A Review of Shortest Path Problem in Graph Theory

Khabibullo Nosirov — Tashkent University of Information Technologies
Elnur Norov — Tashkent University of Information Technologies
Email: elnurnorov@gmail.com
Shakhzod Tashmetov — Tashkent University of Information Technologies

ABSTRACT

A graph's shortest-path algorithm identifies the route with the lowest cost connecting two vertices. The literature covers a wide range of shortest-path algorithms and is interdisciplinary. The survey of shortest-path algorithms in this paper is based on a taxonomy that is presented in the paper. The varieties of the shortest-path issue comprise one dimension of this taxonomy. Due to each solution's space and temporal challenges, no general algorithm can solve all incarnations of the shortest-path problem. The shortest-path algorithm's ability to operate on a static or dynamic graph, its ability to provide accurate or approximative results, and whether or not it aims to attain time-dependence rather than just goal-directedness are all significant aspects of the taxonomy. According to the proposed taxonomy, shortest-path algorithms are examined and categorized in this survey. The poll also outlines the issues and suggested fixes related to each taxonomy category.

Keywords: shortest path; algorithms; route; performance; review; graph; node;

Introduction

One of the well-researched areas in computer science, notably in graph theory, is the shortest-path problem. A path that meets the least length requirements between a source and a destination is the ideal shortest path. The topic has many varied applications, leading to a boom in research on shortest-path algorithms. Network routing protocols, route planning, traffic control, pathfinding in social networks, computer games, and transportation systems are just a few examples of these uses.

Selected graph types are taken into account by shortest-path algorithms. An object in mathematics made up of vertices and edges are known as a generic graph. Vertices in a spatial graph have positions that are not perceived as places in space. On the other hand, a spatial graph has vertices that are situated at the ends of the edges. A planar graph is one that is drawn in two dimensions with continuous, non-straight, non-crossing edges.

Additionally, there are numerous contexts in which a shortest path can be found. As an illustration, the graph might be static, with constant vertices and edges. A graph, however, can be dynamic, meaning that new vertices and edges may be added, changed, or removed over time. Edges in the graph can be directed or undirected. Either negative or non-negative weights may be applied to the edges. Real or integer numbers may be used as the values. Depending on the nature of the issue, this.

Two main categories can be used to classify the shortest-path methods. The first category is single source shortest path (SSSP), where the goal is to identify the shortest routes between a single-source vertex and all other vertices. The goal of the all-pairs shortest-path (APSP) category, which is the second, is to identify the shortest routes connecting every pair of vertices in a graph. Strictly speaking, there are two types of answers that can result from the shortest-path calculation. The graph's properties and the application's requirements influence the algorithm that should be used. As an illustration, the goal of approximation shortest-path algorithms is to provide quick results even when dealing with a sizable input graph.

The goal of this survey is to present a breakdown of these shortest-path algorithms through an appropriate taxonomy given the substantial body of literature on algorithms for determining the shortest path. The classification intends to assist researchers, practitioners, and application developers in comprehending the operation of each shortest-path algorithm and in selecting the appropriate type or category of shortest-path algorithms for a given scenario or application area. The proposed classification is shown in Figure 1, where each branch defines a particular type of shortest-path problem.

