Journal of Informatics
Electrical and Electronics Engineering, 2022,
Vol. 03, Iss. 01, S. No. 007, pp. 1-17
ISSN (Online): 2582-7006

A Novel Optimized Travel Planner

Rahul Srivastava[1], Pawan Singh[2]

[1,2]Department of Computer Science Engineering, Amity School of Engineering and Technology, Amity University Uttar Pradesh, Lucknow Campus, India
[1]rahul.srivastava12134@gmail.com, [2]pawansingh51279@gmail.com

How to cite this paper: Author 1, P. Singh, "A Novel Optimized Travel Planner," *Journal of Informatics Electrical and Electronics Engineering* (*JIEEE*), Vol. 03, Iss. 01, S No. 007, pp. 1–17, 2022.

http://doi.org/10.54060/JIEEE/003.01.007

Received: 05/04/2022
Accepted: 24/04/2022
Published: 25/04/2022

Abstract

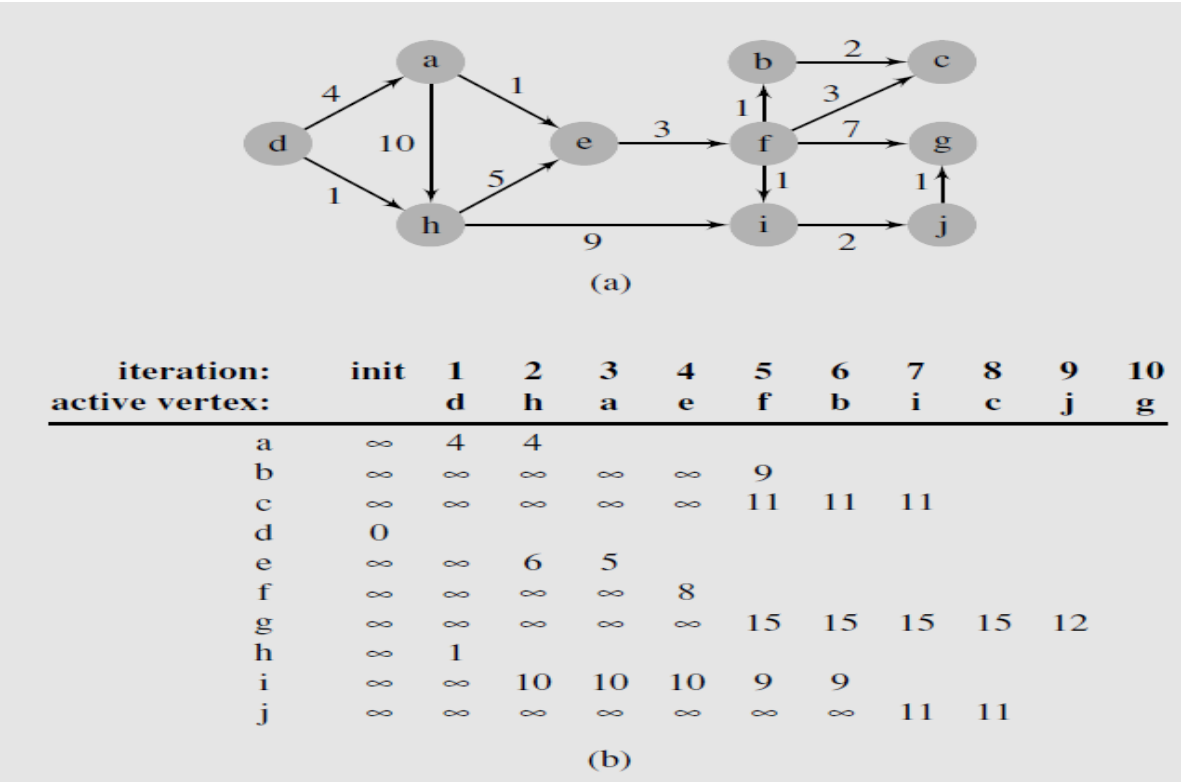
In 1959, Dijkstra announced a method for determining the shortest path between two nodes in a network. This algorithm has received much attention because it can deal with various real-life problems. The algorithm is a greedy type. Other types of algorithms are also created and compared. Application-related changes are made to the original algorithm. The time complexity of the algorithm increases at the expense of space complexity. Space is more complex with modern hardware, so that you can implement such an algorithm.

Keywords

Dijkstra's algorithm, shortest path, graph, node, edge, time complexity, space complexity.

1. Introduction

Dijkstra‟s Algorithm (DA) gets a wide attention as it solves an important problem of graph theory, of finding out the "Shortest Path (SP)". For a graph with each edge having a weight or path length the algorithm determines the SP between two selected vertices. In street network applications, city crises take care of driving guidance systems, where the best route must be established, and shortest route difficulties are inevitable. Because the traffic situation in a city varies from time to time & there are frequently many requests at any given time, it is necessary to find a solution immediately. As a result, the algo's efficiency is critical [1, 3, & 5]. Preprocessing, which computes results before demanding, is a technique used. These findings are stored in memory & they can be used immediately as when a new query is received. This may not be possible if the devices have limited memory. The research will only work on single-source shortest route issues, analyzing several algorithms such as Dijkstra's shortest route algo, Restricted search method, and A* algo to draw relevant generalizations. To verify the three algorithms, a JavaScript application was constructed. The three ways were put to the test and visibly demonstrated. To better understand the algorithm, we need to look at the specific graph.



iteration:	init	1	2	3	4	5	6	7	8	9	10
active vertex:		d	h	a	e	f	b	i	c	j	g
a	∞	4	4								
b	∞	∞	∞	∞	∞	9					
c	∞	∞	∞	∞	∞	11	11	11			
d	0										
e	∞	∞	6	5							
f	∞	∞	∞	∞	8						
g	∞	∞	∞	∞	∞	15	15	15	15	12	
h	∞	1									
i	∞	∞	10	10	10	9	9				
j	∞	∞	∞	∞	∞	∞	∞	11	11		

(b)

Figure 1. (a) An example graph. (b) Execution steps of Dijkstra's Algorithm [1]

The goal is to find the SP between nodes d and j. The number of iterations is shown in the first row of the table in fig. 1b. All distances from source d to nodes other than d are considered infinite for the unit or 0th iteration. The sign here stands for "unknown. "In computation, a considerable number is put in place of infinity to refuse the unknown. The path length from node d to node d is 0. The first column of the table contains these values.

