the depth of binding sites and how this affects binding affinity, and finally finding and comparing the shapes of pockets. While analyzing the distances from these various surfaces to each other or into the molecule or solvent has led to many new and different applications from the original surface analyses, moreover, there remain many avenues for inquiry into macromolecular shape and biological function.

Recent studies in medical sciences have implemented network analysis and shortest path analysis in understanding brain functioning, cancer genetics to drug development and so on. A shortest path network, a special weighted network, is constructed from the functional brain network to understand the intricacies of brain functioning. Co-expression networks (GCN) are developed in genetics to represent relationships between different genes. The “load points” analysis of metabolites in a metabolic network is based on ratio of the number of valid k-shortest path passing through the metabolites and its nearest neighbor connectivity [25, 26 and 27]. Several directed relationships, for example “inhibits,” “enhances,” “regulates” etc., in the field of biology can be shown graphically. Systems Biology Graphical Notation (SBGN) visual language describes standards for arrow usage [28].

2.8 Neuroscience

M. Thilaga et al. [29] analyzed the shortest paths, computed using the threshold network of the fully connected Functional Brain Network (FBN) that includes only the influential connections (connections over which the amount of information exchange is high), to identify the brain’s information communication patterns during mild and heavy cognitive load states. Possible shortest paths between all pairs of nodes and number of occurrences of each the edge is calculated to construct Shortest Path Network (SPN) using these influential connections. SPNs of mild and heavy cognitive load states are further analyzed heuristically that highlights the Connection Density (DC) i.e. the edges occurring in the shortest paths many times of the respective state. Moreover, the SPN based FBN analysis technique can be extended further to identify community structures, the connected components, to analyze and understand the topology of SPN networks for different cognitive load state.

2.9 Logistics

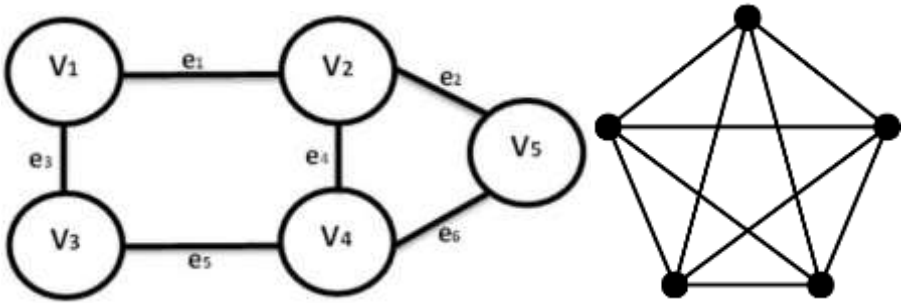
Logistics distribution vehicles shortest route is the goal for supply chain management efficiency. Zhang et al. [30] concluded that shortest path is one of the most common problems in medium or large logistics enterprises. The metrics including geographical shortest distance, time, cost, line capacity, and traffic congestion and so on may affect distribution path. Cost control and profit maximization distribution route can be transformed to the shortest path problem by setting weight for each of the factors.

Advanced shortest path methods model the optimality of the path amongst possibilities [31, 32] therefore, plays important role in trajectory planning. Shortest path algorithms establish linear correlation between shortest path length and the water age for optimal water distribution [33] and crime spots and police stations [34]. A very little material is referred above from a plethora available in research that establishes the need to calculate shortest path in variety of context with diversity of applications in different domains.

3. Graph Theory and Shortest Path Algorithms

Pictorial representation provides convenience in analysis of a complex phenomenon. Graphs are used across many application domains to interpret complex networks of relationships formed by people, roads, financial transactions, etc. as simple yet effective medium to model real life phenomenon [9].

Tow vertices u and v of a graph G are connected if there exists an edge e between the vertices. A complete graph Kn contains n(n − 1)/2 edges as show in Fig. 1(b). The complete graphs pose the worst case problems to the shortest path algorithms. The degree of a vertex v in a graph G, denoted by dG(v) or deg v or simply d(v), is the number of edges incident with v. Vertex v of degree 0 is called an isolated vertex and vertex v of degree 1 is called a pendant vertex. Sum of the degrees of the vertices of a graph G is twice the number of edges [35].



(a) **(b)**

Figure. 1. (a) A simple graph (b) A complete graph

A weighted graph is a graph in which E, set of edges between the vertices u and v (E = {(u, v) | u, v ∈ V}), is associated with a weight function w:E → R, where R denotes the set of all real numbers. Most of the times, the weight wuv of the edge between nodes u and v represents the relevance of association [36]. The graphs are stored in computer memory using data structures such as Adjacency Matrix, Incidence Matrix and Adjacency List [37], and these graphs storage techniques play vital role in graph traversal algorithms from the computation prospect.

Adjacency Matrix Adjacency matrix A(G) of graph G, a square matrix of size N × N where N is the number of vertices, is illustrated in Table 1 for the simple graph in Fig. 1(a). It indicates presence of connection A[u,v] = 1 and absence of connection A[u,v] = 0 and A[u,v] = wuv for a weighted graph where wuv is weight of the edge between adjacent nodes. The simple graph matrix denotes zeros diagonally. The adjacency matrix is symmetric (transpose-rows and columns are equal) for undirected simple or weighted graphs and non-symmetric for directed graph, thus differentiating its upper and lower part of its diagonal line (uv is not the same as vu).

