ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Competent Cache Supported Route Planning on Road Maps

Lakshman Tunk
Student,
Aurora Scientific Technological and Research Academy.

Ch Srivalli
Associate Professor,
Aurora Scientific Technological and Research Academy.

Abstract:

Pathfinding or pathing is the plotting, by a computer application, of the shortest route between two points. A journey planner, trip planner, or route planner is a specialised application used to find an optimal means of travelling between two or more given locations, sometimes using more than one transport mode. Typically route planning uses an efficient in-memory representation of the network and timetable to allow the rapid searching of a large number of paths. Database queries may also be used where the number of nodes needed to compute a journey is small, and to access ancillary information relating to the journey. A single engine may contain the entire transport network, and its schedules.

Path planning is a basic operation of route driving or flying a vehicle to somewhere/figuring out how to get somewhere) services. It finds out route between our needed/demanded starting place and ending place. Now there are number of computer programs at present like GPS and digital mapping. But due to unexpected difference/different version in driving direction, losing of GPS signal like many issues we need to propose this path planning method. We implement a system, cash based path planning method. It respond to us with respect our question, also it return result that (before that/before now) asked that stored in our computer file full of information. In this paper, we deliver a innovative framework for reusing the formerly cached query results as well as an effective algorithm for improving the query evaluation on the server.

Keywords:
Path Planning, Cache, GPS, Routhe Planning, Roads, Data analytics.

Introduction:

Pathfinding is closely related to the shortest path problem, within graph theory, which examines how to identify the path that best meets some criteria (shortest, cheapest, fastest, etc) between two points in a large network. At its core, a pathfinding method searches a graph by starting at one vertex and exploring adjacent nodes until the destination node is reached, generally with the intent of finding the cheapest route. Although graph searching methods such as a breadth-first search would find a route if given enough time, other methods, which "explore" the graph, would tend to reach the destination sooner. An analogy would be a person walking across a room; rather than examining every possible route in advance, the person would generally walk in the direction of the destination and only deviate from the path to avoid an obstruction, and make deviations as minor as possible. There is a great deal of variation among different journey planner applications, yet many share several common features. These typically include an interface for helping the user identify their origin and destination locations. These may include a geocoder which can find locations from street addresses or a web map that users can click on. A location finding process will typically first resolve the origin and destination into the nearest known nodes on the transport network in order to compute a journey plan over its data set of known journeys.

Related Work:

The first digital public transport journey planner software was developed by Eduard Tulp, ainformatica student at the University of Amsterdam on an Atari PC. He was hired by the Dutch Railways to build a digital journey planner for train services. In 1990 the first digital journey planner for the Dutch Railways (on diskette) was sold to be installed on PCs and computers for off-line consultation. The principle of the software program was published in a Dutch university paper in 1991This was soon expanded to include all public transport in the Netherlands. Other European countries soon followed with their own journey planners. The software supported telephone inquiries to public transport companies. Before, experienced staff was needed with good geographical knowledge and experience in consulting the paper timetables.

Early journey planning engines were typically developed as part of the booking systems for high value transport such as air and rail, using mainframe databases and OLTP systems. Well known examples of such computer reservations system (CRS) include Sabre, Amadeus, Galileo, and the Rail Journey Information System developed by British Rail. As computing resources became more widely available, journey planner engines were developed to run on minicomputers, personal computers, and mobile devices, and as internet based services accessible though web browsers, Mobile browsers, SMS, etc.

In the early 2000s large scale metropolitan web planners such as Transport for London's journey planner became available. Starting in 2000 the Traveline service provided all parts of the UK with multi-modal journey planning and in 2003 the Transport Direct portal was one of the first nationwide systems, allowing comparison of travel by any mode between any two points in the country. Many entities, including municipal government, state and federal government, and for-profit companies now operate web sites offering trip planning services for large metropolitan areas, or even whole countries. Transport companies such as EasyJet, National Rail Enquiries or Deutsche Bahn typically operate sites free to people planning trips, relying on ticket sales and advertising for revenues. As the size of the transport systems covered by journey planners has increased, protocols and algorithms for distributed journey planning have been developed, allowing the distributed computation of journeys using networks of journey planners, each computing parts of the journey for different parts of the country. JourneyWeb, EU Spirit, Xephos, and the Delfi Protocol are all examples of distributed journey planning protocols. Another development in the 2000s has been the addition of real-time information to update the current schedules to include any delays or changes that will affect the journey plan.