The distances between the source and nearby nodes replace the initial infinite value in the first iteration. The nearby nodes in this example are "a" and "h." As seen in picture 1, the distance from node d to h is 1, and from the node a to an is 4. The active vertex in the second iteration is h since it has the smallest distance from d.

The total of the distances from d to h and h to nearby nodes is now calculated, and the direct path, if any, is compared. The third column has a smaller value. For example, the distance from d to e is calculated as the distance from d to h (1) + the distance from h to e. (5). As a result, the total distance in the 5th row and 3rd column of the table is 6. The distance between the active vertices h to a is the sum of the lengths d to h (1) and h to a (10) 11, or the straight path distance is 4. The smaller value, 4, is in the table's first row and first column. Since the third iteration, a has become the active vertex. The last path has been removed due to the d, 2 straight routes of active vertex hits. As a result, the absolute path is determined as d->a->e with path length 6. The iteration stops when the destination node j is an active vertex. Then the final route becomes d->a->e->f->i->j with SP length 11.

Determining SP is a fundamental task that needs to be completed from time to time in different places. This approach can determine the shortest distance between two points [2]. This figure shows all road or rail connections at accessible train

stations. If you want to handle a variable number of edges, you need to change the method. Some roads are closed due to repair work or bad weather.

If the length of the path can represent something else, the algorithm finds a greater use. If you can divide your project into many different tasks, you can use the nodes that represent the tasks to create the chart. The path length is replaced with the task execution time. In this situation, DA finds the fastest completion time for the project. Application-specific changes may be required. If a certain path length becomes negative, DA will stop functioning. Algorithms have been presented to address this type of problem [1]. As technology evolves, algorithms discover new uses. With the advancement of routing techniques in computer and mobile networks, algorithms are receiving a lot of attention. Despite the fact that the method was proposed many years ago [3,] several new proposals to improve it have just been presented. Various types of data structures are used to speed up execution.

More RAM is reserved to hold intermediate results that can be reused to reduce computation time. As a result, the time complexity is increased along with space complexity. This is a typical pattern in algorithm development. There may be greater space complexity as modern hardware can provide more significant memory. Another trend is to divide networks into subnets and apply algorithms to them. Since the time complexity is proportional to the square of the number of nodes, the time complexity improves significantly as the number of nodes is reduced. Other networks for which DA is improved, operating more efficiently and reducing time complexity have been noted. The DA algorithm is a greedy algorithm that seeks the best instant result. Other methods were also tested, and the results were compared. There are several proposed algorithms for SP, and their number is proliferating. Several algorithms can be used for SP compared in this article.

For real road networks, determination of SP from place to place is extremely important. Zhan [4] compared 15 algorithms for this application. Following table 1 lists all these algorithms.

Table 1. 15 algorithms for real road networks, determination of SP from place to place

Abbreviation	Implementation
BFM	Bellman-Ford-Moore
BFP	Bellman-Ford-Moore with Parent--checking
DKQ	Dijkstra's Naive Implementation
DKB	Dijkstra's Buckets -- Basic Implementation
DKM	Dijkstra's Buckets -- Overflow Bag
DKA	Dijkstra's Buckets -- Approximate
DKD	Dijkstra's Buckets -- Double
DKF	Dijkstra's Heap -- Fibonacci
DKH	Dijkstra's Heap -- k--array
DKR	Dijkstra's Heap -- R--Heap
PAP	Graph Growth -- Pape
TQQ	Graph Growth with Two Queues -- Pallottino
THR	Threshold Algorithm
GR1	Topological Ordering -- Basic
GR2	Topological Ordering -- Distance Updates

All of these algorithms were written in the C programming language. The TQQ algorithm has the most outstanding performance for one-to-all SP. Because it may be stopped as soon as the destination node is reached, DA has certain advantages

for one-to-one or one-to-several SPs. Two DA variants, DKA and DKB, were implemented. DKA works well for paths less than 1500 meters. More than 1500 DKB is preferable. The implemented algorithms demonstrate this such behavior is beyond comprehension. If DKB is required for paths less than 1500 meters, the lengths can be ramped up to reach 1500 meters. This paper makes no such recommendation. DKA has an O(mb+n(b+C/b) worst-case complexity.

Arjun et al. [5] used an efficient data structure to enhance DA. The SP is found via heap sort in this case. The temporal complexity of heap sort is log(n), where n is the number of nodes to sort. Magzhan and Jani [6] studied and assessed four strategies for determining SP. The time complexities of the three algorithms are listed in the table below.

