EFFECTIVE ROUTE PLANNING IN ROAD NETWORKS USING MULTI CONSTRAINT ROUTING ALGORITHM

P.Jayasheelan
Assistant Professor
Department of Computer Sciences
Kristu jayanti College ,Banglore 560077
sheelan.jsp@gmail.com

Dr.F.Mary Magdalene Jane
Professor & Director
Department of Computer Applications
Dr.N.G.P Institute of Technology,Coimbatore 48
janejayakumar@gmail.com

***Abstract:* -** Today, applications that take the congestions caused by traffic jams into account already exist. During rush hour, it is common that multiple congestions occur on the road network. If the number of traffic jams is too big, the radio stations that transmit TMC messages to the end user often only report the longest traffic jams. Those applications use the Traffic Message Channel (TMC) technique to calculate a route around the congestion whenever the application receives a message about a traffic jam being present. The study for finding the optimal shortest path on graphs with nonnegative weights has taken many forms. The weights of a road network are usually either the length of the arcs or the time it takes to traverse the arc from one vertex to another. The latter depending on the length of the arc and the speed a vehicle is allowed to travel. The majority of route planning applications use the shortest path as a synonym for the *fastest* path.

The Multi Constraint algorithm uses three arrays to calculate the shortest path. The first one contains the neighbors of each node. The second array stores the number of neighbors each node has, and the third contains the link weights. In order for the algorithm to work, the road network has to have all these arrays available.

***Keywords:* - Road Network, Shortest Path, traffic, Message channel.**

1. INTRODUCTION

Computing best possible routes in road networks from a given source to a given target location is an everyday problem. Many people frequently deal with this question when planning trips with their cars. There are also many applications like logistic planning or traffic simulation that need to solve a huge number of such route queries.

Current commercial solutions usually are slow or inaccurate. The gathering of map data is already well advanced and the available road networks get very big, covering many millions of road junctions. Thus, on the one hand, using simple-minded approaches yields very slow query times. This can be either inconvenient for the client if he has to wait for the response or expensive for the service provider if he has to make a lot of computing power available.

On the other hand, using aggressive heuristics yields inaccurate results. For the client, this can mean a waste of time and money. For the service provider, the developing process becomes a difficult balancing act between speed and sub optimality of the computed routes. Due to these reasons, there is a considerable interest in the development of more *efficient* and *accurate* route planning techniques.

1.1. Multiple Constraints Routing Problem

Finding an optimal algorithm for specifically a graph of a *road network* is a popular topic. Especially since the Global Positioning Satellite (GPS) technology has been made publicly available, commercial companies seek to find such an algorithm to create a route planning application. A road network graph (or just road network from now on), however, is most of the times too large for Dijkstra's algorithm. A field of interest, aimed especially at improving Dijkstra's algorithm for a road network has emerged.

Most algorithms treat the road network as a static graph with predefined weights. The shortest paths the algorithms compute therefore only hold under the ideal circumstances. But what if we would try to calculate the shortest path on a dynamic road network? The distance weights remain fixed, but the traveling time weights may change (e.g. because of congestions on the road); alternative, shorter paths may exist. However, intuitively, one is unlikely to traverse around the traffic jam and drive additional kilometers in order to save only a couple of minutes. Therefore, constraints should be set to the number of kilometers the new shortest path may be. Present algorithms would calculate the shortest path under ideal circumstances and calculate it again with the altered travel time weights. Once both paths are calculated, one would find out if the new shortest path meets the set constraints.

The multiple constraints routing problem as follows. Consider a graph $G(N,E)$ where N denotes the set of nodes and E the set of links. Each link $u \rightarrow v$ from node u to node v is characterized by a m dimensional link weight vector $\vec{w}(u \rightarrow v) = [w_1(u \rightarrow v), w_2(u \rightarrow v),\ldots, w_m(u \rightarrow v)]$ where component $w_i > 0$ is a QoS (Quality of Service) measure such as delay, jitter, loss minimum bandwidth, cost, etc. In the case of our

978-1-5090-1936-6/16/$31.00 ©2016 IEEE

problem, the weights are delay and length with $m = 2$. The QoS routing algorithm calculates a path P that obeys multiple constraints, $w_i(P) \leq L_i$ for all $1 \leq i \leq m$.

Such a multiple constraints QoS routing algorithm is for Self-Adaptive Multiple Constraints Routing Algorithm and solves the *Multiple Constraints Optimal Problem* (*MC*(*O*)*P*). It uses the following four concepts; (1) nonlinear definition of the path length; (2) a *k*-shortest path approach; (3) non-dominance; and (4) look-ahead.