Technology:

Typically journey planners use an efficient in-memory representation of the network and timetable to allow the rapid searching of a large number of paths. Database queries may also be used where the number of nodes needed to compute a journey is small, and to access ancillary information relating to the journey. A single engine may contain the entire transport network, and its schedules, or may allow the distributed computation of journeys using a distributed journey planning protocol such as JourneyWeb or Delfi Protocol. A journey planning engine may be accessed by different front ends, using a software protocol or application program interface specialised for journey queries, to provide a user interface on different types of device. The development of journey planning engines has gone hand in hand with the development of data standards for representing the stops, routes and timetables of the network, such as TransXChange, NaPTAN, Transmodel or GTFS that ensure that these fit together. Journey planning algorithms are a classic example of problems in the field of Computational complexity theory. Real-world implementations involve a tradeoff of computational resources between accuracy, completeness of the answer, and the time required for calculation.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The sub-problem of route planning is an easier problem to solve as it generally involves less data and fewer constraints. However, with the development of "road timetables", associating different journey times for road links at different times of day, time of travel is increasingly relevant for route planners as well.

Algorithms:

Journey planners use a routing algorithm to search a graph representing the transport network. In the simplest case where routing is independent of time, the graph uses (directed) edges to represent street/path segments and nodes to represent intersections. Routing on such a graph can be accomplished effectively using any of a number of routing algorithms such as Dijkstra's, A*, Floyd-Warshall, or Johnson's algorithm.[11] Different weightings such as distance, cost or accessibility may be associated with each edge, and sometimes with nodes (e.g. where there are traffic signals).When time-dependent features such as public transit are included, there are several proposed ways of representing the transport network as a graph and different algorithms may be used such as RAPTOR.

EXISTING SYSTEM:

• Path planning needs to be delivered in a timely fashion. The requirement of timeliness is even more challenging when an overwhelming number of path planning queries is submitted to the server, e.g., during peak hours. As the response time is critical to user satisfaction with personal navigation services, it is a mandate for the server to efficiently handle the heavy workload of path planning requests.

• Jung and Pramanik propose the HiTi graph model to structure a large road network model. HiTi aims to reduce the search space for the shortest path computation. While HiTi achieves high performance on road weight updates and reduces storage overheads, it incurs higher computation costs when computing the shortest paths than the HEPV and the Hub Indexing methods.

• To compute time-dependent fast paths, Demiryurek et al. propose the B-TDFP algorithm by leveraging backward searches to reduce the search space. It adopts an area-level partition scheme which utilizes a road hierarchy to balance each area.

DISADVANTAGES OF EXISTING SYSTEM:

• A cached query is returned only when it matches completely with a new query.

• The time complexity is high.

• The cache content may not be up to date to respond to recent trends in issued queries.

• The cost of constructing a cache is high, since the system must calculate the benefit values for all sub-paths in a full-path of query results.

PROPOSED SYSTEM:

• To meet existing need, we propose a system, namely, Path Planning by Caching (PPC), that aims to answer a new path planning query efficiently by caching and reusing historically queried paths (queried-paths in short).

• The proposed system consists of three main components: (i) PPattern Detection, (ii) Shortest Path Estimation, and (iii) Cache Management.

• Given a path planning query, which contains a source location and a destination location, PPC firstly determines and retrieves a number of historical paths in cache, called PPatterns, that may match this new query with high probability.

• The idea of PPatterns is based on an observation that similar starting and destination nodes of two queries may result in similar shortest paths (known as the path coherence property).

