A Four-Pronged Low Cost and Optimized Traffic Routing Solution

https://doi.org/10.3991/ijim.v14i10.15057

Mohammad Talha Qureshi (✉)
Indus University, Karachi, Pakistan
mohammad.talha229@gmail.com

Athaul Rai
Sindh Madressatul Islam University, Karachi, Pakistan

Noman Islam, Ghazala Shafi Sheikh
Iqra University, Karachi, Pakistan

Abstract—This paper proposes a novel four-pronged solution for estimating routes and total time required to travel in a transportation network. The proposed solution attempts to optimize the Google's Map API request flow using open street map (OSM). As it is known that each request to Google Map API will incur some latency, since different computation are performed on Google server for each request. To avoid this latency, route estimations (distance and time) are calculated using a four-pronged approach based on Google map API, open street map (OSM), routing cache and logical grid of locations. The objective is to create a generalized routing system that tries to use Google services in optimized fashion. The proposed approach stores each requests/ response from Google Map API into an optimized data structure called cache. After passage of time, these cache records are moved to another data structure called random access memory (RAM) file. In scenarios where requests can't be served via stored data, the proposed system attempts to give approximate estimations based on open street map (OSM) using an A* search for finding route. In addition to this, data that has been cached in proposed approach can be used for further analysis or applying machine learning algorithm for off-line route calculations later.

Keywords—Patraffic routing; geo-caching; off-line calculations; Google maps; A* algorithm.

1 Introduction

During past few years, information and communication technology (ICT) has been used for solving a wide range of daily life problems. This paper attempts to employ ICT technology to provide solutions for transportation problems. Most of applications in this domain are based on employing Google maps to obtain a real-time estimate of the time/ distance for travelling from a specific source to destination. However, the Google maps' services are limited by various means. First, it requires online connectivity and hence, it can't be used in offline mode. Secondly, the online request/ response will incur delay in obtaining a time/ distance estimate. Finally, the Google map services are based on pay-as-you-go model and require payment after some initial number of free requests / day (limited quota problem). The hypothesis of this paper is that one can use the Google maps in optimized way by employing a four-pronged solution and utilizes multiple secondary sources to facilitate end-user in achieving their objectives.

The objective of the paper is to facilitate geo-estimations intensive tasks or applications such Careem, Uber and similar type of food services. In such applications, several end-user's request for the distance and time estimations from source to destination and sometimes these requests have overlapping parameters. Hence, for each of request, often calculations are performed at server end repeatedly. For example, a person from location *A* request for the Google real time estimations such as route, distance between the points *A* and *B* along with the time required to travel. Google Map Application Programming Interface (API) will do these calculations for user at location *A* to location *B*. Now suppose some other user at Location *A* or near Location *A* make request for Google estimations for location *B* or near location *B*. Google Map API will repeat the calculations for this user as well. Considering this as redundant calculation and supposing that traffic behavior may not change much in last 10 minutes, if one can store previous calculations of user 1 and furnish these calculated values to user 2, one can save time and increase performance in term of response time. Now consider the scenario, where a lot of users are requesting for the routes and time. There is a high chance that most of the users' requests will have overlapping parameters. Suppose, there is Pizza shop that provides home delivery services to users from different locations (in most of the cases users will make a request from nearby locations and most of the users will be near to each other). We can save a lot of time by providing pre-compiled results to requests that are nearby. This will help to improve routing service of applications and hence will reduce networks load. The scenario has been shown in Figure 1.

Figure 1 show a location map in which there is an on-line pizza free home delivery system and users from different locations place their orders. The pizza shop system should provide the route, time and distance estimates. Suppose there are 16 users placing orders in any sequence and the system hits the Google MAP API 16 times to calculate routes. If average time for hitting Google Map API is *t* seconds, then total calculations will take 16*t* time. But, if one carefully observes scenario, the users in