Table 1: Adjacency matrix of the simple graph in Fig. 1(a).

	V1	V2	V3	V4	V5
V1	0	1	1	0	0
V2	1	0	0	1	1

V3	1	0	0	1	0
V4	0	1	1	0	1
V5	0	1	0	1	0

Incidence Matrix An incidence matrix I(G) of G, a M × N (where N is the number of vertices and M is the number of Edges), is illustrates in Table 2 for the simple graph in Fig. 1(a). The matrix M(G) = [mij] represent mij = 1 if the vertex vi is an incident to the edge ej and 0 otherwise.

Table 2: Incidence matrix of the simple graph in Fig. 1(a).

	e1	**e2**	**e3**	**e4**	**e5**	**e6**
V1	1	0	1	0	0	0
V2	1	1	0	1	0	0
V3	0	0	1	0	1	0
V4	0	0	0	1	1	1
V5	0	1	0	0	0	1

Incidence matrix is memory greedy than adjacency matrix, specially, when number of edges is higher than vertices. Neighborhood scanning takes n steps for all the vertices even if deg(v) is much less than n, however, the query for incidence of an edge responded in constant time.

Adjacency List an adjacency list Adj(G) of G is a collection of n lists where there is a list for each vertex v of G consisting record of each of its neighbors. Adjacency list is a space efficient representation in comparison to adjacency matrix and incidence matrix, particularly when the input graph is not a complete graph. Fig. 2 illustrates the adjacency list.

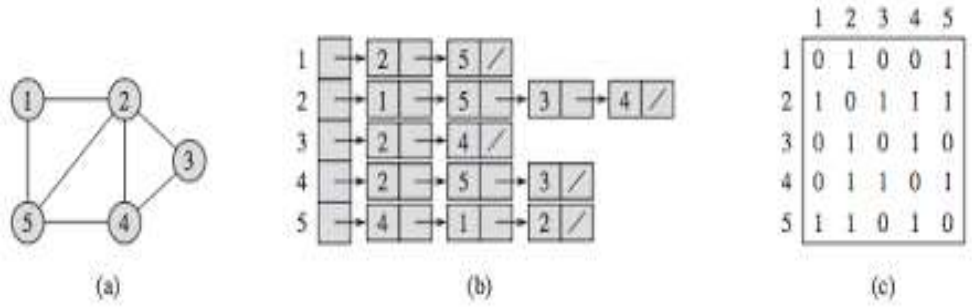


Figure. 2. Adjacency List

Graph attributes representation Algorithms need to maintain attributes for vertices and/or edges to operate on graphs. Notations, such as v.d are used for an attribute d of a vertex v, similarly, an attribute, say f, of an edge (u, v) is represented as (u,v).f. In a digraph G = (V, E), the predecessor list Pv ⊆ V and successor list Sv ⊆ V of a vertex v are defined and listed (Pv and Sv list (fig.9) of the digraph in fig.10) as follows:

$P_v = \{u \in V : (u, v) \in E\}$

$S_v = \{u \in V : (u, v) \in E\}$

Table 3: P_v and S_v lists of vertices in Fig. 5

V	Pv	Sv
1	{2}	{2,3,4}
2	{1,5}	{1,5,4}
3	{1,4}	{4}
4	{1,2,3}	{3,5}
5	{2,4}	{2}

4. Analysis of an Algorithm

Regardless of how fast computers become or how cheap memory gets, efficiency of algorithms will always remain an important consideration [38]. Time complexity analysis of algorithms. Analyzing an algorithm discovers its characteristics to evaluate its suitability for various applications or to compare it with other algorithms for the same application. To determine how efficiently an algorithm solves a problem, it is to analyze the algorithm and by analyzing several candidate algorithms for a problem the most efficient one can be identified. Such analysis may indicate more than one viable candidate, but we can often discard several inferior algorithms in the process. Computational complexity deals with inherent cost of solving an information processing problem. Analysis of algorithms is a special case of computational complexity. The cost is measured in terms of various well-defined characteristics of interest and the most primary characteristics are the time and the space, particularly the time [38]. The prime goal of analysis of algorithms is to estimate time and space requirement of the algorithm as function of the input size, particularly, to determine the asymptotic behavior of the function, specially, with very large input size [2].

A reasonable measure of the input size is the number of input elements for algorithm. For example the n number of items in the input array A[] of the algorithm A given below in figure 3, is a simple measure of the size of the input, therefore, n is called the input size of the algorithm. However, at times, size of the input using two numbers is more appropriate to measure input size, for example when a graph, consists of vertices and edges, is the input to an algorithm, in such a situation the input size consists of both parameters.

```
Algorithm A (A[], x)
1  for i = 1 to A.length
2 // Total elements of A[] are denoted by A.length)
3      if A[i] = x
4      return TRUE
5  return FALSE
```

Figure. 3. Algorithm A

After determining the input size, the most important factor to determine is the basic operation of an algorithm. There is an instruction or group of instructions, such as comparison instruction in algorithm A, which is performed as a basic operation in the algorithm. Efficiency of the algorithm is evaluated by determining how many times an algorithm performs this basic operation for each value of n. Generally,

the total work done by the algorithm is roughly proportional to the number of times the basic instruction or group of instructions is performed.