Table 2. time complexities of the three algorithms

Algorithm	Time complexity
Dijkstra	$n^2 + m$
Bellman-Ford	n^3
Floyd-Warshall	nm

** n->No. of nodes. m->No. of edges

The time complexity of the fourth algorithm, the genetic algorithm, cannot be estimated because there are many random processes. However, "Choice" is a deterministic process. Other authors appreciate the complexity of time. Chromosomal complexity cannot be compared to the other three algorithms shown in the table above because they depend on the size of the population. SP was established by Betz and Rose [7] for component routing in an FPGA. They created VPR, a tool that can handle initial component placement and routing in an FPGA. The "Pathfinder Negotiated Congestion Algorithm" is used for routing. DA determines SP for nets with a small number of nodes in this approach. Then a modified version is used for global routing. The time complexity is considerably reduced because the approach only works for a small number of nodes per net.

Table 3. Routing tracks decided by various tools

Global R / Detail R.	LocusRoute[17]		GBP [20]	OGC [21]	IKMB [22]	VPR	TRACER [24]	VPR
	CGE[18]	SEGA[19]				SEGA[23]		
9symml	9	9	9	9	8	7	6	6
alu2	12	10	11	9	9	8	9	8
alu4	15	13	14	12	11	10	11	9
apex7	13	13	11	10	10	10	8	8
example2	18	17	13	12	1	10	10	9

k2	19	16	17	16	15	14	14	12
terml	10	9	10	9	8	8	7	7
too_large	13	11	12	11	10	10	9	8
vda	14	14	13	11	12	12	11	10
Total	123	112	110	99	94	89	85	77

VPR requires minimal tracks for all benchmarked blocks, according to the results. Huang et al. [8] increased DA by permitting higher spatial complexity. The flowchart of their algorithm is shown in the diagram below.

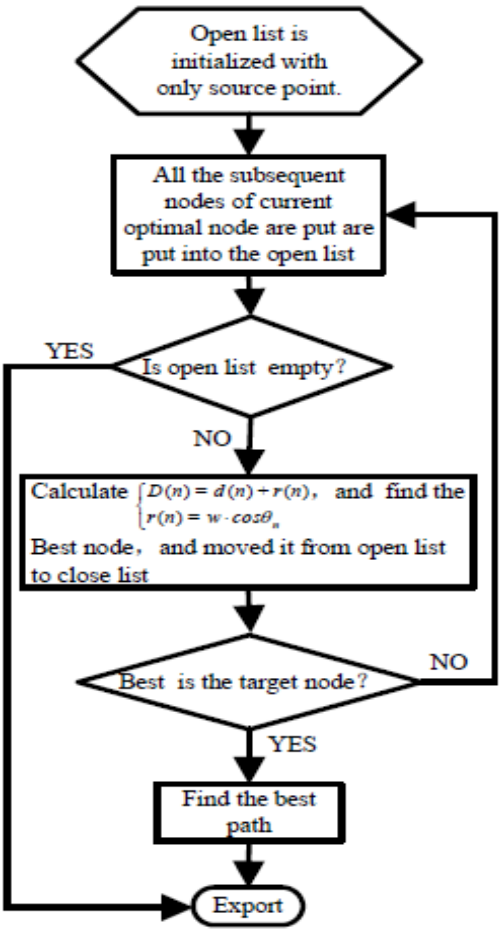


Figure 2: Flowchart of the developed algorithm [8]

Visual C++ is used to code the simulation. The experiment is conducted on a network of 50X50, 100X100, 200X200, and 250X250

dimensions. The following graphs show the comparisons.

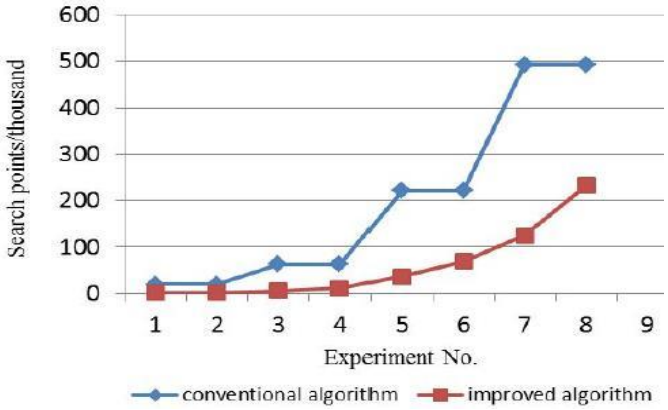


Figure 3: Search time comparison between proposed algorithm and DA [8]

With the proposition, both the number of search spots and the search times are considerably decreased. For more extensive networks, the improvement is more noticeable. Similarly, Sivakumar and Chandrasekar [9] enhanced the algorithm. The authors called their method MDSP (Modified Dijkstra's Shortest Path). They compared their approach to Dijkstra's algorithm using various bucket kinds. Dijkstra's algorithm with Approximate buckets (DKA), Dijkstra's method with Double buckets (DKD), and Dijkstra's algorithm with Buckets are the algorithms that were compared (DKB). The comparisons are presented in the following two tables.

Table 4: Searched nodes for various algorithms [9]

Algorithm	Nodes
MDSP	16
DKA	25
DKD	32
DKB	35

Table 5: Search time for different algorithms [9].

Algorithm	Time (in minuted)
MDSP	2
DKA	4
DKD	6
DKB	10

The suggested method has the quickest execution time, according to the results. With more excellent data storage, Kadry et al. [10] improved DA in the situation of primarily linked nodes to the active node. In this method, the number of iterations is lowered. This study does not provide a detailed comparison based on actual data. DA plays a crucial role in reducing data packet transmission delays in a network. Jain and Kumawat [11] looked at several forms of network delays and computed cost functions from node to node, which they reduced using DA. The delay caused by nodal processing was not considered in previous DA applications. The following graph depicts these two comparisons. In series 2, the narrative depicts the application of DA without taking into account the cost of processing time. The processing delay increases the cost of the plot. The plot in series 2 shows cost reduction via DA. Processing time is taken into account.

