Proceedings of the 4th International Conference on Computing and Data Science (CONF-CDS 2022)
DOI: 10.54254/2755-2721/2/20220630

Memory Implementation on Navigation Systems

Qingyuan Yao[1]

[1]Shanghai Foreign Language School, Shanghai, China, 200003

jasonyao21@163.com

Abstract. Nowadays, cases in which people get lost under the guidance of navigation systems still occur. In order to render the situation, the essay seek for ways to improve the navigation system in ways other than enhancing navigational message expression. This paper mainly discusses the feasibility of memory implementation on navigation systems. By "memorizing" past records and past mistakes, the navigation system will be able to provide a revised route for drivers individually, reducing their possibility of making mistakes, and further increasing their system using experience. Meanwhile, with the support of big data, the navigation system could provide better routes for every user, using their own memories. The main method is to mark every "hotspot", that is, a spot where drivers are likely to make mistakes themselves, and provide them with alternative routes that cover as small number of hotspots as possible.

Keywords: system memory, navigation, machine learning, algorithm

1. Introduction

In December 2020, two Russian teenagers in Yakutsk got lost even under the premise that they were equipped with navigation systems. Having driven onto unfamiliar routes, the two teenagers lost direction. Afterwards, they unfortunately came across a technical failure with their cars. Trapped inside their cars under minus 30 degrees Celsius for a whole night, one of the two men died from hypothermia, and another barely lived.

Cases in which people get lost under the guidance of navigation systems are far from rare. As one come across confusing commands, one will easily go onto the wrong route or miss the critical point of decision. This causes loss of direction. Moreover, the drivers' attention consumed to discern the right route can cause accidents, leading to traffic jams and inefficiency of traffic systems.

Current researches have made significant progresses on how to strengthen UI expression of images. Since the birth of GPS navigation, the systems have adapted to numerous adjustments and improvements in order to enhance the delivery of navigational messages including voice delivery enhancements and image delivery enhancements. Yet despite these improvements about navigational message delivery already adapted, there are still cases like the one of the two lost Russian men. In order to further alleviate this situation and prevent accidents to happen, we ought to figure out another way to deduct the possibility of getting lost under navigation systems.

This paper aims to figure out how changes in route-determining algorithms can help prevent getting lost. Specifically, whether adjustments based on drivers' own driving patterns, driving habits and past routes taken can reduce the possibilities of drivers coming across scenarios difficult for them to handle. By trying out several possible methods of adjustments—adjusting the weight of different routes or

DOI: 10.54254/2755-2721/2/20220630

flyovers (edges and vertices in algorithm), and adapting them into route determination, namely—we could reach a conclusion about how these trials can effectively adjust the routes and change the possibility of getting lost under navigation systems.

2. Past researches

Many researchers have conducted researches and surveys about what an ideal navigation system shall contain, as well as how to realize the methods shown in the rest of the essay. Past researches are listed as follows:

In the paper "Enhancing the Route Optimization Using Hybrid MAF Optimization Algorithm for the Internet of Vehicle" [1], Dhanare et al. aims at enhancing the route optimization algorithm by taking a new algorithm into account, The paper "The Shortest Path Algorithm Performance Comparison in GRAPH and Relational Database on a Transportation Network" [2] has compared the performance of Dijkstra algorithm under different system frameworks, including PostgreSQL and Neo4j.

In this paper "Towards drivers' safety with multi-criteria car navigation systems" [3], Solé Leonardo proposes two frameworks to allow special navigation systems to come to effect under different weather conditions, including rain and fog.

Vörös Fanni et al. [4] took a survey in East Europe about what features the drivers wanted, and it turned out that drivers dislike UIs overwhelmed with features. Dandan Zhu and Sun Junqing in "The Path Optimization Algorithm of Car Navigation System considering Node Attributes under Time-Invariant Network" [5], have put special attributes of important nodes into consideration, they proposes a new algorithm, the ROLA algorithm, optimizing the usage of Dijkstra algorithm. "The Optimal Routing of Cars in the Car Navigation System by Taking the Combination of Divide and Conquer Method and Ant Colony Algorithm into Consideration" [6], this paper adopts divide and conquer method, as well as ant colony theorem, in their design of the Dijkstra algorithm. Kim et al. (2016) [7] designed their algorithms based on past samples and examples. By implementing big data, they were able to design an advanced algorithm. Hui Ma (2011) [8] used nodes to simplify maps, and used "divide and conquer" method as well in solving the route-optimization problem.