1) ***Nonlinear path length definition: -*** The shortest path based on this linear length does not necessarily meet all the constraints. However, the nonlinear path length definition $l(P) = \max[w_i(P)/L_i]$ for $1 \leq i \leq m$, does. If $l(P) \leq 1$, all the weights lie within the constraints and a solution to the MCP problem exists.
2) ***The k-shortest path approach: -*** The *k-shortest path algorithm* stores at node *i* up to *k* shortest paths from source node *s* to node *i*. The principle of non-dominance decides how many paths are actually stored at node *i* and can reduce the search space.
3) ***Non-dominance: -*** If two paths P_1 and P_2 exist and it holds that for all $1 \leq i \leq m$ weights, $w_i(P_1) < w_i(P_2)$, path P_2 is *dominated* by P_1. Any path that from source node *s* to destination node *t* that contains P_1 will be shorter than any path from *s* to *i* that uses P_2.
4) ***Look-ahead: -*** The look-ahead concept stores at node *n* for all $1 \leq i \leq m$ weights the shortest value from the destination to node *n*. This is done by executing Dijkstra's shortest path algorithm *m*-times for all *N*-1. This way, for each node *n*, an attainable lower bound $b_i(n)$ is computed. While computing the shortest path *P* from node *s* to node *t,* at each intermediate node *n*, the inequality $w_i(P_{s\to n}) + b_i(n) \leq L_i$ for all $1 \leq i \leq m$ weights should be satisfied for all constraints. This inequality check can reduce the number of paths in the search space of possible paths.

Today, applications that take the congestions caused by traffic jams into account already exist. Those applications use the Traffic Message Channel (TMC) technique to calculate a route around the congestion whenever the application receives a message about a traffic jam being present.

1.2. Existing Approaches & their results

The following are the classical approaches and it produces some of the route planning information's. It has some drawbacks.

Dijkstra's Algorithm: - It maintains an array of tentative distances for each node. The algorithm visits (or settles) the node of the road network in the order of their distance to the source node and maintains the invariant that the tentative distance is equal to the correct distance for visited nodes.

When a node u is visited, its outgoing edges (u, v) are relaxed: the tentative distance of v is set to the length of the path from s via u to v provided that this leads to an improvement. Dijkstra's algorithm can be stopped when the target node is visited..

Priority Queues: - The main focus of *theoretical* work on shortest paths has been how to reduce or avoid the overhead of priority queue operations.

The original version of Dijkstra's algorithm runs in O(n2). This bound has been improved several times. Experimental studies indicate that in *practice* even very simple priority queues like binary heaps only induce a factor 2–3 overhead compared to highly tuned ones.

Complete Distance Table. An extreme case would be to pre-compute all shortest paths. This allows constant time queries, but is prohibitive for large graphs due to space and time constraints. Still, it turns out that for some hierarchical approaches, this simple technique can be very useful when applied to the highest level of a hierarchy of networks.

2. PROPOSED APPROACH OF ROUTE PLANNING

The Self-Adaptive Multiple Constraints Routing Algorithm has an advantage over techniques such as TMC if it would use real-time traffic information. Situations may occur that the traffic flow on certain road sections does not reach the maximum speed but no traffic jam messages are generated.

TMC and other techniques are used to improve shortest path algorithms for a road network. The real-time traffic information needed for this project can be gathered using different methods.

2.1. Traffic Message Channel (TMC)

TMC stands for Traffic Message Channel and uses the Radio Data System (RDS)-technology to send traffic reports to the end user. RDS is a system, which sends information along with regular FM signals. This information can be anything from the name of the radio station the user is listening to, the frequency the station is broadcasting at or what music is broadcasted.

The RDS information is invisible for the user. A receiver such as a RDS-compatible car radio or navigation system receives the signal. Depending on the preferences of the user, the information is either ignored or provided to the user by the receiver. Further explanation of the RDS-technology is beyond the scope of this report.

TMC uses the RDS-technology for providing traffic information to the end user. Such information could be the presence of traffic jams, the road condition, the weather, (un-)scheduled road maintenance, etc.

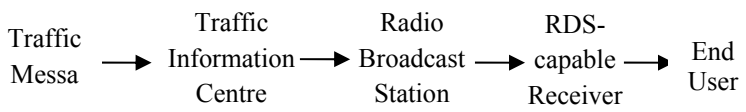

Figure Error! No text of specified style in document.**-1: Flow of a traffic message to the end user**

Traffic Information Centers (TICs) gather traffic messages via different information channels such as traffic monitoring systems emergency services. This information is used to create the TMC messages. These messages are then sent to the radio stations, which broadcast the messages as RDS-signals along with regular FM signals. An RDS-capable receiver receives the RDS-signal messages and depending on the information the message contains, alerts the end user.