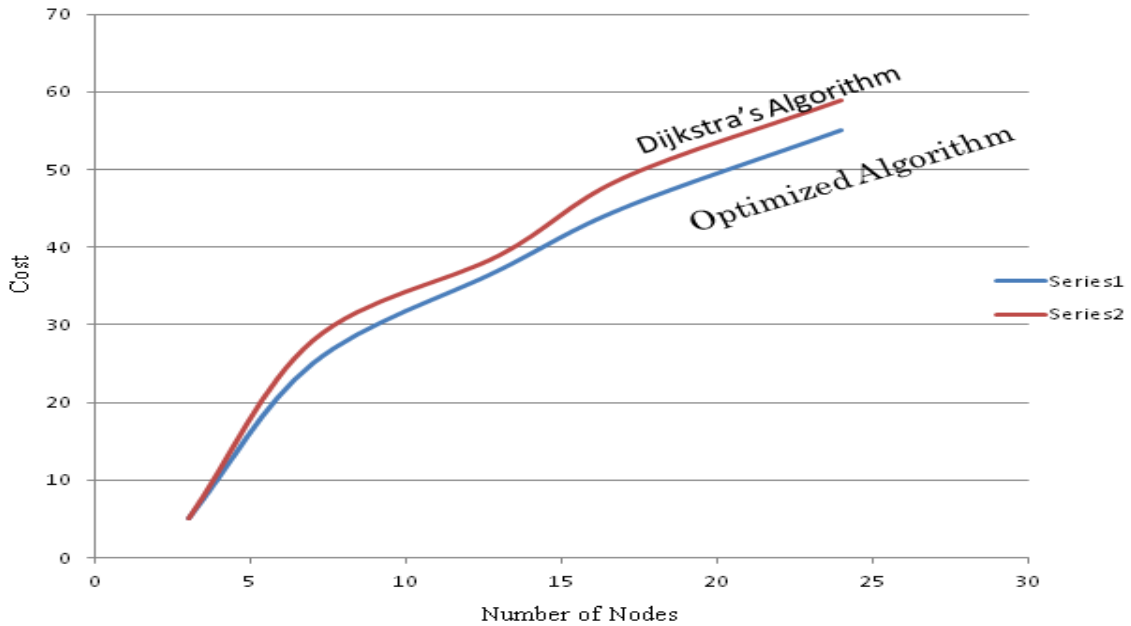


Figure 4: Comparison of cost functions with and without considering delay due to nodal processing [11]

After the calculation of execution time increases, the cost optimization will be improved. To locate the first node, Singal and Chhillar [12] used the global positioning system “GPS". The SP then uses the DA to locate the destination node. Their algorithm diagram is shown in the diagram below.

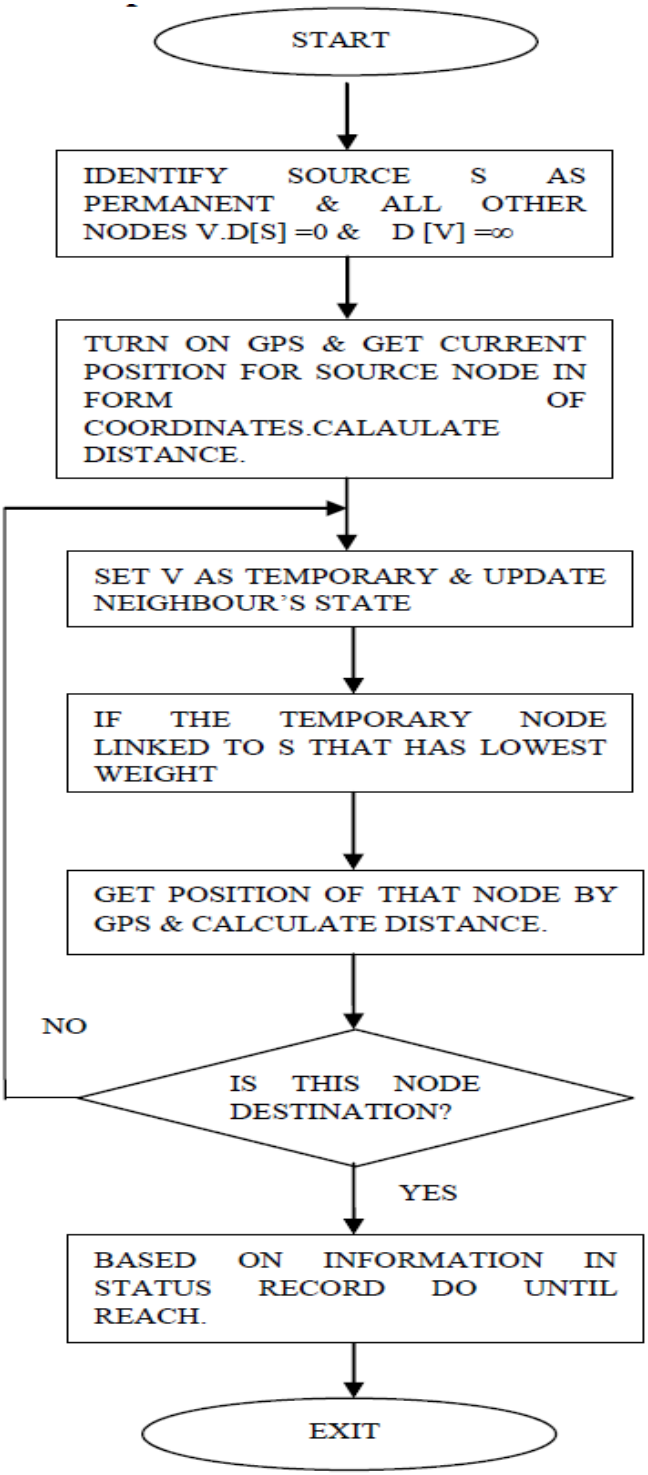


Figure 5: Flowchart of the proposed algorithm [12]