Consider an instance of a problem given in Figure 4, such that: is there key x in a sorted array A[] of n elements (say 32 elements)?

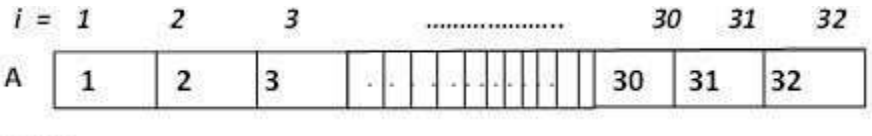


Figure. 4. Algorithm A

Let x = 35, then the algorithm A, based on sequential search technique, searches the array A[] sequentially and performs n number i-e 32 of comparisons to determine that key x is not in the array A[]. Similarly, if x = 32, the algorithm must perform 32 i-e n number of comparisons to determine that key x is present in the array. If T(n) is a function that denotes the number of times the basic operation is performed than:

T(n) = n (Basic operation is performed n times)

It is important to note that above instance of the problem poses worst case scenario, in which algorithm A has to compare all the elements of the array A. The time complexity of worst-case scenario is termed as worst-case time complexity. Time complexity of algorithm A is said to be a function of input size as T(n) = n.

In another instance, let x = 1, then algorithm A will return result after its first iteration as A[1] = 1, here the algorithm compared only one element of the input array A[]. This is the best-case scenario and time complexity is termed as best-case time complexity and the basic operation is performed:

T(n) = 1 (Basic operation is performed 1 time)

Now suppose x is randomly chosen and algorithm A is executed on its input array. How long will it take to determine whether x is there in the array or not? On average, half the elements in A [1. . n] are less and half the elements are greater than A[n/2], therefore, at the average sequential search iterations would be about n/2 thus the resulting average-case running time turns to be a linear function of the input size, just like the worst-case running time. The scope of average-case analysis is limited because it may not be apparent what constitutes an “average” input for a particular problem, although technique of probabilistic analysis applied although a randomized algorithm, which makes random choices, to allow a probabilistic analysis and yield an expected running time [1].

Since worst-case performance provides assurance that the algorithm will never take anymore longer time, therefore, theory of algorithms is mostly interested in worst-case performance characteristics of an algorithm no matter what is the input size [2]. Worst- case defines an upper bound running time of an algorithm for any input. The worst case occurs often for some algorithms, for example searches for absent information may be frequent in some applications therefore, worst case will often occur for searching algorithm when the information is not present in the database [1].

```
Algorithm B (A[], x)
1  int min = 1
3  int max = A.length
4  while (max ≥ min)
5      int mid = [(max + min)/2]
6      if (A[mid] = x)
7          return TRUE
8      else  if (x < A[mid])
9                  max = mid - 1
10         else
11             min = mid + 1
12 return FALSE
```

Figure. 4. Algorithm B

Algorithm A is analyzed so far. Let's consider another algorithm B given in figure 5. The algorithm is designed on binary search technique, in which A[mid] is compared in each iteration. If x < A[mid], only lower half of the array will be considered in next iteration and vise-versa, so half of the remaining input elements are eliminated form comparisons in each iteration, means input elements for subsequent comparison are reduced to half per iteration. Therefore, algorithm B does six comparisons when x = 35 or 32. Notice that 6 = (log2 32) + 1. Hence the worst-case time complexity of algorithm B is:

T(n) = lg n (Basic operation is performed lg n times)

Analysis of the algorithms A and B proved that algorithm B is superior in comparison to algorithm A in terms of time complexity for same application.

Asymptotic notations, adapted from classical analysis [39], primarily describe and bound the running times performance of algorithms. The O-notation is used to denote an upper bound; the **Ω** - notation denotes a lower bound; and the Θ-notation provides a way to express upper and lower bounds.

4.1 Categories of Shortest Path Algorithms

Shortest Path algorithms fall into two broad categories (a) Single-Source Shortest Path (SSSP) where the objective is to find shortest path from a single source vertex to all other vertices (b) All-Pairs Shortest Path (APSP) where the objective is to find shortest paths between all pairs of vertices in a graph. Shortest

Path algorithms are given a graph and a start vertex and asked to find the shortest paths from start vertex to every/any other vertex in the graph [6, 8, 1]. The SSSP algorithm can solve many other problems, including [1]:

- **Single-destination shortest-paths problem:** The algorithms find shortest path to a given destination vertex t from each vertex v. Reversing the direction of edges in the graph, can reduce this problem to a single-source problem. Consider an instance of road networks of six cities, in which it is to find shortest distance to city A from rest of the five cities. Here, instead of computing shortest distance from each the city-to-city A, it is convenient to compute shortest distance from A to all the cities using Single-Source Shortest Path algorithm just by reversing the direction of edges in the graph.
- **Single-pair shortest-path problem:** Finds a shortest path from u to v for given vertices. A solution for the single-source problem with source vertex u, can this problem also.
- **All-pairs shortest-paths problem:** Finds a shortest path from vertex u to v for every pair of vertices u and v. This can be solved by running a single-source algorithm once from each vertex.

In a problem statement such that given a set of N vertices (objects), ordinary numbered from 1 to N and N x N matrix D. The matrix does not necessarily symmetric, whose elements dij represents the weight (cost) of the directed arc (association / relationship) connecting vertex i to vertex j, find the path of shortest length connecting vertex 1 and N. Assume initially that dii = 0, dij ≥ 0. If no arc is directed from vertex i to vertex j, then dij = ∞; or, for the purpose of digital computation, dij is taken large [40].