2.2. Improving Shortest Path Algorithm

The sheer complexity and size of the road network may cause the calculation time of a shortest path algorithm to increase enough to be unsuitable for a navigation system[1]. There are, however some techniques that simplify the road network and/or speed up the calculation time. Some of those techniques are using the road network hierarchy, bi-directional search, directed search, using landmarks or reaches and shortcuts.

A road network consists of different types of roads. In order to travel from one place to a different, not neighboring city, one usually finds the nearest entry point to a highway and travels towards a point on a highway that is nearest to the destination. Without realizing, when calculating the route, one divides the road network into layers. A map also divides the road into hierarchies and displays them differently on a map. The higher the hierarchy, the thicker the road is drawn on a map. The most general types of hierarchies are:

1. Small road in the neighborhood
2. Main road in the city
3. Province road
4. Highway

Almost all commercial software for calculating the shortest path on a road network, use heuristics. Those heuristics assume that the lower hierarchy roads are only traversed when close to either the source or destination. Therefore, when calculating the shortest path, the algorithm will first perform a *local search* prior to switching to a *highway search*. The latter uses a highway network, which is smaller than the entire network (thus speed up the calculation). A shortest path algorithm based on highway hierarchies has been proposed in and improved in.

2.2.1. Bi-Directional Search

Shortest path algorithms normally search a path to another point in all directions, creating a circle as a search grid. This search method can be improved by starting a simultaneous search from both the origin and destination point until the two searches meet. This creates a bisection of the original search space. The bi-directional algorithm alternates searching the original graph from the source node, s, with searching the reverse graph from the destination, t, node while maintaining the shortest path, μ, found so far.

2.2.2. Directed Search

Even though a bi-directional search narrows down the search space, the Dijkstra's algorithm search in directions in which it is unlikely to find the shortest path. The directed search algorithms use lower bounds to determine which vertex is closer towards the destination than others are and prefer the nodes that are nearest to the destination. Suppose we are searching a path from s to t. The search works like the Dijkstra's algorithm except that it selects the vertex v with the smallest value of $k(v) = d_s(v) + \pi_t(v)$ (see [**Error! Reference source not found.**]), where $\pi_t : V \rightarrow \mathrm{R}$ where $\pi_t(v)$ gives an estimate on the distance from v to t.

2.2.3. Landmarks

A third technique is the use of *landmarks*. The landmark technique can be optimized by limiting the calculation of the s-t shortest path for a given s and t to the fixed-sized subset of landmarks that give the highest lower bounds on the s-t distance. This leads to more vertex scans but compared to the improved efficiency of the lower bound computation. Choosing the right amount and location of landmarks proves to be critical for the overall performance of the algorithm. The biggest advantage is that using landmarks saves calculation time, which speeds up the algorithm and results in a smaller search space.

2.3. Collecting Traffic Information

How this traffic information is gathered and distributed to the users when the information becomes publicly available. It is very tedious process.

2.3.1. Motorway Control & Signaling System (MCSS)

This system was designed to utilize the road capacity better, increase the safety on the road, collecting traffic information and help the road manager and emergency services.

The system consists of double loop detectors every 500 meters. At every double loop detector, a detector station (DS) is connected, which reads traffic measurements, prepares them and sends the data to the substation (SS).

A substation is connected to three detector stations. The SS receives the data from the detector stations and uses the data for Automatic Incident Detection. The SS is also connected with the central processor (CP) and can transmit data on request. The main function of the SS however is controlling the matrix signs above the road, which control the traffic flow by setting maximum speeds or closing down lanes and warning the road users.

The CP has two process computers (one as a backup in case of failure and can be used to store raw data). An operator uses the CP to access the data from the SS and act accordingly to that data.

2.3.2. Receiving Traffic Information

MCSS system stores traffic information such as vehicle classification, speed and numbers, as well as the rush hours, traffic jams and ghost riders.

Vehicle Classification: - The raw traffic data is gathered by a DS, connected to one or more loop detectors. A loop detector works by measuring the influence of a vehicle on the electro-magnetic field of the loop itself. If the influence reaches a predefined threshold, it is assumed a vehicle has entered the loop. The moment that same influence drops below a second threshold, the vehicle has left the loop. The second threshold has a different value than the first in order to counteract the rapidly changing of the digital signal if the analogue signal is changing around the threshold value.

Every type of vehicle produces a different analogue signal. This is caused by the fact that not all vehicles have the same density of material at every point. This produces a unique analogue signal 'footprint' for each vehicle type.