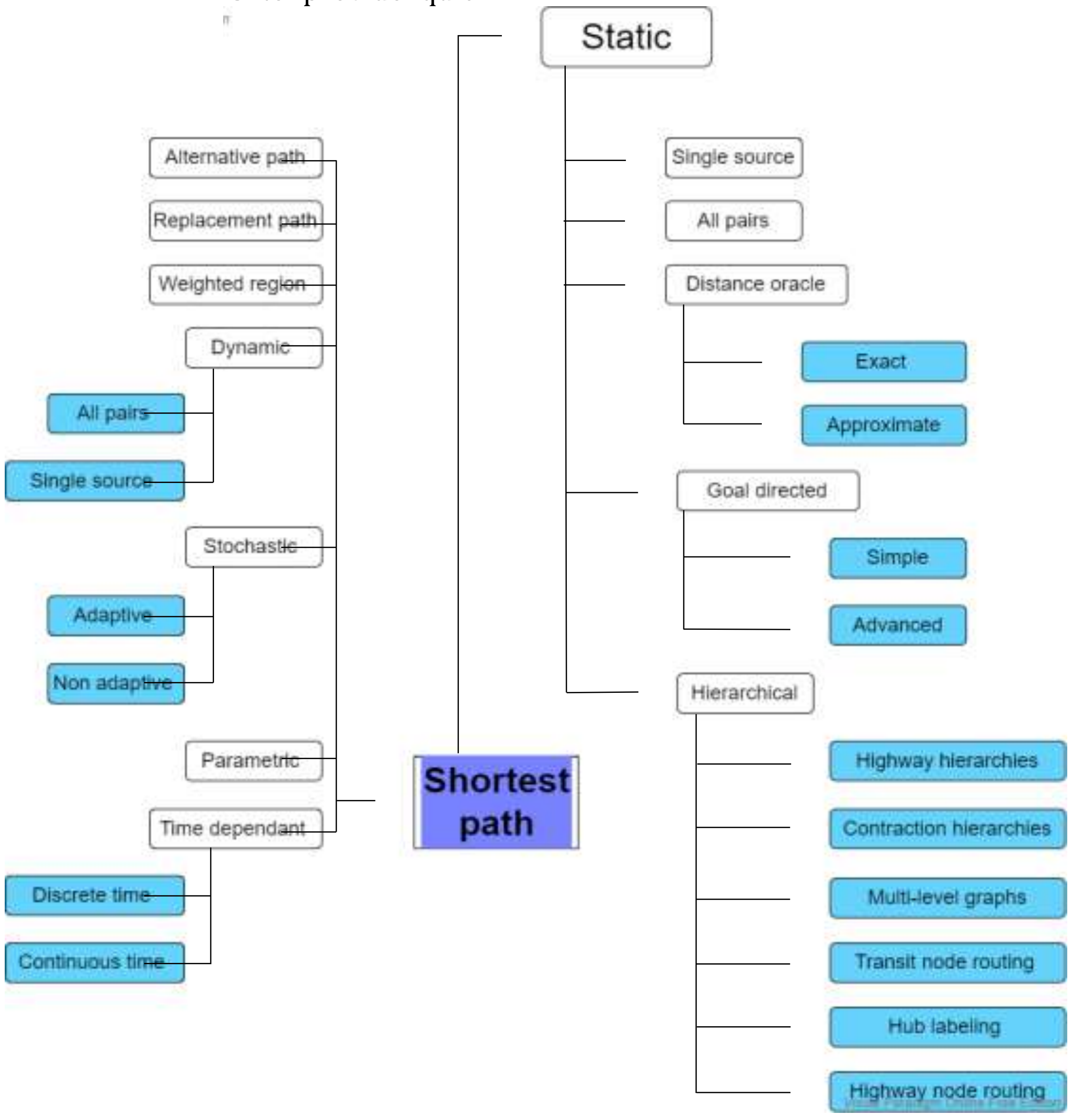

Figure 1: Shortest-Path Calculation Scientific classification

Shortest-Path Calculation Scientific classification

The suggested classification divides the many shortest-path algorithms into numerous high-level branches, just like in Figure 1.

The static branch in Figure 1 has a collection of methods that work with graphs that have fixed edge weights. The weights may indicate cost, travel time, distance, or any other grading factor. Some static algorithms precompute the graph given that the weights are fixed. The methods aim for a trade-off between query speed and pre-computation and storage needs. Two traditional shortest-path algorithms are included in static algorithms: Single-source shortest path (SSSP) and All-pairs shortest path (APSP). The shortest path between a given vertex and every other vertex is calculated by the SSSP algorithms. Between each pair of vertices in the graph, the shortest paths are determined via the APSP algorithms. A linear complexity problem is created from the shortest-path problem using hierarchical algorithms. In terms of calculation, this can result in orders of magnitude better performance. Algorithms that are goal-directed maximize the distance or time to the desired outcome. To reduce the time it takes to find the shortest path, distance oracle algorithms add a preprocessing step. Both precise and approximative distance oracle techniques are available. A set of algorithms that handle update or query operations on a graph over time is shown in Figure 1's dynamic branch. The update operation has the ability to modify edge weights as well as add or remove edges from the graph. The distance between the source and destination vertices is calculated during the query operation. Both (APSP) and (SSSP) algorithms are examples of dynamic algorithms. Target graphs that vary over time in a predictable way are the focus of time-dependent algorithms. By treating the edges as random variables, stochastic shortest-path algorithms can capture the uncertainty around the edges. The results of parametric shortest-path algorithms are based on all possible values for a given parameter. Every edge between the source vertex and the destination vertex is taken into account by replacement path algorithms to determine a solution that avoids the edge in question. Algorithms that use replacement pathways reuse the computations from each edge they avoid to obtain high performance. Alternative path algorithms, on the other hand, also determine the shortest route between vertices while avoiding a certain edge. Replacement pathways do not have to identify a particular vertex or edge to fall into either of the two categories. Alternative shortest paths, on the other hand, avoid the designated edge on the shortest path. The approximate shortest path on weighted planar divisions is discovered through the weighted-regions issue.