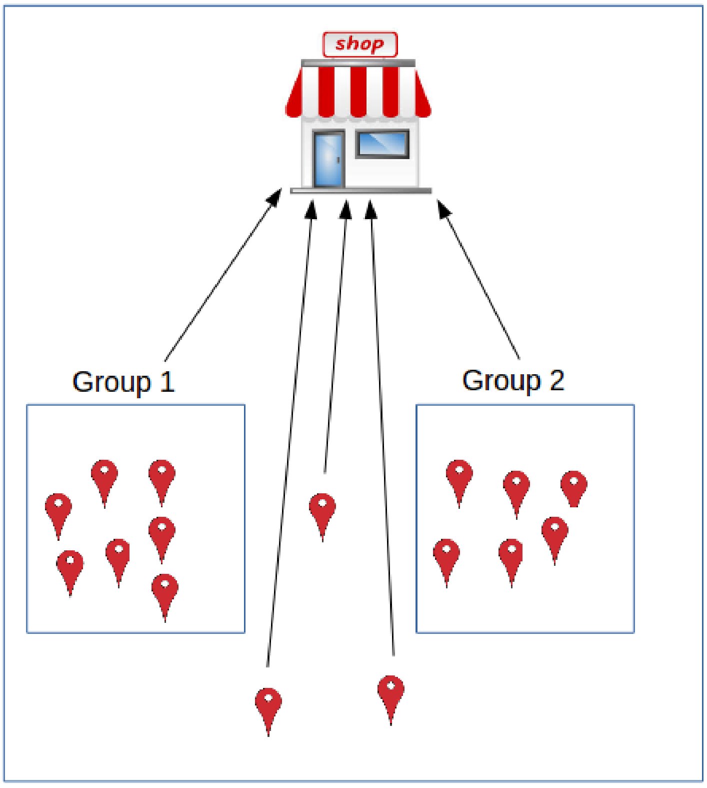


Fig. 1. A pizza shop providing home delivery services from nearby locations

some locations (as shown in figure by group 1 and group 2 labels) are much closer to each other. So, there should not be separate estimations and we don't need to hit Google Map API for users in Group 1. We can use the first requests estimations for all the users in group 1 and same is the case for group 2. So, according to proposed system, Google map API will be hit for one user in group 1 and same compiled results will be used for all the other users. This will take only *t* time. Same will be the case for users in group 2 taking *t* time and all other users (remaining 3 users) will be entertained separately. This is because they are far from group 1 and 2, and one cannot use compiled results for these users. So, the total time to facilitate all requests will be just 5*t* second which is very less than scenario discussed earlier. All the pre-compiled results will be stored locally in some very fast data structure. This paper uses dictionaries because they require constant time to search. The traffic behavior changes after some time depending upon the day and events. So, one cannot use these compiled estimations for all the future requests. Hence, the proposed approach removes these records after some time and moves these records to another file, for later use. This data will be use in future for applying machine learning algorithms and for off-line navigations and estimations up to great extent and make off-line estimations acceptable. In case of any issue such as when Google API key issue arises and maximum limit exceeded and Google Map API is not responding, the proposed approach uses open street map (OSM) to get the route and distance between two locations. The OSM map will not give real-time information, but one can approximate it using the vehicle type and speed using speed formula. Of course, this will not be a real time estimate, but will be very close to actual value. In case of OSM failure, the proposed approach uses old records with adding some traffic ratio. The last resort to respond to user request is Grid, when all the above approaches fail to respond. Grid is compiled information of locations. The proposed approach creates grid of locations and then store information like grid center, distance between all grids, time required to travel between grid, and these estimations will be very far from actual estimations. As one goes down in hierarchy, the accuracy of estimations decreases.

Rest of the paper discusses the propose approach. The next section provides related work. The proposed methodology is then discussed which is followed by results section. The paper concludes with providing future work and recommendations.

2 Related Work

There has been a plethora of literature cited for traffic routing problems. [1] has employed a weighted graph-based approach using open street map for blind pedestrian. A performance comparison of various traffic routing algorithms has been provided in [2]. In [3], the Open Street Map evolution in Germany for car navigations and street view has been discussed. In another work, a large-scale disaster management system has been discussed through large-scale simulation using Global Information System (GIS) data [4]. [5] have shown the OSM network as multi-attribute graph such as traffic signals, road types, points of interest etc. and the traversing of this graph has been shown. Routing algorithm in such multi attributes network has been discussed.

An optimization framework/ dynamic vehicle routing solution along with a backbone algorithm is proposed in [6] that allows dispatching of ride sharing taxis. The problem is formulated as an efficient network flow mixed-integer optimization problem. Through experiments, it was shown that the optimization solution has significantly better performance compared to greedy algorithms.

A dynamic routing and pricing model called extended Pickup and Delivery Problem (PDP) based on electricity price and passenger satisfaction problem has been proposed in [7]. The algorithm attempts to find an optimal route considering all the driver’s requirement along with reducing the total cost of operation. Concept of providing real time services has been shown in [8]. Authors have discussed the map services by Google, Bing and also show the use of large amount of spatial data of OSM. They have shown applications of their proposed system in web and over handheld devices. They have also shown that OSM providing us same shortest routes and information related to round time trip planning. But the only issue they are not dealing about the real time traffic behavior and time estimations.

Authors in [9] describe the comparison between accuracy of Open Street Map and Google Maps and Bing Maps. Five case studies cities and towns were selected for comparison. Comparison was performed under three different parameters spatial, currency and ground truth positional accuracies. The authors came up with conclusion that OSM had almost similar results in comparison with Google Map and Bing Map. Hence, OSM can be used as alternative for location-based services where there is access to high quality geo-spatial data is easy.

[14],[15] and [16] discussed the data caching process for off-line navigation purposes. This can be used for web as well mobile to make accessible world-wide. Concept of downloading off-line maps gives users feel of on-line navigations maps. The proposed system has been capable of giving routes, distance and estimates of cost required to destination by cab, bus or any transportation services. In [17], Google maps have been used to determine traffic levels in cities.

