THE ASIAN BULLETIN OF BIG DATA MANAGMENT
Vol. 4. Issue 3 (2024)
https://doi.org/ 10.62019/abbdm.v4i3.197

An Efficient Approach for Shortest Path Planning in Automotive Navigation System

Khalid Charan, Lachhman Dhomeja, Shahzad Memon

Chronicle

Article history

Received: July 8, 2024
Received in the **revised format:** July 15, 2024
Accepted: July 21, 2024
Available online: July 25, 2024

Khalid Charan, Lachhman Dhomeja & Shahzad Memon are currently affiliated with A.H.S Bukhari Institute of ICT, Faculty of Engineering & Technology, University of Sindh, Jamshoro, Pakistan.
Email: khalid.charan@scholars.usindh.edu.pk
Email: lachhman@usindh.edu.pk
Email: shahzad.memon@usindh.edu.pk

Corresponding Author*

Abstract

Emerging automotive engineering solutions extensively rely on navigation tools. Shortest path evaluation is core competency of optimal path planning, one of the most critical components of any navigation application. Fibonacci heap variant of state-of-the-art Dijkstra's algorithm solve single source shortest path problems in asymptotic O(E + V log V) worst-case time. In spite Fibonacci heap is theoretically dominant in time efficiency, it might not work much better empirically with real-world data, because although large and intricate, the real-world road network graphs are sparse and the graph topology greatly affects the performance of the algorithms. SGO algorithm, a more efficient approch to address single source shortest path problem, is applied to solve optimal path planning problem in automotive navigation applications. Each road segment of the road graph leads to a particular location thus will hold unique distance values all the time. The SGO calculates the total distance from start location to the end of the each road segment and evaluates shortest path based on these calculations. The SGO algorithm and the Fibonacci and Binary heap variants of Dijkstra's algorithm were tested on real-world intercity and intra-city road network graphs to evaluate running time efficiency of the algorithms in automotive navigation systems. The experimental results show that the SGO algorithm outperformed both the variants. The empirical superiority of the SGO algorithm suggests its usefulness in automotive navigation systems.

Keywords: Graph Theory; Shortest Path Algorithms; Optimal Path Planning; Vehicle Navigation System; Time Efficiency.

INTRODUCTION

Extensive reliance on navigation technologies has emerged from widespread use of Global Positioning System (GPS), Geographic Information System (GIS), and mobile communication networks (Song et al. 2017; Liu et al. 2019; Zou et al. 2019). Use of navigational tools in automobiles help in driving fatigue reduction, avoidance of traffic congestion and road accidents, environmental pollution control, time & energy savings and so on (Wang et al. 2021; Ngandu et al, 2020; Zhao et al., 2021). Innovative automotive engineering solutions of Autonomous Vehicles (AVs) have further accentuated the dependency on navigation technologies. The optimal path planning is the most critical component of an automotive navigation system (Zheng & Green, 2010; Sariff & Buniyamin, 2006); likewise, shortest path evaluation is core competency of optimal path planning (Jain et al. 2022; Yang et al. 2022; Wei et al. 2020). The real-world road networks, intrinsically detached from space and time dimensions, can be expressed as graphs by representing locations and road segments as vertices (V) and edges (E), respectively (Sedgewick, 2013), subsequently, attributes such as distance, time, cost, penalties and etc. can be assigned to the edges as shown in Figure 1. Typically, multiple paths exist to reach from

one location to another in the networks, where each of the paths cover different path length. The shortest path problem (SPP), an abstraction among many often-occurring combinatorial problems in algorithmic graph theory, aims to identify the shortest path between two points of an input graph (Cormen et al., 2009).