Related works

In his study on the precise and approximative shortest pathways, Zwick adopts a theoretical stance. Single-source shortest-path (SSSP), all pairs shortest-path (APSP), spanners (a weighted graph variation), and distance oracles are all covered by Zwick's study. The survey provides examples of the different approaches each category takes to managing graphs with directed and undirected edges as well as edge weights that are negative and non-negative. With a focus on spanners and distance oracles, Sen explores approximate shortest-paths techniques. The construction of spanners and distance oracles algorithms as well as their practical applicability in a static all-pairs shortest-paths scenario are covered in Sen's survey. In his study on query processing algorithms, Sommer examines trade-offs between query time and index size. The transportation network class of algorithms is also introduced in Sommer's study, along with algorithms for general graphs, planar graphs, and complex graphs.

Numerous studies concentrate on algorithms that are aimed at traffic applications, particularly route-planning techniques. A network refers to a graph in this connected body of work. Dijkstra's algorithm modifications are categorized by Holzer et al. based on the speedup techniques used. Their study places a focus on methods that ensure accuracy. It makes the case that the type of data has a significant impact on how effective speed-up techniques are. In addition, the layout, memory, and acceptable preprocessing time all affect the ideal speedup strategy. Fu et al. [60] survey algorithms that focus on heuristic shortest-path algorithms to quickly determine the shortest path as opposed to optimal shortest-path algorithms. The purpose of heuristic algorithms is to reduce computing time. The survey suggests the primary heuristic algorithm differentiators as well as the associated processing expenses. From a theoretical perspective, Goldberg examines how well point-to-point shortest path algorithms perform over road networks. When given a section of the graph, Goldberg analyzes algorithms like Dijkstra and A and demonstrates heuristic methods for calculating the shortest path. A graph's good worst-case and average-case boundaries are demonstrated by the survey. It also discusses reach-based pruning and exemplifies how to modify all-pairs shortest-path algorithms to compute reaches while keeping the same time constraint as their original equivalents. Route planning speedup methods are compared to certain shortest-path issues, including their dynamic and time-dependent variations, by Delling and Wagner [37]. For instance, the authors contend that time-dependent networks cannot be served by the same shortcuts utilized in static networks. In essence, they look into which networks can use the current methods. Bast [11] provides examples of speed-up methods for quick routing between transportation networks and road networks. According to Bast's survey, both networks' algorithms are unique and call for various speedup methods. The survey also shows how the speed-up method compares to Dijkstra's algorithm. The survey also raises two unanswered questions: (1) how to increase speed despite the absence of a hierarchy in transportation networks; and (2) how to compute local searches effectively, such as in neighborhoods.

In their examination of algorithms that study fully dynamic directed graphs, Demetrescu and Italiano [39] place a focus on dynamic shortest-paths and dynamic transitive closures. The definitions of the algebraic and combinatorial properties as well as the tools for dynamic approaches are the main points of the survey. The review addresses two key issues, namely whether fully dynamic single-source shortest path algorithms can be solved quickly over general graphs and if dynamic shortest pathways can achieve a space complexity of O (n^2). Techniques for dynamic graph weights and dynamic graph topology are reviewed by Nannicini and Liberti. They list both established and cutting-edge methods for locating trees and shortest paths in massive graphs with changing weights. They focus on two variations of the issue: time dependency and what they refer to as weight cost updates. The time-dependent methods in a dynamic environment are the main topic of Dean's survey [35]. It examines a single particular instance, the First-In-First-Out (FIFO) network, revealing structural characteristics that enable the creation of effective polynomial-time algorithms.