After referencing above paper, this paper chooses the Dijkstra algorithm as the final algorithm to implant the "system memory" into, because Dijkstra algorithm is an optimal solution for all system frameworks. In addition, further consideration about the balance between safety and efficiency in navigation system designing, namely, how to continuously cope with different situations is included in this paper.

3. Analysis about navigation systems and breakthrough points

3.1. The way systems attain best routes for drivers

Although there exists a broad spectrum of navigational algorithms currently, and their approach of obtaining a best route quite diverse, all the approaches to several best routes can be categorized into three steps: data input (attainment), data analysis and data output (results and visualization). Instead of changing the process of data analysis, which is time-and-effort-consuming, this method would make adjustments on data attainment, which is, by changing the weights or attributes of the edges and vertices, reach a different best route as a result.

DOI: 10.54254/2755-2721/2/20220630

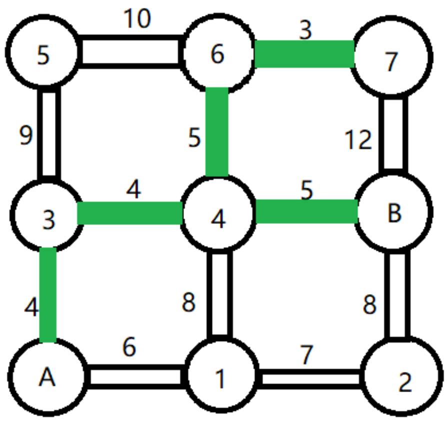

Figure 1. Map 1.1.

Take the map in Figure 1 as an example, in which green stands for highways (faster and frequently chosen routes). If we aim to travel from point A to point B, the best route is obviously the one from point A to point 3 to point 4 to point B. However, at point 4 exists a complex flyover, which would befuddle the drivers. Some may take the wrong route and go to point 6, and take the route A-3-4-6-7-B eventually. If so, we could try to adjust the weight of point 4, and change the route to go from point A to point 1 to point 2 to point B. This route costs drivers 21 minutes, 7 minutes less than the route A-3-4-6-7-B. It is noteworthy that such route determination steps need to take mathematical expectation of minutes into account. For instance, if a driver has a 80% chance to pass point 4 without a mistake, and reach the destination in 13 minutes, we wouldn't need to suggest the driver to go through the route A-1-2-B, since the mathematical expectation of minutes for route A-3-4-B will approximately be 16 minutes, less than 21 minutes for route A-1-2-B.

3.2. Breakthrough points

The main breakthrough points in solution of the problem would be specific edges and vertices with which drivers frequently make mistakes. Take the map in Figure 1 as an instance again. By implanting data collective functions inside the navigation apps, we can reach a conclusion how frequently drivers make mistakes when passing through point 4, an intricate flyover. Then, when considering routes passing through point 4, systems can add a specific amount of time needed, or multiply the time by a specific coefficient to reduce the competitivity of these routes. By implementing such methods, one can choose the best route, which takes both time and risks into account.

Meanwhile, for certain drivers who have taken the same route several times, changes on data input can also be made. For instance, if a driver have travelled on route A-3-4-B for several times, it is unlikely for him to take wrong routes when travelling on edges 3-4 and 4-B. The navigation system could collect data of the past routes, and reduce the weight of these edges afterwards. As a result, if the driver wants to go from point 3 to point 2, and there exists two routes of equal weight, route 3-4-B-2 and route 3-A-1-2, it would give the former route as a best route, since the driver is familiar with it, and the weight has been reduced.

4. Realization of the method

4.1. Data collecting functions

Implementations of data collecting functions are feasible. Each time a navigation route is adapted, the function would make records. Once the number of records exceeded a certain frequency, the system would start to make adjustments based on the past records. The function would also include deviation collecting functions, which collects past wrong routes taken or ignorance of the messages. All deviations

DOI: 10.54254/2755-2721/2/20220630

would reduce the possibility of taking a same route for a second time by adjusting the weight. As the navigation system is used more often, the system could tailor specific routes for every driver, both saving their time and improving their app-using experience. Not only would it benefit drivers, but it would also do good to the navigation system designers, since similar approaches taken could enhance users' stickiness to the system, and outperform other navigation apps when making future recommendations for routes.

4.2. Adjustments to weights

As data has been recorded, the next process, which features changes to weights of edges and vertices, take place. For every route taken over three times, the weight of the route will be multiplied by 0.9. If it would take four minutes to travel from one point to another, it would take 3.6 minutes in the navigation system's vision. The reduction of weight can be seen as an alternative to an increase of the value of an edge. In order to avoid the occasion in which this extra adjustment to weights hinder the consideration of traffic jams, the multiplication would not be taken into account when the weight of an edge increases itself, or, in other words, a traffic jam occurred on one edge.