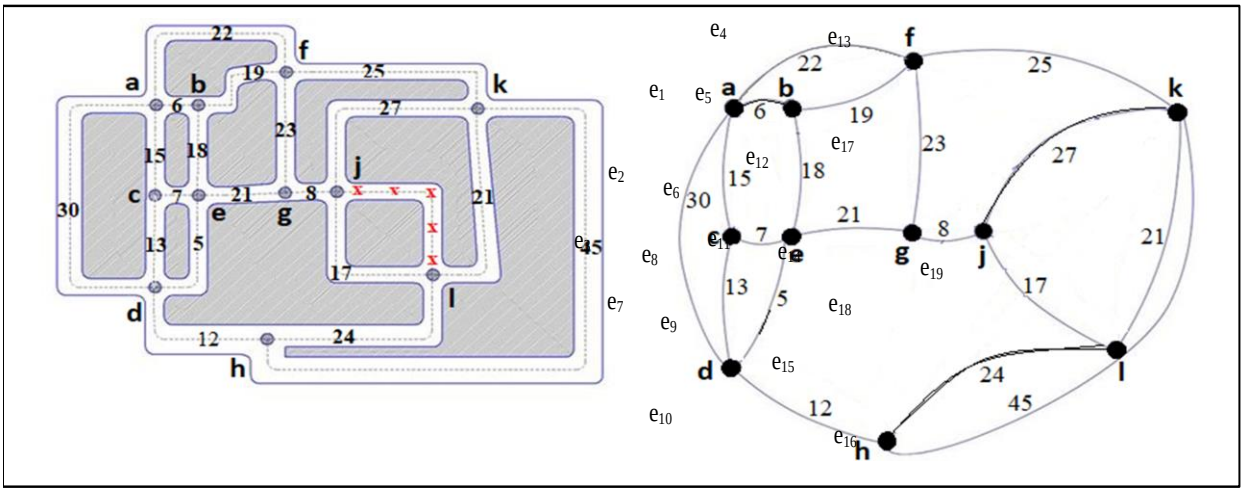


Figure. 1.
Transmutation of an imaginary road network as a graph (Pósfai & Barabási, 2016).

Ascertaining shortest path among all the available paths to reach from one location to another in a road network can be transmuted to SPP (Korte et al., 2011) and shortest path can be evaluated based on the shortest distance, the lowest cost, the least time, and the maximum / minimum penalties etc. Therefore, running time efficient and accurate shortest path algorithms with less programming complexity are essentially required in automotive navigation systems (Yasrab & Pound, 2021; Abd Algfoor et al., 2017). Single Source Shortest Path (SSSP) Dijkstra's algorithm (Dijkstra, 2022) and its variants are widely used in navigational applications including Google Maps (Koritsoglou et al., 2022; Haria et al., 2019; Alshammrei et al., 2022; Zhang et al., 2012).

The Dijkstra's algorithm (DA) guarantees 100% accurate shortest path each time (Zhou & Gao, 2019). The Fibonacci heap (DA Fibonacci) variant of DA achieved theoretical O(E + V | log V) asymptotic worst-case running time complexity that sets criteria yet to surpass (Danilovic, 2021). However, DA Fibonacci has inherent constant factors and programming complexity overheads (Cormen et al., 2009). In our previous work (Charan et al., unpublished), we delineated the empirical performance of Fibonacci and Binary heap variants of DA. We found that, although theoretically inferior in asymptotic upper bound time complexity, the Binary variant outperformed Fibonacci variant, empirically, in SSSP computations for real-world datasets, especially on sparse input graphs, due to its low execution overhead cost and coding simplicity.

Based on the acquired insight; therefore, we proposed sparse graph optimal SGO (Charan et al., unpublished), an efficient single source shortest path algorithm, for sparse graphs. Large and intricate real-world road networks are often sparse in nature, and each road segment of the graph leads to a particular location thus will hold unique distance value all the time. Therefore, we utilized the SGO algorithm to calculate the shortest paths in automotive navigation systems. The objectives of this article are to:

demonstrate the density of real-world large scale intercity and intra-city road network graphs, study applicability of the SGO algorithm for real-world road network graphs, and assess performance of the proposed solution implementation for real-world road network graphs. The running time efficiency of SGO algorithm for shortest path computation is assessed in comparison to the variants of state-of-the-art DA. The structure of the remainder of this article is as follows. Section 2 determine the density of some real-world road network graphs. The effectiveness of the SGO algorithm is studied in the section 3. The section 4 evaluates and analyzes performance of the SGO algorithm in competition to DA Fibonacci and DA Binary. Finally, the section 5 concludes the article.

Density of Real-World Road Network Graphs

A simple graph G (V, E) with |V| vertices and |E| edges where vertices are connected by edges, E = {(i, j): i, j ∈ V}, thus largest possible size of |E| would be |V| (|V| − 1) in a complete graph. The density of a graph is said to be the ratio of the edges present over all possible edges. Thus, graph density for undirected graphs, as each of the edges is counted twice ((i, j) & (j, i)), is given by:

$$d=2|E|/(|V|(|V|-1)) \quad (1)$$

Density of a directed graph, as each of the edge is counted once only ((i, j) or (j, i)) according to the direction, is given by:

$$d=|E|/(|V|(|V|-1)) \quad (2)$$

There is no specific cutoff between sparse and dense graphs, the graph densities vary between a fully dense graph of |V| vertices which has O(|V| (|V|-1)) edges where each vertex is adjacent to every other vertex in the graph (a complete graph) in contrast to an entirely sparse connected graph which has O(|V|) edges (Beamer, 2016). Graph topology is crucial in the performance of graph algorithms.