These features that set this survey apart from all of its predecessors are presented here. It begins by presenting a taxonomy that can help in determining the best algorithm to employ given a particular circumstance. Second, the algorithms are presented in a chronological order for each branch of the taxonomy, illustrating how the ideas and algorithms have changed through time. Additionally, our survey is more thorough.

Issue Definition

Given a set of vertices V , a source vertex s, a destination vertex d, where s, d ∈ V , and a set of weighted edges E, over the set V , find the shortest-path between s and d that has the minimum weight. The input to the shortest-path algorithm is a graph G that consists of a set of vertices V and edges E. The graph is defined as G = (V, E). The edges can be directed or undirected. The edges have explicit weights, where a weight is defined as w(e), where e ∈ E, or unweighted, where the implicit weight is considered to be 1. When calculating the algorithm complexity, we refer to the size of the set of vertices V as n and the size of the set of edges E as m.

Static Shortest-Path Algorithms

We go over the solutions to the single-source shortest-path (SSSP) and all-pairs shortest-path (APSP) problems in this part.

Definition: Given a Graph G = (V,E) and Source s ∈ V , compute all distances δ(s, v), where v ∈ V

The graph is unweighted in the simplest case for SSSP. According to Cormen et al. [34], the breadtfirst search can be easily used by beginning a scan from a root vertex and looking at the surrounding vertices. The path with the fewest edges between the source and destination vertices is found by probing the non-visited vertices of each neighboring vertex.

The single source shortest-path (SSSP) problem between a given vertex and all other vertices in a graph is resolved by Dijkstra's algorithm [42]. Utilizing directed graphs with non-negative weights, Dijkstra's algorithm is applied.

The method separates vertices into two categories: solved and unsolved. The source vertex is initially set as a solved vertex, and any additional edges related to the source vertex (through unsolved vertices) are then checked for shortest paths to the destination. The algorithm then adds the appropriate vertex to the list of solved vertices after determining the shortest edge. Up until all of the vertices are solved, the algorithm iterates. The time complexity of Dijkstra's algorithm is O (n^2). The algorithm has the benefit of not having to look into every edge, which is one advantage. This is especially helpful if some of the edges' weights are pricey. The approach only works with non-negative weighted edges, which is a drawback. Additionally, it only applies to static graphs.

Dijkstra's algorithm is regarded as a greedy algorithm since it uses a brute-force search to identify the best shortest path. The successive approximation method used by Dijkstra's algorithm is based on Bellman Ford's optimality premise [17]. This suggests that the reaching approach, a technique used by Dijkstra's algorithm to solve the dynamic programming equation, can be used [41].

By focusing on the smaller issues, dynamic programming has the advantage of avoiding the brute-force search method. While avoiding explicitly looking at every potential answer, dynamic programming algorithms probe an infinitely huge set of solutions. The optimal solution is found using both the greedy and the dynamic programming variants of Dijkstra's algorithm. The distinction is that both may take different routes to the best answers.

In contrast to Dijkstra's method, Bellman, Ford, and Moore [18,53] create an SSSP algorithm that can handle negative weights. It functions similarly to Dijkstra's algorithm in that it tries to compute the shortest path but selects all neighbor edges rather than just those with the smallest distance. It then continues in n 1 cycles to ensure that all modifications have been distributed throughout the graph. Dijkstra's algorithm can solve problems more quickly than Bellman-algorithm, Ford's however it can't recognize negative cycles or work with negative weights.

All-Pairs Shortest-Path (APSP)

Definition: Given a graph G = (V,E), compute all distances between a source vertex s and a destination v, where s and v are elements of the set V .

A graph with non-negative edge weights is the most typical use of APSP. For each vertex in the graph in this situation, Dijkstra's algorithm can be computed separately. $O(mn+n^2 logn)$ will be the time complexity.

Many algorithms handling real edge weights have been put forth for the all-pairs shortest-path problem. Finding all pairs of shortest paths (APSP) in a weighted network with positive and negative weighted edges is the goal of the Floyd-Warshall algorithm [52]. Although their system can recognize negative-weight cycles, it cannot break them. Floyd-Warshall algorithm has an $O(n^3)$ complexity, where n is the number of vertices. The diagonal path matrix is probed to find negative-weight cycles. Because the Floyd-Warshall technique does not save the intermediate vertices as it calculates, it is unable to identify the precise shortest pathways between vertex pairs. However, one can save this data within the algorithmic stages via a straightforward update. The algorithm's space complexity is $O(n^3)$.