The second adjustment would be alleviations to deviations. The method would be that for every time an adjustment to the route was made, meaning the driver's taking a route other than expected, the weight of the routes that pass through the edges will be multiplied by 1.5. On the one hand, if wrong routes taken at a specific vertex or edge has been taken more than once, the multiplication method would allow the weight to grow faster than linear addition. On the other hand, if on one route existed several points where drivers are likely to make a mistake, the weight would also grow faster than linear addition. This ensures the enhancement of mistake prevention, and further reduces the possibility of taking a deviated route.

4.3. Experiment and results

Adopting Dijkstra algorithm, a representative algorithm for generating a fastest route between any of two points on a map, some inspiring and encouraging results were shown. For the specific example of map 1.2, generated using a random number generator (weights of edges generated between 1 and 30 to mimic real scenarios, maps adjusted to be more realistic and avoid edge crossing together), results were like this:

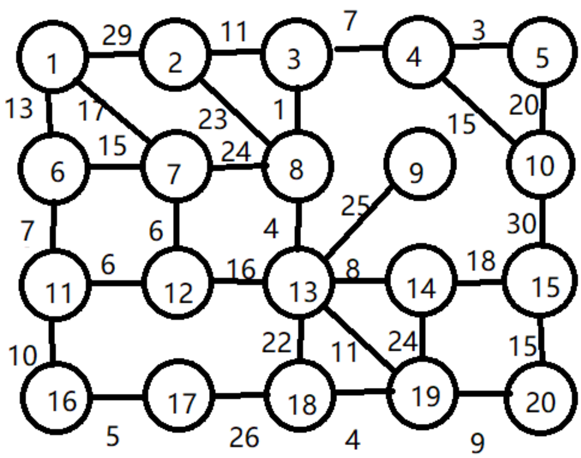

Figure 2. Map 1.2.

For this original map in Figure 2, the fastest route to go from the top left vertex (point 1) to the bottom left vertex (point 20) has a total weight of 59. The route is via point 1, 7, 12, 13, 19 and 20.

Supposing the driver lives near point 11, and all the routes near point 11 are more than recognized for them, the three edges of 6-11, 11-12 and 11-16 will get a multiplication of 0.7. Under this scenario, the best route from point 1 to point 20 will change to the route via point 1, 6, 11, 12, 13, 19 and 20. The total weight will be 58.1, outperforming the original best route. It is worth noting that although from the vision of the navigation system, the weight for edges 6-11 and 11-12 have been reduced, in reality they remain the same as 7 and 6 minutes, respectively. The total time spent on the new route will be 62 minutes, slightly longer than the original route.

If the driver takes the route 3-8-13-18 often, as generated, the weight of this route will be multiplied by 0.7. Under this scenario, the fastest route from top left to bottom right will still be the route via point 1, 7, 12, 13, 19 and 20. This proves the ability of making decisions between faster routes and more familiar routes. If an unfamiliar route is significantly faster, the difference of familiarity of routes would not cause a difference.

Supposing that the driver has made a deviation before. The driver drove onto edge 7-8 when he should have driven to edge 7-12 on the 1-7-12-13-19-20 route. The new best route would change for the driver's past mistakes. It would be the route via point 1, 6, 11, 12, 13, 19, 20. This would avoid possible mistakes made at point 7.

All the examples shown above has proven the effectiveness and accurateness of the algorithm.

5. Big data memory implementation and its benefit

Data obtained can not only be used in using experience improvement for a single user. Gathering up data can result in a large database, and is able to improve all users' using experience of the navigation system.

Take Figure as an example again. If numerous drivers have deviated from their routes at point 7, the online database would mark it as a hotspot on the map, which is, a vertex where mistakes are likely to be made. As a vertex is marked as hotspot, the routes passing through it would get a multiplication of minutes needed, just as the process mentioned in section 4.2. The coefficient this time, though, would be smaller than the one implemented on a single driver (1.5 in section 4.2), based on the proportion of deviations that occurred at the hotspot.

Meanwhile, the implementation of big data can also benefit people other than drivers. For example, a well-established database can provide city planners with a large amount of useful data, concentrating on the preference of the drivers and the hotspots.

For routes frequently took by drivers there is likely to be a large traffic, so city planners can look for ways to allay the heavy traffic, by widening up the road, providing alternatives for the route, etc.