Real-world large scale intercity and intra-city road network graphs consist millions of vertices and edges (de Bruin et al., 2021) and these networks display various structural characteristic (Pósfai & Barabási, 2016) including density, which has a significant influence on SSSP algorithms. Hence, different sets of publicly available real-world road network graphs were studied to determine their densities.

Intercity Road Network Graphs

Intercity real-world road network datasets are available online. Network Repository (NR), an interactive data repository that houses hundreds of benchmark datasets for more than 30 different domains ranging from social network data to biological data including real-world road networks (Rossi & Ahmed, 2016) and 9th challenge for shortest path problem of Discrete Mathematics and Theoretical Computer Science (DIMACS) foundation (Demetrescu et al., 2006). The datasets were investigated thoroughly and details are presented in Table 1 from the road network graph density standpoint. Table 1 depicts that density of these very large-scale intercity road network graphs with millions of vertices and edges is a small fraction of 1 (unity), indicating that these graphs belong to sparse graphs category. Number of the edges w.r.t nodes is close to multiple of a small coefficient in each graph.

Table 1.

Intercity Real-World Road Graphs Density.

Intercity Graphs	*Vertices*	*Edges*	*E = c*N*	*Density*	
NR Dataset					
Asia	*12000000*	*13000000*	*1.083333*N*	*0.0000002*	
Belgium	*1000000*	*2000000*	*2*N*	*0.0000040*	
Germany	*12000000*	*12000000*	*1*N*		*0.0000002*
Great Britain	*8000000*	*8000000*	*1*N*		*0.0000003*
Italy	*7000000*	*7000000*	*1*N*		*0.0000003*
Luxembourg	*115000*	*120000*	*1.043478*N*		*0.0000181*
Netherlands	*2000000*	*2000000*	*1*N*		*0.0000010*
Full USA	*23947347*	*58333344*	*2.4359*N*		*0.0000002*
DIMACS Dataset					
Central USA	*14081816*	*34292496*	*2.435233*N*		*0.0000003*
Western USA	*6262104*	*15248146*	*2.434988*N*		*0.0000008*
Eastern USA	*3598623*	*8778114*	*2.439298*N*		*0.0000014*
Great Lakes	*2758119*	*6885658*	*2.496505*N*		*0.0000018*
California	*1890815*	*4657742*	*2.463352*N*		*0.0000026*
Northeast USA	*1524453*	*3897636*	*2.556744*N*		*0.0000034*
Northwest USA	*1207945*	*2840208*	*2.351273*N*		*0.0000039*
Florida	*1070376*	*2712798*	*2.534435*N*		*0.0000047*
Colorado	*435666*	*1057066*	*2.426322*N*		*0.0000111*
San Francisco	*321270*	*800172*	*2.490653*N*		*0.0000155*
New York	*264346*	*733846*	*2.776081*N*		*0.0000210*

Intra-city Road Network Graphs

Intra-city real-world road network graphs of well-known world metropolitan cities are available at the figshare dataset (figshare dataset, 2016). Table 2 interprets graphs density analysis of these intra-city road networks. The facts presented in the Table 2 advocate that edges are not far more than the vertices in intra-city road networks. Therefore, the networks' densities are very low, resulting in to extremely sparse graphs.

Table 2.
Intra-city Real-World Road Graphs Density.

Intra-city Graphs	*Vertices (N)*	*Edges(E)*	*E = c*N*	*Density*
Bangkak	154352	187799	1.302409*N	0.0000158
Barcelona	211655	296048	1.629884*N	0.0000132
Boston	356262	463999	1.437656*N	0.0000073
Buenos Aires	266314	434061	1.344572*N	0.0000122
Chicago	390510	561419	1.446935*N	0.0000074
Dallas	408484	549236	1.303219*N	0.0000066
Istambul	262658	380049	1.365092*N	0.000011
London	288016	375348	1.425936*N	0.000009
Los Angeles	405049	552929	1.349597*N	0.0000067
Madrid	358007	510495	1.37805*N	0.000008
Manila	243080	328060	1.358254*N	0.0000111
Milan	189951	261762	1.419897*N	0.0000145
Moscow	692891	941122	1.365067*N	0.0000039
New York	439739	624384	1.377079*N	0.0000065
Paris	474148	647244	1.320459*N	0.0000058
Philadelphia	284110	391242	1.349624*N	0.0000097
Rome	162991	215223	1.332991*N	0.0000162
St. Petersburg	372803	503144	1.448388*N	0.0000072
San Francisco	439210	585463	1.284362*N	0.0000061
Sao Paolo	214437	310588	1.302409*N	0.0000135
Washington	513708	659787	1.629884*N	0.000005