However, by employing a single displacement array, this space complexity can increase to $O(n^2)$. The algorithm's ability to manage negative-weight edges and find negative-weight cycles is one of its strongest points. The biggest disadvantage is that Dijkstra's algorithm will have an $O(mn + n^2 \log n)$ time complexity when applied to all vertices to transform it from SSSP to APSP. If and only if $m < n^2$, this timing complexity is less difficult than $O(n^3)$ (i.e., having a sparse graph).

$O(n^2 \log n)$ [63] is the best non-negative edge weight complexity. The algorithm starts by weighting all adjacency lists in ascending order. After that, it iteratively conducts an SSSP calculation n times. It picks and labels the edge with the lowest potential in the first phase using the idea of potential over the vertices' edges. Potential is a probability distribution on fully directed graphs with arbitrary edge lengths and no negative cycles. It is derived from the potential model. The algorithm comprises two primary stages, each having a different invariant and executing at a speed of $O(n^2 \log n)$.

Goal-Directed Shortest-Paths

An annotation is a piece of extra information that is added to a vertex or edge of a graph as part of a goal-directed shortest-path search technique. Using this data, the algorithm can choose which area of the graph to remove from the search space.

A straightforward goal-directed algorithm called A is proposed by Hart et al. The technique suggests using a heuristic method to identify the shortest path. In contrast to Dijkstra's algorithm, A is an informed algorithm that looks for paths that will take it to its intended destination. The best best-first-search greedy algorithm is A. But A differs from other algorithms in that it keeps in mind the distance that it went. If a valid heuristic is employed, A always finds the shortest route. The algorithm's main advantage is that it searches fewer vertices than Dijkstra, which should make it faster. A will not find the shortest path if a good heuristic approach is not used, which is a drawback.

A related answer to the issue is provided by Gutman, whose work is founded on the idea of reach. The Euclidean coordinates of each vertex and a reach value are both stored in Gutman's method. In contrast to the work by Goldberg and Werneck, Gutman's approach beats their suggested strategy given one landmark while it performs worse given sixteen landmarks. This is due to the fact that it can be integrated with the A algorithm. The disadvantages of Gutman's method include its reliance on domain-specific presumptions, increased preprocessing complexity, and inapplicability in dynamic environments.

A strategy called edge labels depends on computing the data for an edge (e) and its vertices (M) in advance. All of the shortest-path vertices that begin with the edge e are represented by the superset M(e). The graph is first divided into a set of identically sized sections and a predetermined set of boundary vertices.

An SSSP computation is performed on the regions for all the border vertices in order to calculate the edge flags. Additional edge-label modifications are shown in a number of works, including those by Kohler et al. [62], Schulz et al, and Lauther.

The arc-flag technique is an algorithm proposed by Mohring et al. for sparse directed graphs with non-negative edge weights. By splitting the graph into areas and identifying whether an arc in a particular zone is on the shortest-path, the arc-flag approach preprocesses graph data to produce information that speeds shortest-path queries. The arc-flag approach outperforms the typical Dijkstra's algorithm over a large graph by a factor of 500, given an appropriate partitioning scheme and a bi-directed search. By conducting a single search for each location, Schilling et al. offer an improvement. On a subnetwork with one million vertices, their method results in a speedup of more than 1,470.

Hierarchical Shortest-Path

In the pre-processing stage, multi-layered vertex hierarchies are created using hierarchical shortest-path methods. In regions like road networks, where it displays hierarchical characteristics such arranging significant streets, freeways, and urban streets, a hierarchical structure is prevalent.

In general, techniques utilizing contraction hierarchies offer little spatial complexity. Reach-based techniques, highway hierarchies, and vertex routing are just a few of the many variations of contraction hierarchies.

Conversely, Transit-vertex Routing and Hub Labels offer quick query times.

The parts that follow talk about different hierarchical algorithms.

