A Grid Based Design to Estimate Shortest Path for Road Network Using Path Planning By Caching

Santosh Patil¹, S Laxman Kumar², Jambula Hareesha³ and Mothe Rakesh⁴

¹²³ Assistant Professor, Department of Computer Science and Engineering, Institute Of Aeronautical Engineering, Jawaharlal Nehru Outer Ring Road, Dundigal, Hyderabad, Telangana 500043, India.

Abstract

Data mining is one of the orientations in today's research field. In road network navigation service migrating from the source to destination became a essential application model in a mobile device, using the global positioning system (GPS). The path planning service must be delivered appropriately. Novel path can be generated efficiently by caching and reiterating historical queried-paths, through Path Planning by Caching (PPC) in very least time. Therefore it extremely reduces the entire system workload, only need to compute unmatched segments. In this, we are using three main components or the algorithms which are used for the whole implementation of the process. The shortest path is detected for the new query from the cache by matching path patterns; if the new query is unmatched then it computes the query by using the grid-based design. Based on the patterns detected above, estimate the shortest path for a new query by matching queries. The matched query may be either complete hit or partial hit. The unmatched new query is to be updated in the cache for future use to get the path pattern in a timely fashion. As a result, the path planning by cache gives ten times more accurate than the existing work.

Keywords: global positioning system, Path Planning, Caching, shortest path, pattern.

INTRODUCTION

The advancement of the global positioning system (GPS) and ongoing popularity of mobile devices; which are utilized for the navigation application onto a mobile platform with an aid of the internet; As the latency to the user is sluggish so it is a mandatory for the server to expeditiously manage the labored workload. Novel path can be generated efficiently by caching and reiterating historical queried-paths, through Path Planning by Caching (PPC) in very least time.

In the Path planning caching, it matches the completely with a new query, but in PPC, it matches partially or completely with the queried path, which can be used to answer the path pattern of new query. The shortest path is detected for the new query from the cache by matching path patterns, if the new query is unmatched then it computes the query by using the grid-based design. In existing, the query has to be computed by approaching the server where it takes a long time. But in this proposed work, the query directly matches from the cache in shortest time compared to existing.

An ingenious system, named, path planning by caching (PPC), is efficiently return a novel path planning query by using cached paths to avoid undergoing a time-consuming shortest path computation. On average, we save up to 32% of the time in comparison with a conventional path planning system. The notion of PPattern is 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% of all cache hits on average.

An innovative probabilistic paragon is proposed, that detects the cached paths which have a high probability to be a PPattern for the novel query based on the consistency property of the road networks. Our experiments indicate that these PPatterns save retrieval of path nodes by 31.69% on average, representing a ten-fold improvement over the 3.04% saving achieved by a complete hit. The experimental results show that our advanced cache replacement policy increases the overall cache hit ratio by 25.02% over the state-of-the-art cache replacement policies.

The foremost work before the data mining is processed is to get the whole data which is to be worked out and get the accurate results from the data mining tool which we are using. For that, we have some of the steps to get the data which is to get the correct ideas about the proposed work and make the dataset by using java in our proposed work. The whole thing should be done in a much-costumed manner.

The main approach for designing this application begins with developing a mail-enabled podium for the small firm in which it is simple and commodious for sending and receiving message which consists of entertaining games, search engine and address book.

A significant outcome of the prior investigation is the decisive that the system request is feasible. This is can be achieved with limited resource and by taking less time. The proposed work surely reduces the time and energy. The proposed system is based on local area network with the company, which allows the work of the company to work form at any time; this proves that the proposed system is feasible. The
usage of the java for the creating of the dataset with the functionality of the algorithms which we are used gets the whole technically feasible as compared to the other technical stuff.

METHODS

"A Note on Two Problems in Connexion with Graphs, E. W. Dijkstra".[1] Here author proposed novel Algorithms for determining the shortest paths, which are more feasible than previous algorithm on networks which are relatively implausible. In arcs Known results which are the results of this paper surveyed and analyzed. A novel implementation for priority queues are developed, and a class of "arc set partition" algorithm is imported, for the single source problem on networks with nonnegative arcs, a running time of "O (m log (n + e))" is achieved.

"Real-Time City-Scale Taxi Ridesharing, Shuo Ma, Yu Zheng, et.al".[2] presented "shortest path algorithms" that use "A* search" in combination with a graph-theoretic lower-bounding approach that is based on landmarks and triangle inequality. The proposed algorithm is optimal one that finds the shortest path on any directed graph. Many algorithms such as A* ATL have been established to improve its performance by exploring geographical constraints as heuristics. Gutman proposes a reach-based approach for computing the shortest paths. An improved version adds shortcut arcs to reduce vertices from being visited and uses partial trees to reduce the preprocessing time. This work further combines the benefits of the reach-based and ATL approaches to reduce the number of vertex visits and the search space.