Vehicle Speed: - Calculating the speed of a vehicle requires two induction loops. Two induction loops located on a lane and the digital signal of the two loops in time.

When calculating the time it took to cover the distance between the two loops the speed of the vehicle can be measured:

$$v = \frac{S}{t_2 - t_1} = \frac{2.5}{t_2 - t_1} m/s \qquad \textbf{(Error! Reference source not found.}.1)$$

Once the vehicle speed is known, it can be used to locate a traffic jam or indicate during which hours it is rush hour. In addition, the speed can also be used to detect vehicles moving in the opposite direction.

3. DISTRIBUTING TRAFFIC INFORMATION TO USER

All the traffic flow speeds are known at all times, one might wonder how this can be accomplished in real life and if it is necessary to know *all* the traffic flow speeds. This information is not widely available for the following reasons:

- Not all roads are equipped with sensors to monitor the traffic speed.
- Traffic information is not publicly available.

When a route needs to be calculated, it is necessary to know all the traffic flow speeds on that route. Also the traffic flow speeds from the surrounding roads need to be know, otherwise Self-Adaptive Multiple Constraints Routing Algorithm cannot calculate an alternative route and correctly assume this route is faster/shorter. Knowing all the traffic flow speeds can be advantageous because Self-Adaptive Multiple Constraints Routing Algorithm can calculate every route with up-to-date traffic flow speeds, but this requires a lot of information to be sent. The traffic flow information can be limited to the region of interest.

Once it has been established which information is needed, that information should be delivered to the user. There are several techniques for doing this.

3.1. Internet

Nowadays, numerous car navigation systems are not fixed in the vehicle but are mobile in the form of a PDA, mobile phone or laptop.

The Internet can provide the navigation system with all up to date traffic flow speeds or, if limited memory is available, only the traffic flow speeds in the region(s) of interest

3.2. Mobile Connections

The traffic flow information that is required needs to be transmitted to the user. WAP has a data transfer speed of up to 9.6 Kbit/sec. If much traffic information is required (i.e. when the calculated route is long and traverses multiple highways), sending this information to a mobile navigation system with WAP tends to be too slow. The successor of WAP, GPRS, is roughly 5.5 times as fast as WAP, reaching transfer speeds up to 56 Kbit/sec, and more suitable for receiving the information. Especially while driving, time is not that essential. It may take more than just a couple seconds to receive traffic information for the route lying ahead. However, when planning a route at the beginning of the trip, time is more essential. The user is waiting for the navigation system to provide the calculated route. Therefore, every second that is saved during the information transfer, is time saved for the user.

3.3. Wi-Fi / WiMAXi

The wireless technologies Wi-Fi and WiMAX are excellent technologies for distributing information to mobile devices.

Both technologies rely on the fact that the mobile devices should be in range of a wireless access point. The range of the access point is limited, especially for the Wi-Fi technology. WiMAX has a range of up to fifty kilometers compared with the limited range of one to two kilometers Wi-Fi has.

There are two options for locating the access points. One is allowing the mobile device to be disconnected every now and then. While the device is connected, it can receive the traffic flow information. If a certain period has passed, the mobile device asks for updated traffic information once it is in reach of an access point.

The other option is making sure the mobile device is in reach of an access point the entire time. This requires access points to be distributed along the entire route. Especially when using the Wi-Fi technology, in which case the range is limited.

The Streetlight is just an example of a way to distribute wireless access points throughout a city. Navigation systems that enter the range of the Streetlight access point can

use its broadband internet connection to retrieve the traffic flow information.

4. PARAMETERS FOR TESTING THE PROPOSED SCHEME

When creating a test case in order to test the Self-Adaptive Multiple Constraints Routing Algorithm, several parameters may influence the outcome.

i) The length of the chosen route is also a big influence.

ii) The another influence is called traffic jam.

iii) A third influence depends on the psychology of the human being.

iv) A final influence is the traffic jam itself. Shorter traffic jams are easier to cope with than longer traffic jams.

To test Self-Adaptive Multiple Constraints Routing Algorithm on the road network, several tests can be done. First, it is helpful to test how often alternative routes are found that bypass a traffic jam. Second, Self-Adaptive Multiple Constraints Routing Algorithm needs to be compared with other shortest path algorithms. In addition, Self-Adaptive Multiple Constraints Routing Algorithm can optimize the calculated route for both the time and distance weight. A third test will verify if this optimization method will produce routes that are more efficient compared to optimizing for one weight only.