Multi-Level Graphs

The shortest paths in a multi-level overlay graph do not employ vertex from the higher levels if a set of vertices are located at a particular level. This strategy also depends on choosing the right vertices to serve as markers on higher levels. A multi-level graph-based decomposition method with space reduction as its goal is proposed by Schulz et al. This approach precomputed the shortest-paths and substitutes a weight equal to the length of the shortest-path for the weights of single edges. As a result, the subgraph is scaled down in comparison to the main graph. The distances in a subgraph between a group of vertices are equal to the distances in the original graph's shortest-path graph between the same group of vertices.

Algorithms for Dynamic Shortest Paths

The efficient live processing of updates and query operations is the primary need for dynamic shortest-path algorithms. The update procedure involves adding or removing edges from the graph. The distance between vertices is computed during the query operation.

The only algorithms that can handle insertions and deletions are those that are fully dynamic. Insert operations can be handled by incremental algorithms, while delete operations cannot. Insert operations cannot be handled by incremental algorithms, whereas delete operations can. This suggests that both incremental and decremental algorithms have some degree of dynamic behavior.

Time-Dependent Shortest-Path Algorithms

Processed by a time-dependent shortest-path method are graphs with edges connected by an edge-delay function. How long it takes for a signal to go from one vertex to another is shown by the edge-delay function. The query operation searches the graph for the path with the shortest distance between the source and the destination vertex. The result that was returned is the best departure time that was located within the specified time frame.

Algorithms for the Stochastic Shortest Path

A stochastic shortest-path makes an effort to represent the uncertainty surrounding the edges as random variables. The goal then shifts to finding the shortest paths based on the lowest anticipated expenses. Adaptive and non-adaptive algorithms are the two prominent study areas in this issue. Based on the graph's current state at a certain time instance, the adaptive algorithms estimate what the best next hop would be. The non-adaptive algorithms prioritize reducing the path's length.

Adaptive Algorithms

To find the apriori least-expected-time pathways from all source vertices to a single destination vertex, Miller-Hooks and Mahmassani offer an approach. When the graph is busy, a computation is made for each departure time. Additionally, they suggest a lower-bound for these a priori least-expected-time pathways.

According to Nikolova et alalgorithm, the probability can be maximized without going over a predetermined limit for the length of the shortest pathways. Edge weights for the probabilistic model they define are taken from a predetermined probability distribution. The path that has the highest likelihood of indicating one that fails to cross a particular threshold is the best option.

Non-Adaptive Algorithms

The idea put out by Loui is to combine the length of the path with a utility function that is monotone and non-decreasing. The utility function can be divided into its edge lengths when it behaves linearly or exponentially. Thus, pathways that maximize the utility function can be used to identify the utility function using conventional shortest-paths procedures.

An algorithm for the best route planning under uncertainty is put forth by Nikolova et al. They determine the target as a function of the source's departure time and the path's length. Due to the penalizing behavior they display for early and late arrivals, they suggest that the path and start time can be simultaneously

Conclusion

Which method is best, have to think about which one is appropriate for the kind of graph you're working with and the shortest path problem you're trying to solve. In many applications, including communications, graph theory, and electronics difficulties, the shortest path algorithm is a significant problem. The shortest path Algorithm is a major issue in many Applications like communications, in graph theory and electronics problems. Various algorithm for resolving SPA i.e. Dijkstra's, Bellman Ford, Johnson's, has been discussed with there drawbacks and applications

In this article, we suggest a classification system for the shortest-path issue in terms of science. We highlight the cutting-edge research for each branch of the classification and illustrate the distinguishing aspects. The classification gives shortest-path problem researchers a roadmap for where a necessary problem definition fits within the existing relevant work

References

[1] I. Abraham and D. Delling. A hub-based labeling algorithm for shortest paths in road networks. Experimental Algorithms, 2011.

[2] I. Abraham, D. Delling, A. Goldberg, and R. Werneck. Hierarchical hub labelings for shortest paths. AlgorithmsESA 2012, 2012.

[3] I. Abraham, A. Fiat, A. V. Goldberg, and R. F. Werneck. Highway dimension, shortest paths, and provably efficient algorithms. Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, pages 782–793, 2010.

[4] R. Agarwal, P. B. Godfrey, and S. Har-Peled. Approximate distance queries and compact routing in sparse graphs. IEEE INFOCOM, pages 1754–1762, 2011.