In Fig. 5 there are multiple paths from v2 to v3, such as [v2, v4, v3], [v2, v1, v3], and [v2, v1, v4, v3] which means there are more than one candidate solutions and each solution has an associated value. The shortest path from v2 to v3 is [v2, v4, v3] that has an optimal value therefor it is the solution to the instance. It is obvious that shortest paths problem is an optimization problem where an optimal value solution is the solution to an instance of the problem. The optimal value is either minimum or maximum depending on the problem. In Shortest Paths problem the associated value is the length of the path from source vertex to target vertex and the optimal value solution would be the path with minimum length.

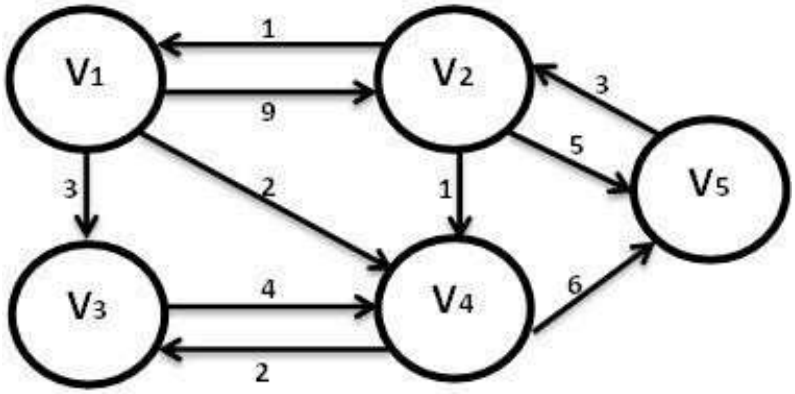


Figure. 5. Directed Weighted Graph

Algorithm for shortest path problem would determine the lengths of all the paths from source vertex to each other vertex and will compute the minimum of these lengths. The algorithm will result in worse than exponential-time for a complete graph, where the number of paths from first vertex to another vertex is the total of the paths that start at the first vertex, end at the other vertex while passing through all other vertices in the graph. Subsequently, at the second vertex there will be n − 2 vertices, at the third vertex there will be n − 3 vertices and so on. At the second to-last vertex there would be only one vertex, the total number of paths from one vertex to another vertex that pass through all the other vertices is (n-2) (n-3) ……1 = n!, which is worse than exponential. This same situation arises in many optimization problems therefore; the algorithm that considers all possibilities is exponential-time or worst, whereas the goal is to find a more efficient algorithm [38].

The quest for a linear-time SSSP algorithm on directed graphs with weighted edges is an ongoing hot research topic [41]. Several informed (heuristic) and uninformed (optimal) algorithms such as Breadth First Search (BFS), Dijkstra's algorithm, Symmetrical Dijkstra's algorithm, A* algorithm, Bellman-Ford algorithm, Floyd-Warshall algorithm and Genetic algorithm are frequently used to discover shortest path [42]. However, this article restricts the scope of to the uninformed algorithms, only, because it seems rational to compare the similar category at a time. The BFS is ideal solution for unweighted or the graphs with identical weighted edges. The BFS algorithm operates in O(V + E) time complexity [43].

5. Dijkstra Algorithm

The Dijkstra algorithm calculate the shortest paths from the given source to all other nodes in the graph, increasing node by node to get a shortest path tree and finds path of minimum total length between source and target nodes [44, 45 and 46].