The following are the test cases implemented to Self-Adaptive Multiple Constraints Routing Algorithm

- Traffic jam speed.
- Route length
- Diversion length
- Traffic jam length
- Traffic jam location
- Road capacity

The first four tests are for comparison and do not have any traffic jams or constraints. First, all the routes are calculated with only one weight (time and distance respectively) which makes Self-Adaptive Multiple Constraints Routing Algorithm.

to act as a Dijkstra algorithm. During the next two tests, the routes are calculated with both weights but optimized first for distance respectively.

Before calculating the path, the program resets the traffic jams. This way, all tests have the same begin situation.

All the predefined traffic jam speeds are stored in an array called trafficjam, while the diversion constraints are stored in an array called diversion. An infinite diversion constraint is defined as -1. First, all the routes are calculated with one particular traffic jam and different diversion lengths. After all the routes are calculated with all different diversion lengths, everything is calculated again, with a different traffic jam speed.

After calculating a route, it is tested if the calculated route bypasses the traffic jam by testing the presence of the nodes of the traffic jam in the calculated route.

5. CONCLUSIONS

The development of car navigation systems has increased dramatically the last couple of years. The ability to calculate fast routes while avoiding any traffic jams that might be present does not only allow vehicles to reach their destination faster, traffic will also be more evenly distributed, leading in the decrease of traffic jams.

Shortest path algorithms can use the traffic intensity on the road network to calculate the fastest path. However, when constraints to the distance a person is willing to drive to bypass the traffic jam are set, Self-Adaptive Multiple Constraints Routing Algorithm has an advantage; even with tight constraints and a traffic flow of 70 km/h, the chance of finding an alternative route is almost 20%. This number only increases when the constraints loosen and the traffic flow speed decreases. When using relative constraints, the numbers increase even more.

However, the time saved when choosing an alternative route only becomes interesting when the traffic flow speed drops below 20 km/h. A traffic flow speed of less than 50 km/h generates a Traffic Message Channel (TMC) message. Navigation systems that use this technology therefore may also divert the user around the traffic jam. Taking all traffic flow information into account, even when it is slightly less than the optimal speed therefore proves in this case not to be beneficial.

6. REFERENCES

[1] 9th DIMACS Implementation Challenge. Shortest Paths. http://www.dis.uniroma1.it/~challenge9/, 2006.

[2] R. K. Ahuja, K. Mehlhorn, J. B. Orlin, and R. E. Tarjan. Faster algorithms for the shortest path problem. *Journal of the ACM*, 37(2):213–223, 1990.

[3] H. Bast, S. Funke, and D. Matijevic. TRANSIT—ultrafast shortestpath queries with linear-time preprocessing. In *9th DIMACS Implementation Challenge [1]*, 2006.

[4] H. Bast, S. Funke, D. Matijevic, P. Sanders, and D. Schultes. In transit to constant time shortest-path queries in road networks. In *Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 46–59, 2007.

[5] H. Bast, S. Funke, P. Sanders, and D. Schultes. Fast routing in road networks with transit nodes. *Science*, 316(5824):566, 2007.

[6] R. Bauer. Dynamic speed-up techniques for Dijkstra's algorithm. Diploma Thesis, Universität Karlsruhe (TH), 2006.

[7] R. Bauer and D. Delling. SHARC: Fast and robust unidirectional routing. In *Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2008. To appear.

[8] R. Bauer, D. Delling, and D. Wagner. Experimental Study on Speed-Up Techniques for Timetable Information Systems. In *7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'07)*. Schloss Dagstuhl, Germany, 2007.

[9] T. Bingmann. Visualisierung sehr großer Graphen. Student Research Project, Universität Karlsruhe (TH), 2006.

[10] U. Brandes, F. Schulz, D. Wagner, and T. Willhalm. Travel planning with self-made maps. In *Workshop on Algorithm Engineering and Experiments (ALENEX)*, volume 2153 of *LNCS*, pages 132–144. Springer, 2001.

[11] U. Brandes, F. Schulz, D. Wagner, and T. Willhalm. Generating node coordinates for shortest-path computations in transportation networks. *ACM Journal of Experimental Algorithmics*, 9(1.1), 2004.

[12] F. Bruera, S. Cicerone, G. D'Angelo, G. Di Stefano, and D. Frigioni. Maintenance of multi-level overlay graphs for timetable queries. In *7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'07)*. Schloss Dagstuhl, Germany, 2007.

[13] B. V. Cherkassky, A. V. Goldberg, and T. Radzik. Shortest path algorithms: Theory and experimental evaluation. *Math. Programming*, 73:129–174, 1996.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

[15] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1962.

[16] D. Delling, M. Holzer, K. Müller, F. Schulz, and D. Wagner. Highperformance multi-level graphs. In *9th DIMACS Implementation Challenge [1]*, 2006.