[5] A. V. Aho and J. E. Hopcroft. The Design and Analysis of Computer Algorithms. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1974.

[6] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. STOC, 2000.

[7] D. Aingworth, C. Chekuri, and R. Motwani. Fast Estimation of Diameter and Shortest Paths (without matrix multiplication). SODA, pages 547–553, 1996.

[8] J. Arz, D. Luxen, and P. Sanders. Transit Node Routing Reconsidered. SEA, 2013.

[9] M. Babenko, A. Goldberg, A. Gupta, and V. Nagarajan. Algorithms for hub label optimization. Automata, Languages, and Programming, 2013.

[10] M. J. Bannister and D. Eppstein. Randomized Speedup of the Bellman-Ford Algorithm. ANALCO, 2011.

[11] H. Bast. Car or public transport two worlds. Efficient Algorithms, pages 355–367, 2009.

[12] H. Bast, S. Funke, D. Matijevic, P. Sanders, and D. Schultes. In Transit to Constant Time Shortest-Path Queries in Road Networks. ALENEX, 2007.

[13] S. Baswana and S. Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. Random Structures and Algorithms, pages 532–563, 2007.

[14] G. Batz, R. Geisberger, S. Neubauer, and P. Sanders. Time-dependent contraction hierarchies and approximation. Experimental Algorithms, pages 166–177, 2010.

[15] R. Bauer and D. Delling. SHARC: Fast and robust unidirectional routing. Journal of Experimental Algorithmics (JEA), 2009.

[16] R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner. Combining hierarchical and goal-directed speed-up techniques for dijkstra's algorithm. Journal of Experimental Algorithmics, pages 303–318, 2010.

[17] R. Bellman. Dynamic Programming. Princeton University Press, 1957.

[18] R. Bellman. On a routing problem. Quarterly of Applied Mathematics, 1958.

[19] A. Bernstein. Fully Dynamic (2 + epsilon) Approximate All-Pairs Shortest Paths with Fast Query and Close to Linear Update Time. 2009 50th Annual IEEE Symposium on Foundations of Computer Science, pages 693–702, 2009.

[20] A. Bernstein. A Nearly Optimal Algorithm for Approximating Replacement Paths and k Shortest Simple Paths in General Graphs. Proceedings of the Twenty-First Annual ACM-Siam Symposium on Discrete Algorithms, pages 742–755, 2010.

[21] A. Bernstein. Maintaining shortest paths under deletions in weighted directed graphs. STOC, page 725, 2013.

[22] A. Bernstein and L. Roditty. Improved dynamic algorithms for maintaining approximate shortest paths under deletions. ACM-SIAM Symposium on Discrete Algorithms, pages 1355–1365, 2011.

[23] P. v. E. Boas. Preserving order in a forest in less than logarithmic time. pages 75–84, 1975.

[24] P. v. E. Boas, R. Kaas, and E. Zijlstra. Design and implementation of an efficient priority queue. Mathematical Systems Theory, pages 99–127, 1976.

[25] S. Cabello. Many distances in planar graphs. Algorithmica, pages 361–381, 2012.

[26] K. Cechlrov and P. Szab. On the monge property of matrices. Discrete Mathematics, 81(2):123– 128, 1990.

[27] T. M. Chan. All-pairs shortest paths for unweighted undirected graphs in o(mn) time. Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, pages 514–523, 2006.

[28] T. M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. Proceedings of the thirty-ninth annual ACM symposium on Theory of computing, pages 590–598, 2007.

[29] L. Chang, J. X. Yu, L. Qin, H. Cheng, and M. Qiao. The exact distance to destination in undirected world. The VLDB Journal, 21(6):869–888, 2012.

[30] W. Chen, C. Sommer, S.-H. Teng, and Y. Wang. A compact routing scheme and approximate distance oracle for power-law graphs. ACM Transactions on Algorithms, pages 1–26, 2012.

[31] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick. Reachability and Distance Queries via 2-Hop Labels. SIAM Journal on Computing, 32:1338–1355, 2003.

[32] E. Cohen and U. Zwick. All-Pairs Small-Stretch Paths. Journal of Algorithms, pages 335–353, 2001.

[33] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. Journal of Symbolic Computation, pages 251 – 280, 1990.