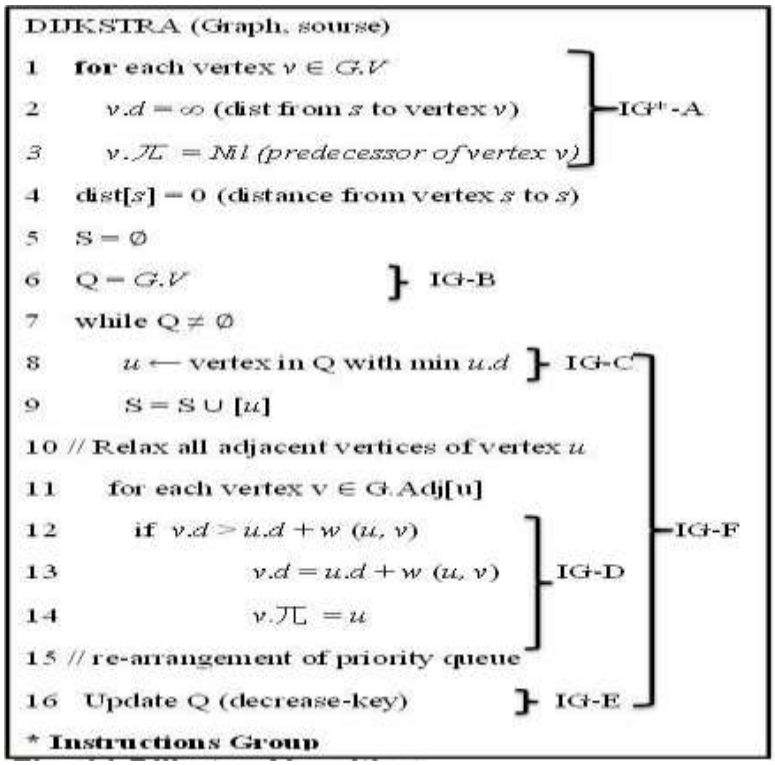


Figure. 6. Dijkstra's Algorithm

5.1 Time Complexity analysis

As discussed earlier, in addition to input size it is most important to determine an instruction or group of instructions as basic operations of an algorithm. Instructions Groups (IG) are marked as IG-A, IG-B and so on in Dijkstra algorithm above. The IG-A initializes the shortest-path-estimates (v.d) and predecessors (v.π) in O(V) time. IG-B builds minimum-priority-queue Q in O(V) time. The selection of a node with minimum values for subsequent iteration is performed at IG-C. IG-D is performed a total of |E| times within all iterations of while loop as relax process is performed at each vertex thru each of its edge. Priority queue decrease operation is performed at IG-E during each iteration of while loop. It is important to note that IG-C thru IG-E are fallen within while loop at IG-F.

The selection of subsequent node is based on minimum priority queue technique therefore an efficient priority queue implementation reduces the asymptotic behavior of Dijkstra's algorithm. So, the variants of Dijkstra's algorithm competing for running time performance implement the minimum priority queue using different methods. Relax operation at IG-D tighten an upper bound that affects the running time performance of the Dijkstra's algorithm. The efficient time complexity achieved so far is O(E+VlgV) using Fibonacci heap, however the time complexity can be reduced to O(E + V lg V) using Integer priority queue.

Array Indexed-by-Vertex: Let the graph G is represented as an adjacency matrix and priority queue Q is represented as an unordered list. Here, A[i,j] stores the information about edge (i,j). Priority queue is implemented by an array indexed by vertices numbered from 1 to |V|. In such implementation, the time taken for updating dist[j] for each neighbor of node i is O(1). However the time taken at IG-C for selecting node i with the smallest distance costs O(V) time within each iteration of the while loop, as there is 1 iteration for each vertex therefore there would be |V| number of iteration that will lead to O(V2) time. Scanning of neighborhood at each vertex takes O(|E|) time and one vertex is deleted from Q. Thus, the sum of running time of each instructions group would be the total time complexity for a graph of V vertices [1]:

O(V) + O(V) + O (1) + O(E) + O(V(V))

O(V2+ E)

O(V2) * according to the order of the growth

Binary Heap: However, if the given graph G is represented as an adjacency list and priority queue Q is represented as a binary heap then all vertices of the graph can be traversed using BFS in O(V + E) time. A heap can support any priority-queue operation like extract-min and decrease-key value and takes O(lg V) time on a set with

|V| vertices, so, the overall time complexity would be [47, 48]:

O(V) + O(V) + O(V lg V) + O(E lg V)

O(E lg V)

D-ary (D-way) heap: Minimum priority queue implementation using D-ary heap insert and Heap-decrease-Key operations cost O(lg V / lg d) and extract-min takes O (d lg V / lg d), therefore total running time for |V| nodes is [49]:

O((V . d + E)lg V / lg d)

Fibonacci Heap: The maximum for a graph with V vertices the heap size is |V|-1. A Fibonacci-heap, will take O(V lg V + E) time for heap operations and O(V + E) time for other tasks thus running time for Dijkstra's algorithm[1, 50, 8] would be reduced to:

O(V lg V + E)

Integer priority queues: Integer priority queue where a key can be deleted in O(lg lg V) time and O(1) time for other operations [51], the time complexity is achieved as:

O(E + V lg lg V)

A concise view of the variants of Dijkstra's algorithm competing for better time complexity using different priority techniques are appended in Table 3 below:

Table 4: Different priority Queue implementation Time Complexity (Dijkstra Algorithm)

Priority Queue implementation	Total running time cost
Node-indexed array	$O(V^2)$
Binary heap	$O(E \lg V)$
D-way heap	$O((V \cdot d + E) \lg V / \lg d)$
Fibonacci Heap	$O(V \lg V + E)$
Integer priority queue	$O(E + V \lg \lg V)$

6. Bellman-Ford Algorithm

The Bellman-Ford algorithm, shown in figure 7, finds single source shortest paths in a graph with even negative edge weights (but no negative cycles). The algorithm has inherent ability to detect negative cycles. Bellman-ford relaxes minimum-path-estimates iteratively and finds guaranteed minimum path from source node to all other node in in |V|-1 iterations for a graph of |V| nodes.