SGO ALGORITHM

Preliminaries

Suppose a graph G(V,E) consists |V| of vertices and |E| of edges. Function $f(u, v)$ provides weight of an edge joining any two vertices u, v ∈ V, and weights of edges of the graphs are non-negative. The algorithm employs labeling method where the predecessor u.π and the status u.s ∈ {labeled, unreached} are maintained for every vertex u. The real-world road graphs can be transmuted into such graphs as shown in Figure 1. The status u.s = unreached and predecessor u.π = Nil for every vertex u ∈ V, except for start vertex s where s.s = labeled, s.π = s and all edge incidents esu of s are inserted into priority queue. The SGO algorithm, iteratively, selects next vertex and sets status u.s and u.π, accordingly until meets the end-criterion.

The algorithm's method

The SGO algorithm implies priority queue, Q, of edges (the road segments) of the real-world road networks in accordance to their distance from start location to end of the road segment. The distances from start vertex to the heading vertex of the edge is termed as Arrowhead Distance (AD). Vertex s, the start vertex, is labeled, and all its edge incidents esu are considered to compute AD=f(s,u) and inserted into th e Q. In its each iterations, SGO selects minimum AD value edge from Q, if and only if, the heading vertex of the edge not marked as labeled. The SGO algorithm applies labeling actions to heading vertex of the selected edge, marks it labeled, and sets its predecessor value with the tailing vertex. Resultantly, all incident edge e(u, v) from heading vertex are computed for AD value using the formula ADe(u,v)=ADe(s,u)+f(u,v) and e(u, v) is inserted into the Q, if and only if, vertex v of e(u, v) is not labeled. SGO algorithm continued its iteration until ends.

Pseudo-code

PROPOSED SGO ALGORITHM

```
SGO (Graph G, Source s)
1     Q ← empty (priority queue)
2     for each vertex u ∈ V
3     |     u.π = Nil
4     |     u.s = {unreached}
5     s.π = s
6     s.s = {labeled} (indicates status of a vertex either labeled or unreached)
7     for each vertex u ∈ G.Adj[s]
8     |     ADe_su = f(s, u) (AD value for incident e_su)
9     |     Insert e_su into priority queue Q
10    while end criterion meets
11    |     e_uv = incident e_uv with unreached v from Q
12    |     u.π = v
13    |     u.s = {labeled}
14    |     for each unreached vertex w ∈ G.Adj[v]
15    |     |     ADe_vw = ADe_uv + f(v, w)
16    |     |     Insert e_vw into priority queue Q
```

Dry run of the algorithm

Table 3 exhibits the iteration-based execution of SGO algorithm for an imaginary intercity

/ intra-city road network graph shown in Figure 1, where circles (a, b, …, l) represent cities / location and edges (e1, e2, …, e19) denote road segments and the numbers on the road segments are distance, time, cost, penalties or etc. The SGO algorithm initializes all values and inserts incidents of the start vertex a into Q, the algorithm then starts its iteration by selecting an edge eab from Q with smallest AD label value where vertex b is not yet marked labeled, and conceptually traverse the incident edges of the vertex b. The algorithm follows the steps and applies labeling actions accordingly until end criterion meets. The priority queue is implemented using Binary heap data structure in the SGO algorithm.

Table 3.
Iteration based execution of SGO algorithm.