[34] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. Introduction to Algorithms. McGraw-Hill Higher Education, 2nd edition, 2001.

[35] B. Dean. Shortest paths in FIFO time-dependent networks: Theory and algorithms. Rapport technique, 2004.
[36] D. Delling, T. Pajor, and R. Werneck. Round-based public transit routing. ALENEX'12, 2012.
[37] D. Delling and D. Wagner. Time-dependent route planning. Robust and Online Large-Scale Optimization, 2:1–18, 2009.
[38] C. Demetrescu and G. Italiano. A new approach to dynamic all pairs shortest paths. Journal of the ACM (JACM), pages 1–29, 2004.
[39] C. Demetrescu and G. F. Italiano. Dynamic shortest paths and transitive closure: Algorithmic techniques and data structures. Journal of Discrete Algorithms, pages 353–383, 2006.
[40] U. Demiryurek, F. Banaei-kashani, and C. Shahabi. Online Computation of Fastest Path in Time-Dependent. SSTD, pages 92–111, 2011.
[41] E. V. Denardo. Dynamic Programming: Models and Applications. Dover Publications, 2003.
[42] E. W. Dijkstra. A note on two problems in connexion with graphs. Numerische Mathematik, pages 269–271, 1959.
[43] B. Ding, J. X. Yu, and L. Qin. Finding time-dependent shortest paths over large graphs. Proceedings of the 11th international conference on Extending database technology Advances in database technology EDBT 08, page 205, 2008.
[44] H. Djidjev. Efficient algorithms for shortest path queries in planar digraphs. Graph-Theoretic Concepts in Computer Science, pages 151–165, 1996.
[45] W. Dobosiewicz. A more efficient algorithm for min-plus multiplication. Internat. J. Comput. Math, 1990.
[46] D. Dor, S. Halperin, and U. Zwick. All-pairs almost shortest paths. SIAM Journal on Computing, 2000.
[47] J. Driscoll and H. Gabow. Relaxed heaps: An alternative to Fibonacci heaps with applications to parallel computation. Communications of the ACM, pages 1343–1354, 1988.
[48] M. Elkin and D. Peleg. (1+epsilon,beta)-spanner constructions for general graphs. SIAM Journal on Computing, pages 608–631, 2004.
[49] Y. Emek, D. Peleg, and L. Roditty. A near-linear-time algorithm for computing replacement paths in planar directed graphs. ACM Transactions on Algorithms, 6:1–13, 2010.
[50] J. Erickson. Maximum flows and parametric shortest paths in planar graphs. SIAM, 2010.
[51] J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. Journal of Computer and System Sciences, pages 868–889, 2006.
[52] R. Floyd. Algorithm 97: Shortest Path. Communications of the ACM, pages 344–348, 1962.
[53] L. R. Ford. Network flow theory. Report P-923, The Rand Corporation, 1956.
[54] L. Foschini, J. Hershberger, and S. Suri. On the complexity of time-dependent shortest paths. Algorithmica, 2014.
[55] M. Fredman. New bounds on the complexity of the shortest path problem. SIAM, pages 83–89, 1976.
[56] M. Fredman and R. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM (JACM), pages 338–346, 1987.
[57] M. Fredman and D. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science, pages 719–725, 1990.
[58] M. Fredman and D. Willard. Surpassing the information theoretic bound with fusion trees. Journal of computer and system sciences, pages 424–436, 1993.
[59] M. L. Fredman and D. E. Willard. BLASTING through the information theoretic barrier with FUSION TREES. Proceedings of the twenty-second annual ACM symposium on Theory of computing - STOC '90, pages 1–7, 1990.
[60] L. Fu, D. Sun, and L. Rilett. Heuristic shortest path algorithms for transportation

applications: State of the art. Computers & Operations Research, pages 3324–3343, 2006.

[61] C. Gavoille, D. Peleg, S. Pŕerennes, and R. Raz. Distance labeling in graphs. J. Algorithms, pages 85–112, 2004.

[62] E. KoËhler, R. MoËhring, and H. Schilling. Acceleration of shortest path and constrained shortest path computation. Experimental and Efficient Algorithms, pages 1–17, 2005.

[63] A. Moffat and T. Takaoka. An all pairs shortest path algorithm with expected running time o(n^2 log n). SIAM J Computing, page 10231031, 1987.