"Shared Execution of Path Queries on Road Networks, H. Mahmud, A. M. Amin," et al.[3] this paper proposes, a HiTi (Hierarchical MultiTi) graph model for construction of large topographical roadmaps to reduce the cost and to bust up the route computation. This graph model uses the hierarchical fashion for abstracting and constructing topographical road maps. Author proposes the another new algorithm named 'SPAHi', for determining the shortest path, the uses the HiTi graph model for topographical computing.

Whenever the cost of the graphs are updated, it is required to alter the graph, the costs of the edges in G(V, E). For example, roadmap is required to update, based on traffic condition. Thus, an efficient updating algorithm for a HiTi graph is essential for applications like road navigation.

"An Efficient Path Computation Model for Hierarchically Structured Topographical Road Maps, S. Jung and S. Pramanik." [5] In this proposed work author uses the number of bucket levels to determine the performance. Dijkstra's algorithm shows that a 2-level bucket data structure improves the performance. A 2-level bucket structure improves the time-bound and reduces the memory requirement. They confirm that the bucket level-1 implementation is not robust. This Level-1 bucket not used until and unless the depth of the network D is a less than number of nodes. The level-2 bucket is a robust choice for determining the shortest path problems.

"Exact and approximate distances in graphs—a survey, U. Zwick"[6] presented a algorithmic graph theory, the most basic and most studied problem is the problem of determining the shortest paths in graphs. Many algorithms were designed for finding the shortest path, even of these basic algorithms; the area of research is still open. The most algorithmic techniques used for finding the shortest path are considered as the combinatorial. However, there is some relation between the shortest path and matrix multiplication. Thus some of the algorithm is depends on matrix multiplication, which are considered mostly for the APSP.

"Bayesian Hierarchical Modeling of Traffic Flow, Zammit, M. Attard, et.al".[9] the author presented a novel method, Speed-up the process of finding the fastest path in timely fashion, the proposed model computes the shortest path online. Cooke and Halsey provided a method by studying TDFP, for solving dynamic programming in discrete time . A bidirectional A* search algorithm improves the computation time and increases the storage capacity.

PROPOSED METHOD:

PPattern is a historical path in a cache, which is determined by the path planning by caching (PPC) and this PPattern might match with new query with higher probability. The proposed work contains a new probabilistic model that matches the new query with the historical path, which can be used for answering new query. In the PPattern, if the query is perfectly matched the cache query then it is immediately returned to the user. Otherwise, the server is required to calculate the unmatched path segments. The next step is to find the shortest path by using the algorithm named as the shortest path estimation. Unmatched queries are computed in the PPattern are to be updated in the cache management server. At the end, we conducted contingency text on proposed work that the end results assures that the proposed system is not burden for the company. To conduct the feasibility study, it is required to know more about the system essential.

PPattern Detection:

The shortest path is detected for the new query from the cache by matching the path patterns; if the new query is unmatched then it computes the query by using a grid- based design as shown in figure 1. In this, the nodes which are said to be s and t namely called as source and destination are the main things for designing a grid-based design for the planning or tracking of the path patterns.
This way the sub-path retrieved from the cache is to be computed with the new query path. The new query path is to be sent to the cache server to check whether there is the same PPattern match is present or not. This is the whole way by which the PPattern detection is done and the PPattern is sent to the next level for shortest path estimation.

**Shortest Path Estimation:**

Based on the PPatterns detected above, estimate the shortest path for a new query by matching queries. The matched query may be either complete hit or partial hit. A complete hit occurs if there exists at least one queried-path in the cache with a source-destination node pair matching exactly with that of the input query. A partial hit occurs if there exists at least one queried-path in the cache detected as a PPattern of the input query which however is not a complete hit.

**Cache Management:**

The unmatched new query is to be updated in cache server for future use to get the path pattern in a timely fashion. With this the, whole process is developed and controlled for future use and also it is stored the past which is used by us now for getting the accurate and fast results in a timely fashion. The whole process of doing the proposed work is to get the fastest time than compared to the existing system. The proposed work has to calculate the only unmatched query logs which are not stored in the cache server. As shown in the figure 2, architecture of the proposed work explains everything about the process, which is going to do very much work on the basis of the algorithms that are the most important role in the whole process of the proposed work. In existing method a query path is returned when it matches completely with new query, but in Path Planning by caching it uses the partial or complete matched queried-path, are used to answer the path pattern of new query. Therefore it extremely reduces the entire system workload, only it need to compute the unmatched segments. In existing, the query has to be computed by approaching the server where it takes a long time. But in this proposed work, the query is directly matched from the cache in shortest time compared to existing.