Iteration	Edges in Q according to AD values	Selected edge	Vertex *labeled*	*v.π*
Start vertex *a*				
1	e_1 (6), e_2 (15), e_4 (22), e_3 (30)	e_1 (6)	b	a
2	e_2 (15), e_4 (22), e_6 (24), e_5 (25), e_3 (30)	e_2 (15)	c	a
3	e_4 (22), e_8 (22), e_6 (24), e_5 (25), e_7 (28), e_3 (30)	e_4 (22)	f	a
4	e_8 (22), e_6 (24), e_5 (25), e_7 (28), e_3 (30), e_{12} (45), e_{13} (47)	e_8 (22)	e	c
discard	e_6 (24), e_5 (25), e_9 (27), e_7 (28), e_3 (30), e_{11} (43), e_{12} (45), e_{13} (47)	Vertex e (Head of e_6) already labeled		
discard	e_5 (25) , e_9 (27), e_7 (28), e_3 (30) , e_{11} (43), e_{12} (45), e_{13} (47)	Vertex f (Head of e_5) already labeled		
5	e_9 (27), e_7 (28), e_3 (30) , e_{11} (43), e_{12} (45), e_{13} (47)	e_9 (27)	d	e
discard	e_7 (28), e_3 (30) , e_{10} (39), e_{11} (43), e_{12} (45), e_{13} (47)	Vertex d (Head of e_7) already labeled		
discard	e_3 (30), e_{10} (39), e_{11} (43), e_{12} (45), e_{13} (47)	Vertex d (Head of e_3) already labeled		
6	e_{10} (39), e_{11} (43), e_{12} (45), e_{13} (47)	E_{10} (39)	h	d
7	e_{11} (43), e_{12} (45), e_{13} (47), e_{15} (63), e_{16} (84)	e_{11} (43)	g	e
discard	e_{12} (45), e_{13} (47), e_{14} (51), e_{15} (61), e_{16} (84)	Vertex g (Head of e_{12}) already labeled		
8	e_{13} (47) , e_{14} (51), e_{15} (61), e_{16} (84)	e_{13} (47)	k	f
9	e_{14} (51), e_{15} (61) , e_{19} (68), e_{17} (74), e_{16} (84)	e_{14} (51)	j	g
10	e_{15} (61) , e_{19} (68), e_{18} (68), e_{17} (74), e_{16} (84)	e_{15} (61)	l	j
discard	e_{19} (68), e_{18} (68), e_{17} (74), e_{16} (84)	Vertex l (Head of e_{19}) already labeled		
discard	e_{18} (68), e_{17} (74), e_{16} (84)	Vertex l (Head of e_{18}) already labeled		
discard	e_{17} (74), e_{16} (84)	Vertex j (Head of e_{17}) already labeled		
discard	e_{16} (84)	Vertex k (Head of e_{16}) already labeled		
The Q is empty, the Algorithm terminated (one-to-all problems end-criterion)				

The SGO algorithm updates predecessor value for each of the reachable vertices $u \in V$ in the input graph as shown in Table 4. The SGO algorithm assign tailing vertex of a particular incident edge constituting shortest path as predecessor $u.\pi$ to the heading

vertex for each reachable vertex from source so that shortest path from a to l can be traced backwards with the chain of predecessors originating at vertex l (Russell & Norvig, 2016).

Table 4.

Predecessor list of each reachable vertex from source.

Vertex	a	b	c	d	e	f	g	h	j	k	l
$u.\pi$	-	a	a	e	c	a	e	d	g	f	j

Performance evaluation

The objectivity of the comparison can only be achieved when all the competing algorithms tested in similar experimental environment. We have determined the running time efficiency of the competing algorithms executed in similar computer environment, and same input model and time measuring method.

Experimental setup

The specifications of the computer used for the experimentation are Intel(R) Core i7-4700MQ CPU, 2.4 GHz and 8 GB memory installed, and running Microsoft Windows 8. Java standard library JGraphT package contains a collection of state-of-the-art shortest path algorithms and graph implementation methods. The state-of-the-art Java language implementations of Fibonacci and Binary heap variants of DA available in JGraphT were selected for execution in competition and large-scale road network graphs were also modeled and analyzed using same library implementations. We have already implemented the SGO algorithm in Eclipse 2022-03 packages using JGraphT and JHeaps java standard libraries tool (Charan et al., unpublished) and executed the same for optimal path calculations in automotive navigation system for intercity and intra-city road network graphs.

A graph may be constructed from a network data in different ways. The constructing and loading of the graphs is beyond scope of shortest path calculations, therefore the time taken for such activities is not consider; because, it is irrelevant to the elapsed running time of shortest path computation. Our calculations of the elapsed time start from the first iteration of minimum AD value edge selected from Q to the point at which the labeling of all connected vertices from the graph's start vertex is completed for the shortest path problems.

Workloads

A diverse set of graphs is pivotal for investigation and assessment of graph algorithms. We assessed the competing algorithms' performance for intercity workload of five USA States road networks from the available USA states downloaded from DIMACS implementation challenges for experimentation. NE Northeast is the largest road network graph among the selected states. It consists 1524453 vertices and 3897636 edges. Weights of the edges are integers in each of the input graph. Similarly, world's five metropolitan intra-city road network graphs downloaded from figshare dataset where New York City graph consists 439739 vertices and 624384 edges. The intercity / intra-city road network graph datasets are tabulated in Table 5.

Table 5.
USA states Intercity and World's five metropolitan cities Intra-city road networks.

Area	*Description*	*Vertices*	*Edges*
			Intercity road networks
COL	Colorado	435666	1012400
BAY	San Francisco Bay Area	321270	794830
FLA	Florida	1070376	2687902
NW	Northwest USA	1207945	2820774
NE	Northeast USA	1524453	3897636
Intra-city road networks			
Karachi	A city of Pakistan	37584	55332
Seoul	The capital of South Korea	105454	306224
Buenos Aires	The capital of Argentine	266314	434061
Tokyo	The capital of Japan	354297	1035447
New York	A city of USA	439739	624384

Results and Comparisons

DA Fibonacci, DA Binary and the SGO algorithms were tested on five intercity as well as intra-city input road network graphs. Multiple execution results recorded for each graph and 10 ordinary results were used for analysis purposes. The running time performance of the competing algorithms for shortest path computations on the mentioned workload is shown in Figure 2(a) and 2(b). The Figures exhibit that the SGO algorithm is empirically faster in comparison to the competing algorithms that supports the claim of running time efficiency of the SGO algorithm.