• In the component PPatern Detection, we propose a novel probabilistic model to estimate the likelihood for a cached queried-path to be useful for answering the new query by exploring their geospatial characteristics.

• To facilitate quick detection of PPatterns, instead of exhaustively scanning all the queried paths in cache, we design a grid-based index for the PPattern Detection module. Based on these detected PPatterns, the Shortest Path Estimation module constructs candidate paths for the new query and chooses the best (shortest) one.

• In this component, if a PPattern perfectly matches the query, we immediately return it to the user; otherwise, the server is asked to compute the unmatched path segments between the PPattern and the query. Because the unmatched segments are usually only a smaller part of the original query, the server only processes a “smaller subquery”, with a reduced workload.

• Once we return the estimated path to the user, the Cache Management module is triggered to determine which queried-paths in cache should be evicted if the cache is full. An important part of this module is a new cache replacement policy which takes into account the unique characteristics of road networks.

• In this paper, we provide a new framework for reusing the previously cached query results as well as an effective algorithm for improving the query evaluation on the server.

ADVANTAGES OF PROPOSED SYSTEM:

• PPC leverages partially matched queried-paths in cache to answer part(s) of the new query. As a result, the server only needs to compute the unmatched path segments, thus significantly reducing the overall system workload.

• We propose an innovative system, namely, path planning by caching, to efficiently answer a new path planning query by using cached paths to avoid undergoing a time-consuming shortest path computation.

• On average, we save up to 32 percent of time in comparison with a conventional path planning system (without using cache).

• We introduce the notion of PPattern, i.e., a cached path which shares segments with other paths. PPC supports partial hits between PPatterns and a new query. Our experiments indicate that partial hits constitute up to 92.14 percent of all cache hits on average.

• A novel probabilistic model is proposed to detect the cached paths that are of high probability to be a PPattern for the new query based on the coherency property of the road networks. Our experiments indicate that these PPatterns save retrieval of path nodes by 31.69 percent on average, representing a 10-fold improvement over the 3.04 percent saving achieved by a complete hit.

IMPLEMENTATION:

When implementing our paper, itincludes mainly four modules. That issystem construction Module, Probabilisticmodel for PPatterndetection,an efficient grid-based solution and cache construction and update.

A. System Construction Module:

In system construction module, we develop the system with the required entities to implement our proposed model and evaluate the effectiveness of the system. The main goal in this work is to reduce the server workload by leveraging the queried-paths in cache to answer a new path planning query. An intuitive solution is to check whether there exists a cached queried-path perfectly matching the new query. Here, a perfect match means that the source and destination nodes of the new query are the same as that of a queried-path in cache.

B. Probabilistic model for PPattern detection:

To detect the best PPatterns, an idea is to calculate the estimation distance based on each cached path, and select thecached path with the shortest distance. It faces many challenges.

Firstly, the distance estimation requires the server to compute the unshared segments. Therefore, it incurs significant computation to exhaustively examine all cached paths. Secondly, such an exhaustive operation implicitly assumes that each cached path is a PPattern candidate to the query.

Algorithm of PPattern detection:

STEP 1: If distance between nodes s(source) and t(destination) less than threshold(Dl) value, return the result
PPattern = NULL

STEP 2: Divide the target space by grid cell size

STEP 3: Find out source(gs) and destination(gt) grid

STEP 4: Assign Qs = Logged queries whose paths pass start grid(gs)
AndQt = Logged queries whose paths pass destination grid(gt)

STEP 5: Store the intersect value of Qs and Qt
into Q

STEP 6: Assign path from source grid to destination grid for each query into PT

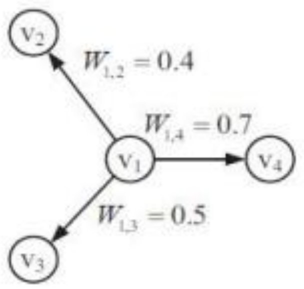
STEP 7: Return cached path(PT).

C. Efficient Grid Based Solution:

To retrieve these patterns, we invent a grid-based solution to further improve the system efficiency. Here divide the whole space into equally sized grid cells, here endpoints of all paths are mapped to the grid cells. By counting the total number of covered grids we will get the distance measure.