This flow is applied for the graph given in the following figure. Following table gives the resultant nodes in every step

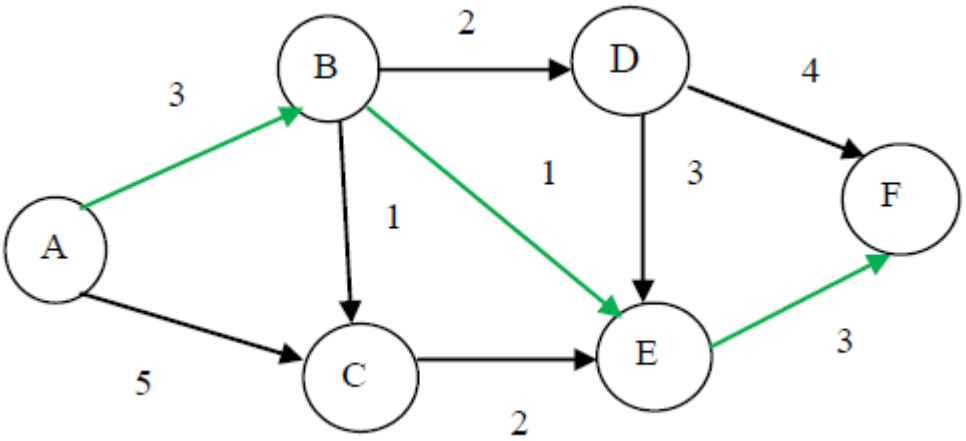


Figure 6: Graph considered for the proposed algorithm [12]

Table 6: Resultant nodes in every step [12]

Steps	N	Position	Distance	D(B), PATH	D(D), PATH	D(E), PATH	D(F), PATH
1	{A}	(2,3)	3,A-B	5,A-C	∞,-	∞,-	∞,-
2	{A,B}	(5,7)	3,A-B	4,A-B-C	5,A-B-D	4,A-B-E	∞,-
3	{A,B,C}	(8,11)	3,A-B	4,A-C	5,A-B-D	4,A-B-E	∞,-
4	{A,B,C,D}	(11,15)	3,A-B	4,A-C	5,A-B-D	4,A-B-E	7,A-B-E-F
5	{A,B,C,D,E}	(14,19)	3,A-B	4,A-C	5,A-B-D	4,A-B-E	7,A-B-E-F
6	{A,B,C,D,E,F}	(17,23)	3,A-B	4,A-C	5,A-B-D	4,A-B-E	7,A-B-E-F

Sniedovich [13] proposed a relationship between DP(dynamic programming) and DA. Management science and operations research will pay more attention to this. Orlin et al. [14] propose an efficient approach to graph specialization. When there are only a few distinct edges, the algorithm works fine. When there are n vertices, m edges, and K unique edge lengths, the time complexity is O(m) if nK2m and O(mlog(nK/m) otherwise.

2. Dijkstra's Shortest Route Algo

The shortest path Algorithms are frequently used to solve the quickest or fastest path from a single source. In graph G (V, E), multiple types of data structures are utilized to determine the path between two vertices; the running time to determine the route in between pair of vertices varies. Few of the data structures can increase the Time. This project replaces Dijkstra's method that uses the binary heap.

Dijkstra Algo: -

Loop node in Graph:

distance[node]=infinite

parent[node] =null.

End

distance[s] = 0;
//Initial Point

h = {s};
//Pushing int to Heap

While heap is not Empty, and value not Found:

Node = Extract the Minimum value form heap

Label node as visited.

loop each nodev adjacent to u:

if distance[v] is greater than distance[node]+
weight[node,v]+distance[nodev]=distance[node] +
weight[node, nodev];

parent[nodev] = node.

Decrement the value [node, H]

Time:

It takes o seconds to run in the first loop (V). The minimum removal of the heap in each iteration of the loop is logV. The inner loop goes to the neighboring nodes of each current node, taking O seconds in total (E). As a result, the algo's temporal complexity is o ((v + e) *log(v) =o (e* log(v)). The algo's accuracy is thoroughly demonstrated.
The no. of nodes in a network grows, so does the time it takes for the applicable algo to run. A city's road network often contains more than 104 nodes. It becomes more desirable to have a quick shortest route algo.

2.2 Restricted Search Algo

The organization of road networks is, in general, quite essential. They are a sparse, linked graph with a big scale. When the Dijkstra method is employed to determine the shortest route, it begins & extends out in a circle until the radius reaches the destination. Most searches are ineffective in a region opposite to the direction of travel. M. Fu et al. [4] proposed a strategy for finding the shortest route for a vehicular navigation system that involves physically closing off areas where the shortest route is not expected to occur. This project uses the algo to achieve this goal. Restricted Search Algo:

Restricted method explains the less area of the left segment of the rectangle Rectangle2, which is cut by two bold lines rather than the entire circle. Rectangle1 is a rectangle along the diagonal of S and D, and Rectangle2 is a rectangle scaled from Rectangle1 to the threshold thresh2. With a distance of t1, two straight, bold lines are parallel to the straight line of SD. thresh1 and thresh2 are two factors that must be determined for the best route to be included in the restricting area that usually ranges between 500 to 1500m