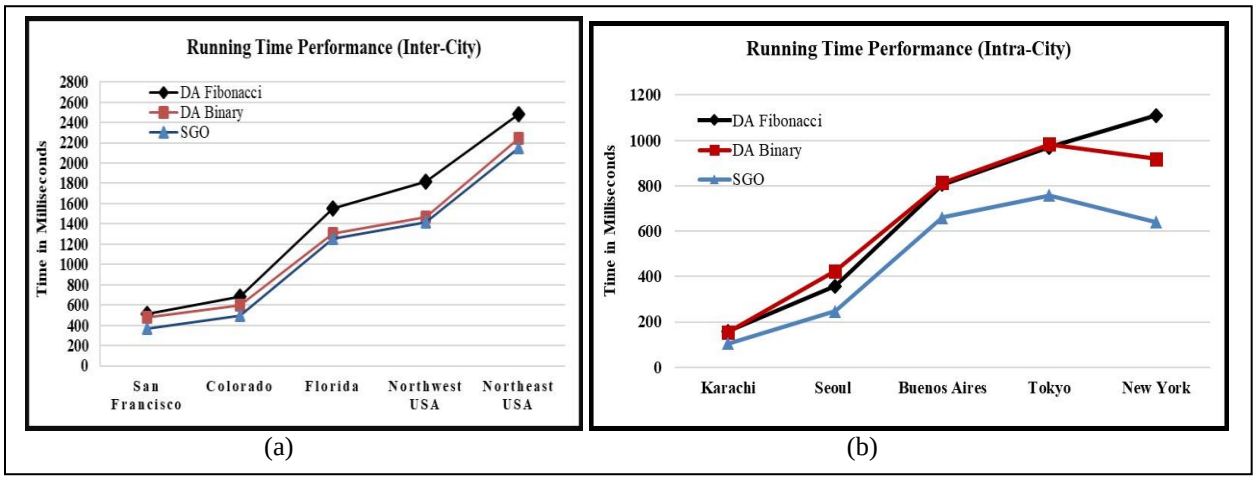
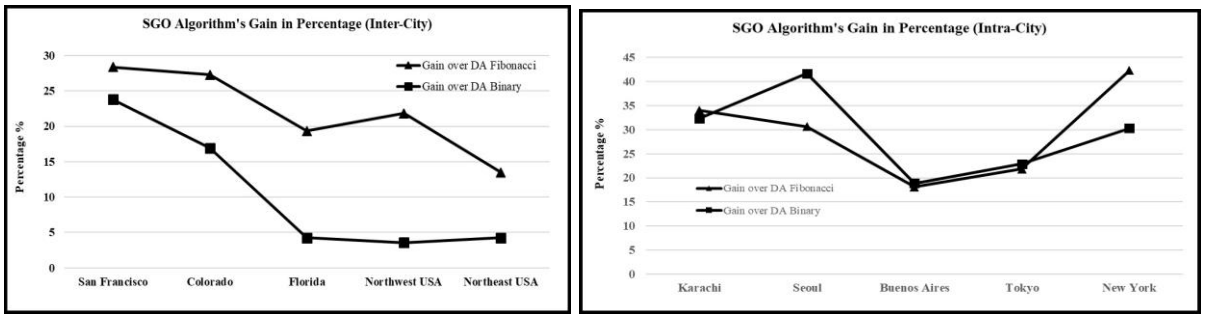


Figure. 2.
Running time performance (Intercity and Intra-city).