**Figure 1:** Grid based design for matching the PPattern.

**Figure 2:** Architecture for advanced route planning in transportation of roads.
The shortest path is detected for the new query from the cache by matching path patterns; if the new query is unmatched then it computes the query by using a grid-based design. Based on the PPatterns detected above, estimate the shortest path for a new query by matching queries. The matched query may be either complete hit or partial hit. A complete hit occurs if there exists at least one queried-path in the cache with source-destination node pair matching exactly with that of the input query. A partial hit occurs if there exists at least one queried-path in the cache detected as a PPattern of the input query which however is not a complete hit. The unmatched new query is to be updated in the cache for future use to get the path pattern in a timely fashion.

Algorithms

The PPattern detection algorithm is done with which the shortest path is detected for the new query from the cache by matching path patterns; if the new query is unmatched then it computes the query by using a grid-based design. Based on the PPatterns detected above, estimate the shortest path detection is done for the second algorithm for a new query by matching queries. The last most algorithm is the cache management unmatched new query is to be updated in the cache for future use to get the path pattern in a timely fashion.

PPattern Detection Algorithm:
Input: Qs, t: a query; Dl: distance threshold; Dg: grid cell size, C: a cache.
Output: All candidates PPatterns PT.

1: if D(s, t) < Dl then
2: Return PT = ∅.
3: end if
4: Divide the target space by size Dg.
5: Determine the start grid gs and destination grid gt.
6: Qs ←Logged queries whose paths pass gs.
7: Qt ←Logged queries whose paths pass gt.
8: Q = Intersect(Qs, Qt).
9: PT ←(Sub)paths from Gs to Gt for each query in Q.
10: Return PT

Shortest Path Estimation Algorithm:
Input: The query source node s′ and destination node t′; all candidates PPatterns PT; Cache C.
Output: The estimated shortest path p∗.

1: if isEmpty(PT), then
2: p∗ ←Calculate path from server and return.
3: end if
4: Initialize Estimated Shortest Distance ESD = ∞.
5: For each path p ∈ PT do
6: if p is a complete hit then
7: Return p∗ s′, t′ = p.
8: end if
9: s∗ = argmin s ∈ Vp D(s′, s).
10: ds = D(s′, s∗).
11: remove s∗ from path node-set Vp.
12: t∗ = argmin t ∈ Vp D(t, t′).
13: dt = D(t∗, t′).
14: Let dr = |SDP(s′, t′)|.
15: d = ds + dr + dt.
16: if d < ESD then
17: ESD = d.
18: Update best PPattern p∗ = ps∗, t∗.
19: end if
20: end for
21: if s′ is not equal to vp∗ s then
22: SDP(s′, vp∗ s) ←Compute shortest path SDP(s′, vp∗ s).
23: end if
24: if t′ is not equal to vp∗ t then
25: SDP(vp∗ t, s′) ←Compute shortest path SDP(vp∗ t, s′).
26: end if
27: Return p∗ = SDP(s′, vp∗ s)⊙p∗⊙SDP(vp∗ t, t′).

Cache Management Algorithm:
Input: a query q, a cache C.
Output: a cache C.

1: PT ←PPatterns Detection.
2: p ←Shortest Path Estimation from PT.
3: if C is not full then
4: Insert p into C; Return C.
5: else
6: {µ}←Calculate usability for each cached path.
7: p∗ ←Path with the minimum usability.
8: if p∗.µ < p.µ then
9: C ←Replace p∗ with p
10: end if
11: end if
12: Return C.

Performance of proposed method:
The last stage of the proposed work is to get the good result to us without any errors for this purpose visualization of the output are the main and the important module of the proposed work. In this, the first of all, there are five attributes which are used in our work real-time dataset that is as follows in below diagrams are for the clear explanation for the class of attributes.
There are 1500 instances for each of the attributes than that of the most important attributes is the cache time and the ordinary time attributes in our proposed work.

The figure 3 is the overview of the whole dataset with the attribute called keyword in this the variation of the attribute is known very accurately.

Figure 3: Graph diagram for keyword.

The figure 4 is the overview of the whole dataset with the attribute called ordinary time in this the variation of the attribute is known very accurately.

Figure 4: Graph diagram for the ordinary time.

The figure 5 is the overview of the whole dataset with the attribute called cache time in this the variation of the attribute is known very accurately.

Figure 5: Graph diagram for the cache time.

The figure 5 is the overview of the whole dataset with the attribute called cache time in this the variation of the attribute is known very accurately.

Figure 6: Query log diagram for the dataset.

Figure 6 shows the log for the dataset which is formed from the application which is made for the preparation of the own dataset for the proposed work. This is the main base for the preparation of dataset with the actual values that compares with the whole dataset.

CONCLUSION

Novel path can be generated efficiently by caching and reiterating historical queried-paths, through Path Planning by Caching (PPC) in very least time. Therefore it extremely reduces the entire system workload, only need to compute the unmatched segments. The real road network database path planning PPC techniques by reducing 32% on average in comparison with the existing one. In this, there should we calculate the whole unmatched PPatterns and it is stored for future use as we store it in the cache.

REFERENCES