Following algo is used reach the goal

Searching Algorithm:

```
loop each node Graph:
        distance[node]=infinite
        parent[node] =NULL.
End loop
distance[s] = 0.
 H = {s};
While heap is Not Empty(H) and Value is Not Found:

        u = Extract Minimum element from (H);

        label u as visited;

        loop each v adjacent to u:

                If u is out Of Range(v),

                        then continue;

                if distance[v] > distance[u] + weight[u, v],

                        then

                        distance[v] = distance[u] + weight [u, v];

                        parent[v] = u;
        Decrease the Key[v, H];
```

Steps if out Of Range

Draw a straight line from the right of the vertex.

count = 0;

loop each edge of area

```
        if intersect with edge
                //Increment count by 1;
                count++;

        if count%2== 0
                (Is even)
                return1;//true returns
         Else
        return 0; //false returns
```

Dijkstra's algo is implemented in the same way. In each loop, instead of resting all neighboring nodes, the method

finds out nodes outside the bounding region by analyzing whether they are out of bounds. The inspection technique counts the number of intersections between the restricted area & the horizontal line running from the node to the right. This strategy shows, if the number of meetings is odd, then the node is within this region; Otherwise, it is not so.

Time:

K1, K2: these are the sole reason for the threshold.
So, the ratio R can be used to improve the productivity,
R = v/V*= c* Optimal / c*
<= 2 * T1 (R + T2 * √2) / πR²
= 2 * K1*R (R + K2* R*√2) / πR² =2 * K1(1 + √2 * K2) / π
Since K1 & K2 are small numbers, great improvement can be achieved. For instance, if K2 = 0.4, P ≈ K1.

Correctness:

Two problems exist in this approach.

1. Threshold may not get the solution properly. As distance increases from the start point to the destination, the shortest route more likely spreads wider from the straight line of SD.
2. Distinct city traffic lines exist, each with different driving speeds—the highway takes you beyond restriction & is also the shortest.

This project uses relative thresholds rather than set ones, referred to as factor K1, K2 (threshold / SD length), to obtain a better result. To make the problem easier to understand, they have the same value implementation. The threshold will increase proportionally with distance from the start point to the destination in each factor. The second issue could be overcome by assigning each node a logical position rather than a physical location. When a node is investigated, its logical place is calculated by multiplying its parent's logical area by the cost of the edge between them. The cost of an edge is the logical distance, which is measured in terms of physical distance & route type.

The logical location of the start point is the same as its physical location. All nodes being examined use their logical position to decide if they are within the restricted area. This ensures that the nodes on different roads can be treated equally in searching. This algo uses the Dijkstra within the restricted area. The shortest route in this area will be found in the existing route. However, this shortest route may not be the shortest one in the whole area. Thus, it is an optimal route with a restricted area.

2.3 A* Search

A heuristic is consolidated into the A* calculation's pursuit technique. Rather than picking the following hub with the most minimal expense (as evaluated from the beginning hub), the cost from the beginning hub is joined with a gauge of closeness to the objective (a heuristic gauge). This answer for tackling the ideal way of revelation was portrayed by F. Engineer [2]. This task utilizes Euclidean distance as assessed distance to the objective. In the looking, the expense of a hub V could be determined as:

$$f(V) = \text{distance from to V + estimate of the distance to D.} = d(V) + h(V, D)$$
$$= d(V) + \text{sqrt}((x(V) - x(D))^2 + (y(V) - y(D))^2)$$

where x(V), y(D) and x(V), y(D)are the coordinates for node V and the destination node D.

A* Search algo->

```
for each i G:

        v[i]=infinite.
        //Initialization

        par[i] =NULL.
        //initializing parent null

End

v[i] = 0; F(v)= 0; h = {i};
//Heap is initialized

while Not Empty(H) and value Not Found: u = Extract minimum element from    the heap (h).

        mark i as visited,

        loop each v dja to u:

                if x[v] > x[u] + weight [u, v],

                        x[v] =x[u] + weight [u, v];

                         parent[v] = u;

                        F(v) = x[v] + h (v, D);
                        Decrement the Value[v,h]
```

Time: Approach improves avg time rather than worst-case Time. The shortest route search begins at the starting location & continues until it reaches the node that leads to the destination. As a result, the method takes substantially less time to run than Dijkstra's.

Correctness: This method adopts a strategy like Dijkstra's, but it adds Euclidean distance & the cumulative edge cost between the current & destination nodes. This value is used to determine the location of the node in the minimum heap. The one with the lowest value will be chosen & discarded from the heap. This value solely impacts the searching order in the implementation. It does not change the edge weights or the total distance. When a node relaxes, the accumulated distance is updated in the same way as Dijkstra. This method is identical to Dijkstra's & is valid.

3. Implementation

The JavaScript programming language was used to create this software. The three strategies were put to use and visually demonstrated. A graph data file containing partial transportation data for Ottawa is used in the road network example.

3.1 Brief Description

In the diagram, there are four types of roads: small roads, regional roads, major roads, & highways. The maximum driving speed varies depending on the kind of road. As a result, the physical distance between the two sites is insufficient to characterize the journey. Instead, the physical distance & the shortest route are represented by a logical distance. By multiplying the actual distance by the road section, the logical distance may be computed.

3.2 Main Feature

It has the following main features:

1. The road network loads & appears in the window. The file “Ottawa city. Gph” contains this map.
2. The user used a mouse to drag & drop between the two nodes' window screens.
3. Windows is able to display the quickest route if someone requests it.

In this graph, there are four types of roads: minor, regional, significant, & highway, depicted in various colors.