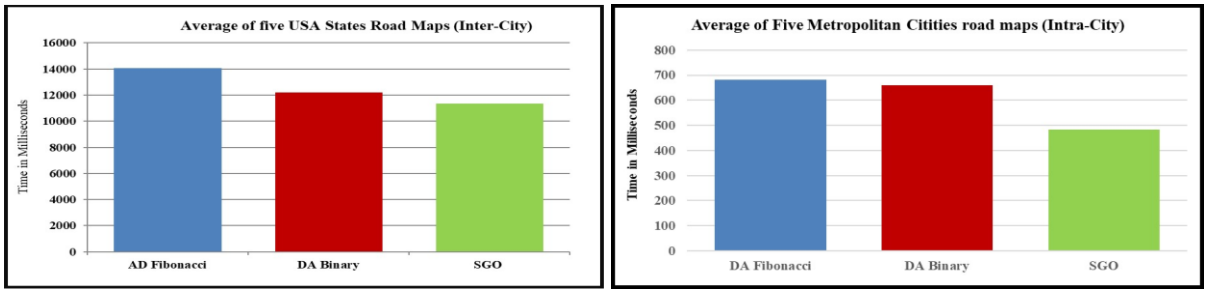
The time efficiency achievements, in terms of percentage, of the SGO algorithm over competing algorithm for intercity / intra-city network graphs is depicted in Figure 3. The Figure presents the time efficiency achievements where the maximum time efficiency achievement of SGO algorithm is ≈30 percentage for an intercity input graph of over 0.3 Million nodes and it reaches to ≈45 percentage for an intra-city input graph of over 0.4 Million nodes. The percentage gain is narrowing for a substantially huge size input graphs of over Million nodes, however the SGO algorithm is efficient all the times. The overall time efficiency gain of minimum ≈19 percentage and minimum ≈5 percentage is recorded for intra-city and intercity road graphs, respectively, for even a huge increase in the number of nodes of the input graphs.



(a) (b)

Figure. 3.
SGO algorithm's % gain over Fibonacci and Binary Heap variants (Inertcity and Intra-city).

Figure 4 displays the average of the performances of each competing algorithm for all the five intercity and intra-city road graphs. The results indicate that the SGO algorithm has better performance achievements in comparison to its competing algorithms; thereby supports the algorithm's proclaimed running time superiority for real-world road network graphs, as the algorithm outperformed its competitors empirically, in terms of running time performance. The SGO algorithm, therefore, is more practical for shortest path planning in automotive navigation system applications for intercity and intra-city road network graphs.



(a) (b)

graphs.

Figure. 4.
Average running time efficiency for five Intercity and Intra-city road graph input

CONCLUSION

The studies demonstrated that large and intricate real-world road network graphs are sparse in nature. This article has tested SGO, a simple and running time efficient algorithm, to address SPP in automotive navigation system applications. The SGO algorithm calculates total distance from start location to end of each edge (road segment) and prioritizes the edges accordingly. Thus eliminates the rearrangement cost of the priority queue, resulting in more efficient shortest path algorithm even the priority queue is implemented using Binary heap data structure. The SGO algorithm and Fibonacci & Binary heap variants of the DA were tested upon real-world road network graphs for running time efficiency. The performance evaluations, according to the experimental results presented in section 4, demonstrate that the SGO algorithm outperformed the variants of state-of-the-art Dijksrta's algorithm for intercity as well as intra-city road

network graphs. Empirical running time superiority suggests that the SGO algorithm will be more practical for optimal path planning in automotive navigation systems.

DECLARATIONS

Acknowledgement: We appreciate the generous support from all the contributor of research and their different affiliations.

Funding: No funding body in the public, private, or nonprofit sectors provided a particular grant for this research.

Availability of data and material: In the approach, the data sources for the variables are stated.

Authors' contributions: Each author participated equally to the creation of this work.

Conflicts of Interests: The authors declare no conflict of interest.

Consent to Participate: Yes

Consent for publication and Ethical approval: Because this study does not include human or animal data, ethical approval is not required for publication. All authors have given their consent.

REFERENCES

Abd Algfoor, Z., Sunar, M. S., & Abdullah, A. (2017). A new weighted pathfinding algorithms to reduce the search time on grid maps. *Expert Systems with Applications*, *71*, 319-331.

Alshammrei, S., Boubaker, S., & Kolsi, L. (2022). Improved Dijkstra algorithm for mobile robot path planning and obstacle avoidance. *Comput. Mater. Contin*, *72*, 5939-5954.

Beamer III, S. (2016). *Understanding and improving graph algorithm performance*. University of California, Berkeley.

Charan, K. N., Dhomeja, L. D., Memon S. A, Mahesar A. W., Rajput S. Nanada A., & Ismail M. (unpublished). Towards An Efficient Approach to Address Single-Source Shortest Path (SSSP) Problems.

Charan, K. N., Memon, S. A., Dhomeja, L. D., & Memon, N. A. (unpublished). An Efficient Single Source Shortest Path Algorithm for Sparse Graphs.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms. Thrid Edition. *London: Massachusetts*.

Danilovic, M., Vasiljevic, D., & Cvetic, B. (2021). A novel pseudo-polynomial approach for shortest path problems. *Networks*, *78*(2), 107-127.

de Bruin, G. J., Veenman, C. J., van den Herik, H. J., & Takes, F. W. (2021). Supervised temporal link prediction in large-scale real-world networks. *Social Network Analysis and Mining*, *11*(1), 80.