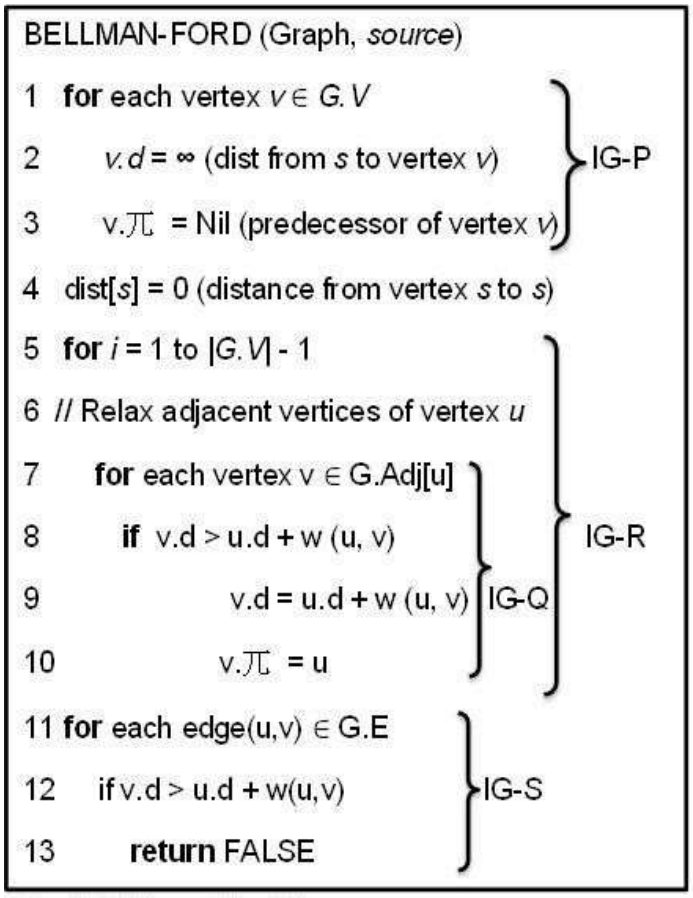


Figure. 7. Bellman-Ford Algorithm

6.1 Time Complexity Analysis

Bellman-Ford algorithm also consists of initialization phase and Relax operation. The IG-P initializes the shortest-path-estimates (v.d) and predecessors (v.π) in O(V) time relax operation is performed at IG-Q on each vertex within IG-R, an outer loop of every iteration. IG-S is performed O(\E\) time to detect negative cycle in graph G, if exist any. Time complexity of Bellman-ford algorithm for sparse graph is O(EV) = O(V2), whereas it is O(V3) as E = ((V. V-1)/2) for complete graphs. Like Dijksrta's algorithm, selection of a node for subsequent execution iteration plays important role in the running time performance for Bellman-Ford algorithm as well. Therefore, variants of Bellman-Ford algorithm employed various queuing techniques to achieve better performance.

The D'Esopo-Pape Algorithm: Based upon D'Esopo [52], Pape et al. [53] proposed a deque (a queue that allows insertions at either ends) method for selection of node for subsequent iterations. The D'Esopo-Pape Algorithm runs in O(V. 2v) time.

The Goldfarb-Hao-Kai Algorithm: The algorithm proposed a dynamic Breadth-First Search (BFS) that maintains depth factor of each vertex and decides accordingly [53]. The Goldfarb-Hao-Kai algorithm runs in O(V. E) time.

Pallottino's Algorithm: Pallottino's algorithm uses the data structure composed of two connected queues Q1 and Q2. The next vertex to be scanned is removed from the head of Q1 until Q1 is not empty and from Q2 otherwise [55]. Pallottino's algorithm runs time complexity is O(V2. E).

The Goldberg-Radzik Algorithm: Goldberg et al. [56] proposed a topological-scan scheme. The proposed algorithm achieves the time complexity of O(E V) that is similar to the Bellman-Ford algorithm.

A concise view of the variants of Bellman-Ford algorithm competing for better time complexities using different techniques are appended in Table 5 below depicts a concise view:

Table 5: Time Complexity of the variants of Bellman-Ford

Different Techniques Implementation	Total Running Time Cost
Straight forward	$O(V^2)$
D'Esopo-Pape algorithm	$O(V. 2^V)$
Goldfarb-Hao-Kai algorithm	O(V. E)
Pallottino's algorithm	$O(V^2. E)$
Goldberg-Radzik algorithm	O(EV)

7. Conclusion

This paper presented review of BSF, Dijkstra and Bellman-Ford algorithms, the classical algorithms for SSSP problem. In addition to BSF, different variants of Dijkstra and Bellman-Ford algorithm were compared amongst each other, respectively, with reference to their time complexity. It is evident from the review that selection of a node for subsequent execution iteration and node relaxing operation are basic operations in the reviewed algorithms, thus affect the running time performance that tighten upper bound time complexity of the algorithms. Therefore, any proposal for reductions in these basic operations may, consequently, reduce running time of the algorithms an in turn may results in better time complexity as well.

References:

[1] Thomas H. Corman, Charles E. Leiserson, Ronald L. Rivest, Clifford stein (2022). INTRODUCTION TO ALGORITHMS 4th Ed The MIT Press Cambridge, Massachusetts London, England.

[2] Robert Sedgewick and Philippe Flajolet (2013). Introduction to the Analysis of Algorithms, 2nd Edition. Addison-Wesley Professional.

[3] Corso, G., Stark, H., Jegelka, S., Jaakkola, T., & Barzilay, R. (2024). Graph neural networks. Nature Reviews Methods Primers, 4(1), 17.

[4] Samah W.G. AbuSalim, Rosziati Ibrahim, Mohd Zainuri Saringat, Sapiee Jamel and Jahari Abdul Wahab (2020) "Comparative Analysis between Dijkstra and Bellman- Ford Algorithms in Shortest Path Optimization". International Conference on Technology, Engineering and Sciences (ICTES) 2020. IOP Conf. Series: Materials Science and Engineering 917 (2020) 012077 IOP Publishing. doi:10.1088/1757-899X/917/1/012077.