3.3 Run program

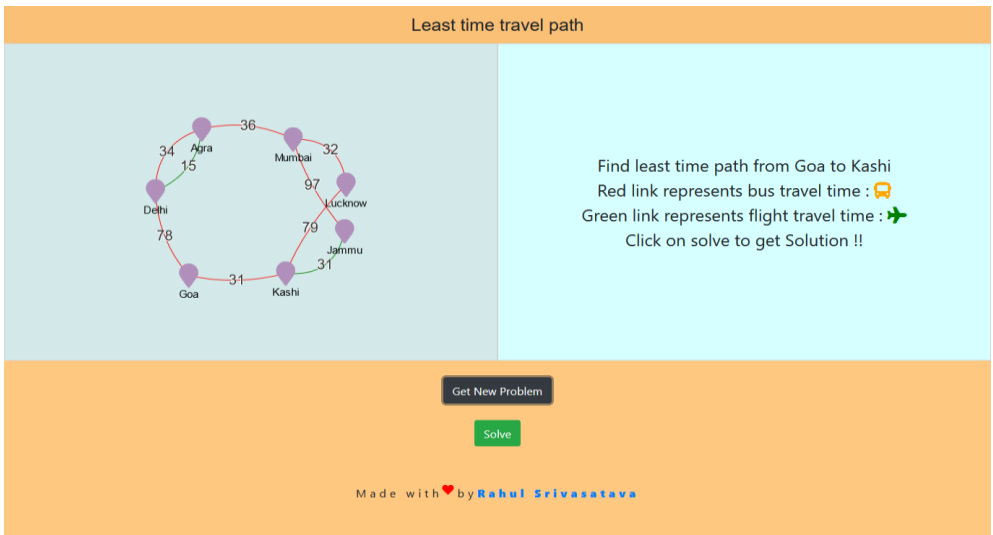


Figure 7. Screenshot of the Algorithm Implemented

Solved Problem: -

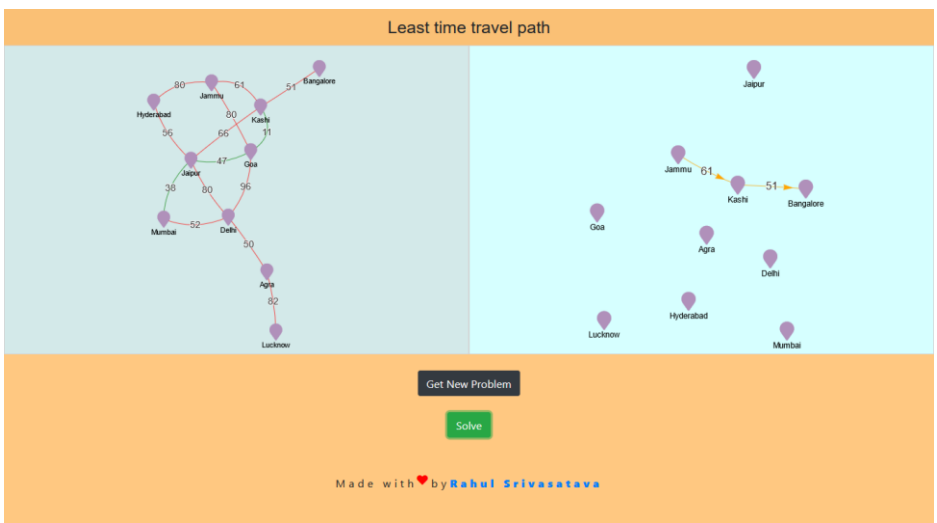


Figure 8. Screen Shot of the Solved problem

4. Experimentation

The initial segment looks at the run times & exactness of the three calculations. Distance is the length of the way, addressed by the consistent distance as examined in segment 3.1. The equivalence of an estimate for a specific distance is the way length is found by this calculation, isolated by the way length is found by Dijkstra. The A* algo outperforms the Dijkstra algo. Because it uses the Euclidean heuristic function to limit its search region towards the target, it generally decreases the running time in half. Because of the same rationale as before, the cutting rate reduces as the distance increases.

5. Analysis

- Dijkstra's algorithm has attracted a lot of attention because it can solve various real-world problems.
- Negative and irrational edge lengths are addressed using more generic methods.
- It appears that DA can be used directly in some situations, such as determining the shortest distance between two cities connected by roadways. In this instance, DA adjustment is also required. A road may be closed for safety reasons due to inclement weather. The rest of the road work remains unchanged. DA must be applied to a portion of the graph.
- The inaugural node may be located using GPS technology.
- For the inclusion of the starting node and the calculation of Euclidean distances, a tweaked method is required.
- For a particular arrangement, improvement may be beneficial. The modifications that may be made to a graph with too many edges are considerably different from those that can be made to a graph with a small number of edges of varying lengths.
- A vast network can be fragmented into smaller overcrowded networks. A detailed routing for smaller networks followed by a global route for the entire network shows to be quite economical for FPGA.
- DA can be used to route data packets in a network. The time spent on data transport from node is shown below. In place of the distance, the node should be considered. All potential delays should be evaluated and accounted for. Give the delay or cost function that is effective. It's worth doing at the very least. the problem deserves attention. The latency of the researcher varies according to the channel and other data packets' location and transmission.
- The algorithms for determining the shortest path are included in the table below. The complexity of available time is contrasted.