For hotspots, on the other hand, city planners can categorize all the mistakes made at the hotspots, and figure out a clearly targeted solution. For instance, for forks in the road where drivers frequently make mistakes, the city planners can enhance the guidance of directions

6. Conclusion

Undoubtedly, this method is still a unfinished method, and needs further examination.

One unfinished factor of the method is the determination of the coefficients. The two coefficients mentioned in chapter 4.2, 0.7 for preferred routes and 1.5 for hotspots, respectively, are reached based on three analogous examples. In reality, the coefficients will have to stand more examinations and adjustments. Only two coefficients may not be effective enough to be adopted by the navigation systems, on the other hand, since conditions in reality are far more intricate than what two coefficients can manage to cope with. There would certainly be a more complete curve of the adjustments of weights. For instance, if there is one hotspot in the route, the total weight would be multiplied by 1.3. If there is two, though, it would be multiplied by 1.3*1.5, rather than simply the square of 1.3.

The determination the coefficients or the curves, meanwhile, have to rely on a big database obtained by the functions, and may involve deep learning methods. The moment deep learning systems figure out the way to fit the conditions in reality, the navigation system would reach its maximum effectiveness.

A limitation of the method is the indiscrimination between coefficients. For different hotspots, for instance, there ought to be different coefficients multiplied, based on the fraction of drivers who made mistakes and the average additional time they have spent on their new routes. The expectation for different drivers shall also be different. For instance, for experienced drivers, the multiplied coefficients

DOI: 10.54254/2755-2721/2/20220630

for hotspots can be lower, since experienced drivers are generally less likely to make mistakes. For new users or new drivers, on the other hand, the multiplied coefficients shall be higher. Since new drivers generally can't discern the signs as clearly as experienced drivers, no matter signs on the road or signs from the navigation system.

Based on memory implementation, a way is found to improve currently existing navigation algorithms, from the aspect of reducing drivers' risks of deviating from designated routes. The new algorithm used past data from one single system user to adjust routes made for the user in the future. It could also use data from all system users to improve the whole system in a larger view. This is an immature algorithm, and have lots of rooms for improvements in the future.

These are all rooms for improvement, and the author believes that once future work has been done, the memory implementation into the navigation system would become more successful, and is of value to navigation app designers.

7. Acknowledgements

I'd like to express my gratitude towards professor Venkatesan Guruswami for his efforts on this essay. Professor Guruswami has introduced us multiple algorithms, including the Dijkstra algorithm I have mentioned in this essay. I'd also like to thank teaching assistant Ms. Li Yan for her efforts in helping us have a better understanding on the algorithms.

References

[1] Dhanare, Ritesh,Nagwanshi, Kapil Kumar,Varma, Sunita. Enhancing the Route Optimization Using Hybrid MAF Optimization Algorithm for the Internet of Vehicle [J]. Wireless Personal Communications, (2022).

[2] Miler Mario, Medak Damir, Odobašić Dražen. The Shortest Path Algorithm Performance Comparison in GRAPH and Relational Database on a Transportation Network [J]. PROMET - Traffic&Transportation, 26(1), (2014).

[3] Solé Leonardo, Sammarco Matteo, Detyniecki Marcin, M. Campista Miguel Elias. Towards drivers' safety with multi-criteria car navigation systems [J]. Future Generation Computer Systems (2022).

[4] Vörös Fanni, Gartner Georg, Peterson Michael P., Kovács Béla. What Does the Ideal Built-In Car Navigation System Look Like?—An Investigation in the Central European Region [J]. Applied Sciences (2022).

[5] Dandan Zhu, Junqing Sun. The Path Optimization Algorithm of Car Navigation System considering Node Attributes under Time-Invariant Network [J]. MOBILE INFORMATION SYSTEMS (2021).

[6] Yousefi Pirayeh, Zamani Roghayeh. The Optimal Routing of Cars in the Car Navigation System by Taking the Combination of Divide and Conquer Method and Ant Colony Algorithm into Consideration [J]. International Journal of Machine Learning and Computing (2013).

[7] Kim, HyunMyung, Nam, Daisik, Cheon, Seung Hoon. Determination of Representative Path Set from Vehicle Trajectory Samples [J]. Journal of computing in civil engineering (2016).

[8] Hui Ma, Zhizheng Zhou , Guangyan Liu. Optimal Path Solution based on Dijkstra Algorithm [C]//. Proceedings of 2011 International Conference on Ecological Protection of Lakes-Wetlands-Watershed and Application of 3S Technology (EPLWW3S 2011 V3), pp.583-586 (2011).