D. Cache Construction and Update:

It is an important part of cache management. Here we invent a cache replacement method by taking unique characteristics of road pattern. By observing many users, we found that certain routes are selected by most of the users. Most of them selectmain roads than branch roads. Because of the efficiency, popularity and capacity of these major roads. In a road network G = (V,E), each edge from node vmtovn is associated with a weight Wm, nit is a system computed value corresponding to the road type. If Wm, n is high indicates that the corresponding road type is high.



Example for edge and node weight

Fig. shows an example for node and edge weight to road types. Here edges are connected with node v1. The weights of three edges are W1,2= 0.4, W1,3= 0.5, and W1,4= 0.7, respectively. The weight of W1is set to W1,4= 0.7.Nodeweight represent show path planning query with it as the source node to be issued later. Information like this can be used to propose the cache replacement policy.

Algorithm of PPattern detection:

STEP 1: Assign PPatterns Detection into variable PT.

STEP 2: Assign Shortest Path Estimation from PTinto variable p.

STEP 3: if cache C is not full then insert p into cache and return C

STEP 4: Otherwise calculate usability for cached path and stored into {µ} and path with minimum usability and stores
It into p*

STEP 5: Check whether the usability value of the current path p is larger than the minimum usability value in thecurrent cache. If so, we place the current query into C. and return C.

CONCLUSION:

In this paper we implemented a cached path planning method. Also it solves all disadvantage of existing system. That is, our implemented system reduces the time complexity, Cached query is returned when it partially matches with a new query also, Cache content is up to date and also cost of constructing cache is low. Here server only needs to calculate unmatched path segments. So here workload of system is very law. That is, our system reduces the system latency almost 32%.

REFERENCES:

[1]Ying Zhang, Member, IEEE, Yu-Ling Hsueh, Member, IEEE, Wang-Chien Lee, Member, IEEE, and Yi-Hao Jhang, "Efficient Cache-Supported Path Planning on Roads", IEEE Transactions on Knowledge and Data Engineering, Vol. 28, No. 4, April 2016

2. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs", Numerische Mathematik, vol. 1, no. 1, pp. 269–271, 1959.

3. U. Zwick, "Exact and approximate distances in graphs – a survey", in Algorithms – ESA 2001, 2001, vol. 2161, pp. 33–48.

4. P.Sudha, S.Usha & G.Chamundi, Autonomous Robot Navigation, IJMETMR (ISSN 2348-4845), Vol 4, Issue 2, http://www.ijmetmr.com/olfebruary2017/PSudha-SUsha-GChamundi-82.pdf

5. S. Jung and S. Pramanik, "An Efficient Path Computation Modelfor Hierarchically Structured Topographical Road Maps", IEEE Transactionson Knowledge and Data Engineering, vol. 14, no. 5, pp.1029–1046, 2002.

6. P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Transactionson Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1967.

7. L. Zammit, M. Attard, and K. Scerri, "Bayesian Hierarchical Modelling of Traffic Flow - With Application to Malta's Road Network", in International IEEE Conference on Intelligent Transportation Systems, 2013, pp. 1376–1381.

8. S. Jung and S. Pramanik, "An Efficient Path Computation Modelfor Hierarchically Structured Topographical Road Maps", IEEE Transactions on Knowledge and Data Engineering, vol. 14, no. 5, pp. 1029–1046, 2002.

9. P.Charitha & S.Suresh, Finding the Shortest Path Computation in online , International Journal & Magazine of Engineering, Technology, Management and Research (IJMETMR), Volume No: 2 (2015), Issue No: 8 (August)
http://www.ijmetmr.com/olaugust2015/PCharitha-SSuresh-83.pdf

10. H. Mahmud, A. M. Amin, M. E. Ali, and T. Hashem, "Shared Execution of Path Queries on Road Networks", Clinical Orthopaedics and Related Research, vol. abs/1210.6746, 2012.