Algorithm /Inventor's Name	Special Features	Time Complexity	Source of Information
Dijkstra's Algorithm	Determines the shortest path between any two nodes in a network.	n2 + m = O(n2)	[1, 6].
L. Ford	Negative edge length is possible.	O(nm)	[1]
Gallo and Pallottino	Irrational edge length can occur.	O(2n)	[1]
DKA	Time complexity decreases as space complexity increases.	O(mb+n(b+C/b))	[4]

Arjun et al.	For networks with large number of edges.	log(n) for heap sort. n2 + log(n) in total.	[5]
Bellman Ford	Networks with a low node count	n3	[6]
Floyd Warshall	Networks having a limited number of edges	nm	[6]
Genetic Algorithm	To be used in large networks. The chances of obtaining the correct answer are great. However, the likelihood is always smaller than one.	Random processes cannot be determined. It can only be used for the process of "Selection." However, it is connected to population size. It has nothing to do with network specifications.	[6]
Pathfinder Negotiated Congestion Algorithm	For comprehensive routing in a crowded network with a small number of nodes. A network with a tiny n and a commensurate m.	Not reported	[7]
Orlin et al.	For a unique network with a limited number of different edges.	O(m) if nK≤2m and O(mlog(nK/m)) otherwise.	[14]

n ->No. of nodes.
m-> No. of edges.
b A chosen constant. Each node can be touched b times at most

6. Conclusion

Many algorithms are compared to Dijkstra's algorithm. If available, time complexities are compared. The given simulation times are compared. Application-based enhancements are scrutinized. With the development of technology, algorithms will find additional uses. The application requires dedicated research and algorithm-based enhancements. Better hardware will improve time complexity at the expense of space complexity. There will be additional storage capacity. Using the Euclidean heuristic function, the A* algo can achieve a faster running time, even though its theoretical Time is like Dijkstra's. It can also ensure that the shortest route is found. The limited method can discover the best route in linear time, but the confined area must be appropriately chosen. The choice is made based on the graph itself; the faster the search, the smaller the search zone, but at

the cost of not finding the shortest or no route. If the first search fails, this strategy, by increasing the factor, may allow a second search.

Reference

[1]. D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, "A survey on algorithmic approaches for solving tourist trip design problems", Journal of Heuristics, vol. 20, no. 3, pp. 291-328, 2014.

[2]. C. Chung-Hua, H. Chenyang, "A Platform for Travel Planning by using Google Maps", 16th IEEE Int. Conf. Mobile Data Management, vol. 2, pp. 120-125, Jun. 2015.

[3]. F.B. Zhan, C. Noon, "Shortest Path Algorithms: An Evaluation Using Real Road Networks", Transportation Science, vol. 32, no. 1, pp. 65-73, November 1996.

[4]. D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, "Mobile recommender systems in tourism", Journal of Network and Computer Applications, vol. 39, pp. 319-333, 2014.

[5]. C. Chung-Hua and H. Chenyang, "A Platform for Travel Planning by using Google Maps", 16th IEEE Int. Conf. Mobile Data Management, vol. 2, pp. 120-125, Jun. 2015.

[6]. T. Angskun and J. Angskun, "A travel planning optimization under energy and time constraints", Int. Conf. Information and Multimedia Technology ICIMT, pp. 131-134, Dec. 2009.

[7]. E. H.-C. Lu, C.-Y. Lin and V. S. Tseng, "Trip-mine: An efficient trip planning approach with travel time constraints", Proc. IEEE 12th Int. Conf. Mobile Data Management ser. MDM '11, vol. 1, pp. 152-161, 2011.

[8]. W. C. Chia, L. S. Yeong, F. J. Xian Lee and S. I. Ch'ng, "Trip planning route optimization with operating hour and duration of stay constraints," 2016 11th International Conference on Computer Science & Education (ICCSE), 2016, pp. 395-400, doi: 10.1109/ICCSE.2016.7581613.

[9]. A.V. Goldberg, “A simple shortest path algorithm with linear average time”, in: ESA, 2001, pp. 230–241.

[10]. J. B. Orlin, K. Madduri, K. Subramani, and M. Williamson, “A faster algorithm for the single source shortest path problem with few distinct positive lengths,” *J. Discrete Algorithms (Amst.)*, vol. 8, no. 2, pp. 189–198, 2010.

[11]. W. A. K. D. Wijesinghe, A. G. L. D. P. Amarasinghe, T. M. U. A. Bandara, Aruna Ishara Gamage, Devanshi Ganegoda, "VOYAGER – Smart Travel Guidance Cross Platform Mobile Application", *2021 3rd International Conference on Advancements in Computing (ICAC)*, pp.163-168, 2021.

[12]. Sumit S. Muddalkar, Nishant S. Chaturkar, Khushal D. Ingole, Shreyash B. Wadaskar, Rahul B. Lanjewar, "Electric Vehicle Charging Station Finding App", *International Journal of Advanced Research in Science, Communication and Technology*, pp.607, 2022.

[13]. Larisa Kuznetsova, Arthur Zhigalov, Natalia Yanishevskaya, Denis Parfenov, Irina Bolodurina, "Application of a Modified Ant Colony Imitation Algorithm for the Traveling Salesman Problem with Time Windows When Designing an Intelligent Assistant", *Advances in Intelligent Systems, Computer Science and Digital Economics*, vol.1127, pp.346, 2020.

[14]. Patrick Kaltenrieder, Jorge Parra, Thomas Krebs, Noémie Zurlinden, Edy Portmann, Thomas Myrach, *Designing Cognitive Cities*, vol.176, pp.235, 2019.

[15]. Baivab Maulik, Aditi P Nayak, Sanjana U, Simmi Alok, Divyaprabha K N, "Design and Implementation of Virtual Tour Guide App", *2022 International Conference on Advanced Computing Technologies and Applications (ICACTA)*, pp.1-6, 2022.