[5] Medová, J., Milicic, G., & Ludwig, M. (2022, February). Graph problems as a means for accessing the abstraction skills. In Twelfth Congress of the European Society for Research in Mathematics Education (CERME12) (No. 07).

[6] Amgad Madkour1, Walid G. Aref, Faizan Ur Rehman, Mohamed Abdur Rahman, Saleh Basalamah (May 8, 2017). A Survey of Shortest-PathAlgorithms. arXiv.org>cs>arXiv:1705.02044v1.

[7] Hector Ortega-Arranz, Diego R. Llanos, and Arturo Gonzalez-Escribano (2015). "Shortest-Path Problem" Analysis and Comparison of Methods, Synthesis Lectures on theoretical computer Science.

[8] Tim Hegeman and Alexandru Iosup (2018). Survey of Graph Analysis Applications. arXiv:1807.00382v1 [cs.SI].

[9] Hasan, R. A., Akawee, M. M., & Sutikno, T. (2023). Improved GIS-T model for finding the shortest paths in graphs. Babylonian Journal of Machine Learning, 2023, 7-16.

[10] Mahariba, A. J., Uthra, R. A., & Brunet, R. G. (2022). Estimation of shortest route with minimum travel time using GIS and MSSTT algorithm. In Advances in Construction Management: Select Proceedings of ACMM 2021 (pp. 565-579). Singapore: Springer Nature Singapore.

[11] Yan, Y., & Wong, R. C. W. (2021). Path advisor: a multi-functional campus map tool for shortest path. Proceedings of the VLDB Endowment, 14(12), 2683-2686.

[12] Chetan Chadha and Shivang Garg (2019). Shortest Path Analysis on Geospatial Data Using PgRouting Visualization of Shortest Path on Road Network. Springer Nature Singapore Pte Ltd.

[13] Šumak, B., & Pušnik, M. (2023). Analysis of the shortest path method application in social networks. In Information Modelling and Knowledge Bases XXXIV (pp. 169-182). IOS Press.

[14] Singh, S. S., Muhuri, S., Kumar, S., & Barua, J. (2025). From nodes to knowledge: Exploring social network analysis in education. ACM Transactions on the Web, 19(1), 1-36.

[15] Selim, H., Zhan, J. Towards shortest path identification on large networks. J Big Data 3, 10 (2016). https://doi.org/10.1186/s40537-016-0042-7

[16] Craveirinha, J., Clímaco, J., Girão-Silva, R., & Pascoal, M. (2024). Multiobjective Path Problems and Algorithms in Telecommunication Network Design—Overview and Trends. Algorithms, 17(6), 222.

[17] Yinbo Feng and Ping Wu (2017) MATEC Web of Conferences 139, 00065 ICMITE (2017) DOI: 10.1051/matecconf/201713900065

[18] Bairamkulov, R., & Friedman, E. G. (2023). Graphs in VLSI. Cham, Switzerland: Springer.

[19] Bao Zhang, Handong Zhao, Xiaochun Cao, (2012). ACM Press the 20th ACM international conference - Nara, Japan (2012.10.29-2012.11.02)] Proceedings of the 20th ACM international conference on Multimedia - MM '12 - Video object segmentation with shortest path. , (), 801–. doi:10.1145/2393347.2396316.

[20] Zhang, D., Shou, Y., & Xu, J. (2024). A mapreduce-based approach for shortest path problem in road networks. Journal of Ambient Intelligence and Humanized Computing, 15(2), 1251-1259.

[21] N. Edmonds, J. Willcock, and A. Lumsdaine. Expressing graph algorithms using generalized active messages. SIGPLAN Not., 48(8):289--290, Feb. 2013.

[22] Daniele Palossi, Andrea Marongiu, (2016). ACM Press the 19th International Workshop - Sankt Goar, Germany (2016.05.23-2016.05.25). Proceedings of the 19th International Workshop on Software and Compilers for Embedded Systems - SCOPES '16 - Exploring Single Source Shortest Path Parallelization on Shared Memory Accelerators. , (), 197–200. doi:10.1145/2906363.2915925

[23] Ryan G. Coleman (2009), Shortest Geometric Paths Analysis in Structural Biology. Publically accessible Penn Dissertations, University of Pennsylvania.

[24] Hemant Singh, Dr Mahavir Yadav, Dr. Balbir Singh, Dr. Sachin Shinde, Dr. Harshad Kurle and Shweta Sharma (2019). ANTILISHMANIAL ACTIVITY OF LEMONGRASS COMPONENTS AGAINST LEISMANIA SPECIES - AN INSILICO STUDY UJC Journal. No. 45489.

[25] Steven Wang and Tao Huang (2020). Applications of Network Analysis in Biomedicine. Precision Medicine.

[26] JIANGPENG CHEN1, XIAOQI DONG2, XUN LEI1, YINYIN XIA1, QING ZENG1, PING QUE1, XIAOYAN WEN1, SHAN HU1 and BIN PENG1 (2018). Non small-cell lung

[27] Dong, X., Vegesna, K., Brouwer, C., & Luo, W. (2022). SBGNview: towards data analysis, integration and visualization on all pathways. Bioinformatics, 38(5), 1473-1476.

[28] M. Thilaga, R. Vijayalakshmi, R. Nadarajan and D. Nandagopal (2017). Shortest path based network analysis to characterize cognitive load states of human brain using EEG based functional brain networks. Journal of Integrative Neuroscience. 17. 1-23.