Demetrescu, C., Goldberg, A., & Johnson, D. (2006). 9th DIMACS implementation challenge–Shortest Paths. *American Mathematical Society*.

Dijkstra, E.W., 2022. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy* (pp. 287-290).

Haria, V., Shah, Y., Gangwar, V., Chandwaney, V., Jain, T., & Dedhia, Y. (2019). The working of google maps, and the commercial usage of navigation systems. *International Journal of Innovative Research in Technology*, 6(5), 184-191.

Jain, P., Patel, D., & Verma, J. P. (2022). Shortest pathfinder for air traffic network: A graph-based analysis. In *Proceedings of the International e-Conference on Intelligent Systems and Signal Processing: e-ISSP 2020* (pp. 699-712). Springer Singapore.

Koritsoglou, K., Tsoumanis, G., Patras, V., & Fudos, I. (2022). Shortest path algorithms for pedestrian navigation systems. *Information*, *13*(6), 269.

Korte, B. H., Vygen, J., Korte, B., & Vygen, J. (2011). *Combinatorial optimization* (Vol. 1, pp. 1-595). Berlin: Springer.

Liu, X., Khan, K. N., Farooq, Q., Hao, Y., & Arshad, M. S. (2019). Obstacle avoidance through gesture recognition: Business advancement potential in robot navigation socio-technology. *Robotica*, *37*(10), 1663-1676.

Ngandu, E. M., Luwes, N., & Kusakana, K. (2020, July). Navigation system for an automatic guided vehicle. In *Journal of Physics: Conference Series* (Vol. 1577, No. 1, p. 012030). IOP

Publishing.
Niu, S., Chen, S., Su, B., Cheng, X., Liu, J., ... & Li, Y. (2019). A smart city used low-latency seamless positioning system based on inverse global navigation satellite system technology. *International Journal of Distributed Sensor Networks*, *15*(9), 1550147719873815.
Pósfai, M., & Barabási, A. L. (2016). *Network science*. Cambridge, UK:: Cambridge University Press.
Road Networks, Urban (2016). Urban Road Network Data. figshare. Dataset. https://doi.org/10.6084/m9.figshare.2061897.v1
Rossi, R. A., & Ahmed, N. K. (2016). An interactive data repository with visual analytics. *ACM SIGKDD Explorations Newsletter*, *17*(2), 37-41.
Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: a modern approach*. Pearson.
Sariff, N., & Buniyamin, N. (2006, June). An overview of autonomous mobile robot path planning algorithms. In *2006 4th student conference on research and development* (pp. 183-188). IEEE.
Sedgewick, R. (2013). *An introduction to the analysis of algorithms*. Pearson Education India.
Song, T., Capurso, N., Cheng, X., Yu, J., Chen, B., & Zhao, W. (2017). Enhancing GPS with lane-level navigation to facilitate highway driving. *IEEE Transactions on Vehicular Technology*, *66*(6), 4579-4591.
Wang, D., Yin, G., & Chen, N. (2021). Optimisation of dynamic navigation system for automatic driving vehicle based on binocular vision. *International Journal of Industrial and Systems Engineering*, *39*(3), 411-428.
Wei, H., Zhang, S., & He, X. (2020). Shortest path algorithm in dynamic restricted area based on unidirectional road network model. *Sensors*, *21*(1), 203.
Yang, Y., Liu, W., & Xu, X. (2022). Identifying Important Ports in Maritime Silk Road Shipping Network from Local and Global Perspective. *Transportation Research Record*, *2676*(12), 798-810.
Yasrab, R., & Pound, M. P. (2021). CNN based Heuristic Function for A* Pathfinding Algorithm: Using Spatial Vector Data to Reconstruct Smooth and Natural Looking Plant Roots. *bioRxiv*, 2021-08.
Zhang, Y., Tang, G., & Chen, L. (2012, August). Improved A* algorithm for time-dependent vehicle routing problem. In *2012 International Conference on Computer Application and System Modeling* (pp. 1341-1344). Atlantis Press.
Zhao, Y., Lmg, Y., & Xia, P. (2021, March). Intelligent vehicle navigation system based on visual detection and positioning. In *Journal of Physics: Conference Series* (Vol. 1861, No. 1, p. 012115). IOP Publishing.
Zheng, C., & Green, R. (2010). Vision-based autonomous navigation in indoor environments,‖ 25th International Conference of Image and Vision Computing New Zealand (IVCNZ). *New Zealand, Queenstown*.
Zhou, M., & Gao, N. (2019, April). Research on optimal path based on Dijkstra algorithms. In *3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019)* (pp. 884-892). Atlantis Press.