[29] 10.3233/JIN-170049.

[30] Zhang Xin, Chen Yan and Li Taoying (2015). Optimization of logistics route based on Dijkstra. [IEEE 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS) - Beijing, China (2015.9.23-2015.9.25)] 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS) -, (), 313–316. doi:10.1109/ICSESS.2015.7339063)

[31] Hao Zou and Tiantian Zhang (2018) Research on Vehicle Routing Algorithm for Distribution Supply Chain Logistics. MATEC Web of Conferences 227, 02003.

[32] Gabriel Michau, Alfredo Nantes, Ashish Bhaskar, Edward Chung, Patrice Abry, and Pierre Borgnat (2017). Bluetooth Data in an Urban Context: Retrieving Vehicle Trajectories. IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.

[33] Robert Sitzenfrei, Martin Oberascher, and Jonatan Zischg (2019). Identification of Network Patterns in Optimal Water Distribution Systems Based on Complex Network Analysis. World Environmental and Water Resources Congress 2019.

[34] Olusina, J.O. & Olaleye, J.B. (2017) Journey to Crime Using Dijkstra's Algorithm. Nigerian Journal of Environmental Sciences and Technology (NIJEST)]. environmental and Social science.

[35] Krishnaiyan "KT" Thulasiraman Editor-in-Chief 92016) "Handbook of Graph Theory, Combinatorial Optimization, and Algorithms". CRC Press © Taylor & Francis Group, LLC.

[36] Koutrouli Mikaela, Karatzas Evangelos, Paez-Espino David, Pavlopoulos Georgios A. "Guide to Conquer the Biological Network Era Using Graph Theory". Frontiers in Bioengineering and Biotechnology, VOLUME=8 YEAR=2020 PAGES=34,://www.frontiersin.org/article/10.3389/fbioe.2020.00034. DOI=10.3389/fbioe.2020.00034 ISSN=2296-4185.

[37] Md. Saidur Rahman (2017). Basic Graph Theory. © Springer International Publishing AG.

[38] Richard E. Neapolitan (2015), "Foundations of algorithms—Fifth edition". Copyright © 2015 by Jones & Bartlett Learning, LLC, an Ascend Learning Company. ISBN 978-1- 284-04919-0 (pbk.)

[39] S. Janson, D. E. Knuth, T. Luczak, and B. Pittel (1993). "The birth of the giant component," Random Structures and Algorithms 4, 233–358.

[40] Stuart E. Dreyfus (1969) An Appraisal to some Shortest-Path Algorithms. Institute for Operations Research and the Management Sciences (INFORMS), Maryland USA.

[41] U. Meyer (2001). Single-source shortest-paths on arbitrary directed graphs in linear average-case

time. In SODA'01, pages 797–806, Philadelphia, SIAM. ISBN 0- 89871-490-7.

[42] Madhumita Panda, Abinash Mishra (2018). A Survey of Shortest-Path Algorithms, International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number (2018).

[43] Li, C., Hu, R., Du, X., & Ji, Y. (2025). Optimized Parallel Breadth-First Search with Adaptive Strategies. In Proceedings of the 1st FastCode Programming Challenge (pp. 28-32).

[44] E. W. Dijkstra (1959). A Note on Two Problems in Connexion with Graph. Numerische Mathematlk l, 269 - 27 I.

[45] Yue, Y. (1999). An efficient implementation of shortest path algorithm based on dijkstra algorithm. Journal of Wuhan Technical University of Surveying & Mapping.

[46] Gass, Saul; Fu, Michael (2013). Gass, Saul I; Fu, Michael C (eds.). "Dijkstra's Algorithm". Encyclopedia of Operations Research and Management Science. pringer. 1. doi:10.1007/978-1-4419-1153-7. ISBN 978-1-4419-1137-7 – via Springer Link.

[47] Edelkamp and Stefan (2012). Heuristic Search || Basic Search Algorithms. doi:10.1016/b978-0-12-372512-7.00002-x.

[48] Mikkele Thorup (1999). Undirected SSSP with positive integer weights in linear time. AT & T Labs Research.

[49] Sanjoy Dasgupta, Christos Papadimitriou, Umesh Vazirani (2008). Algorithms. Published by McGraw-Hill

[50] Michle L. Fredman and Robert T. Tarjan (1984), Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. IEEE, New York, 1984, pp. 338-346, 0 IEEE.

[51] Mikkel Thorup (2004). Integer priority queues with decrease key in constant time and the single source shortest paths problem. Journal of Computer and System Sciences 69 330–353.

[52] Maurice Pollack and Walter Wiebenson (1960). Solutions of the shortest-route problem-a review. Operations Research, 8(2):224-230.

[53] U Pape (1974). Implementation and efficiency of moore-algorithms for the shortest route problem. Mathematical Programming, 7(1):212-222.

[54] D. Goldfarb, Jianxiu Hao, and Sheng-Roan Kai (1991). Shortest path algorithms using dynamic breadth-_rst search. Networks, 21(1):29-50.

[55] Stefano Pallottino (1984). Shortest-path methods: Complexity, interrelations and new propositions. Networks, 14(2):257-267.

[56] A V Goldberg and T. Radzik (1993). A heuristic improvement of the bellman-ford algorithm. Applied Mathematics Letters, 6(3):3-6.