In several studies, Vehicular Ad hoc Network (VANET) has been used for real-time traffic routing purpose [20]. The basic idea is that a number of vehicles on the road collaborate in real-time to solve a wide range of safety and non-safety problems such as parking lot prediction, ramp metering, congestion avoidance, collision avoidance and gaming applications. In this direction, [19] provided a grid-based approach to traffic routing in vehicular ad hoc network. Messiah is an android application that informs about the traffic conditions in real-time and enable the driver to take proactive decisions [21]. An ant-colony based traffic aware routing protocol has been proposed in [22]. To reduce traffic in urban transportation, a traffic guidance model has been presented in [23]. The proposition is based on considering the collective attributes of the entities involved in a transportation system having different interests, goals and regulation. A novel eco-routing strategy has been proposed in [24] that proposes an energy efficient route considering accelerations and impact of road infrastructure. In [25], deep learning has been used to improve the performance of OSM. A walkability evaluation using OSM has been performed in [26]. A ship routing scheme has been proposed in [27]. The proposition considers the ocean currents and waves for the computation of optimal ship tracks. The system is based on estimating hazard and cost associated to known routes in Atlantic Ocean. The system analyzes sea state, hydrodynamics, and meteorological models. Finally, an Internet of Things (IoT) based traffic routing solution has been proposed in [28]. The objective is to cater to traffic jams. The solution is proposed for motorcycle riders.

Research gap: Based on the extensive literature review, it has been found that Google map API has several limitations. It can’t be used in offline mode. The online request/ response model incurs delay in obtaining a time/ distance estimate. Additionally, it is based on commercial model. The objective of this paper is to employ Google map API and open street map in an optimized manner to provide services to geo-intensive applications. To our knowledge, no such approach currently exists. Most of the approaches are based on employing Google API or offline maps. However, no such mechanisms exist for updating routes in offline mode.

3 Methodology

Figure 2 describes the block diagram of the essential components required for the working of proposed scheme. There is a user agent that accepts the requests from the end-user. An agent is an autonomous entity that can act on behalf of end-user to perform several actions. The requests from the end-user can be for a route between two destinations. The end-user will be interacting through a client application that can be a mobile device, a website or a desktop client. There are few data sources available that can be used to serve the end-user's request. For instance, the distance and time estimates can be furnished to end-user from the local cache if the similar request has been entertained sometimes earlier. Alternatively, the information can be fetched from Google maps. In case the requests can't be fetched as the total quota of the Google map requests have been over, then the requests can be entertained using Random Access Memory (RAM) file, open street map or using the grid structure. The cache maintains the currently hot data and then it is moved to RAM file.

Listing 1 shows the pseudo-code describing the working of proposed approach. The end-user request is checked in the cache first. If a record exists there, it implies that someone has already made similar request earlier (the threshold is set to 10 minutes in the experiments). So, the algorithm returns the same estimation saved in cache without posting a request to Google API. The proposed approach avoided the API call time; hence improved the performance in term of response time. The logic behind this is simply that it is most probable that the traffic pattern and other factors may not change in this specific time period. The records in cache after some specific time is moved to other repository called RAM files. RAM contains old records that are transferred from cache after some time *t*. This data is stored for different purposes such as finding patterns and applying machine learning algorithms to provide off-line navigation. This data is used to find estimations when Google API hits and cache failed to answer query. When the quota for free Google API hits completes, then requests will be served from RAM. In RAM, the records are not real-time estimate, but are quite acceptable as the traffic estimates don't change very quickly. If RAM is unable to answer query, then request estimations will be searched in open street map, which is open source community off-line map. This uses A* search algorithm for finding optimal path between source and destination. In case OSM is also unable to answer query, then it is checked in grid, which is based on the area/region divided into grid like structure. This paper stores different information about grid, like grid inter-distance and center of each grid. The proposed approach just picked the grid of source and grid of destination and then finds distance between two grid and return estimations. A number of data structures are used for storing the information in various data sources.

a) **Cache file:** For storing Google data such as estimated distance, estimated time and route information for off-line usage, we store the long textual and numeric information in dictionary i.e. Java Script Object Notation (JSON) format, keeping the memory and searching time constraints in mind.

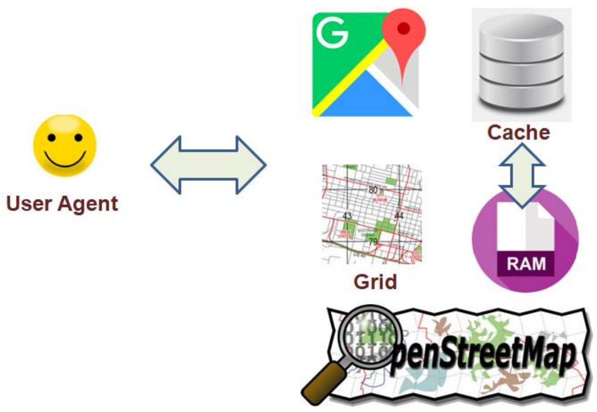
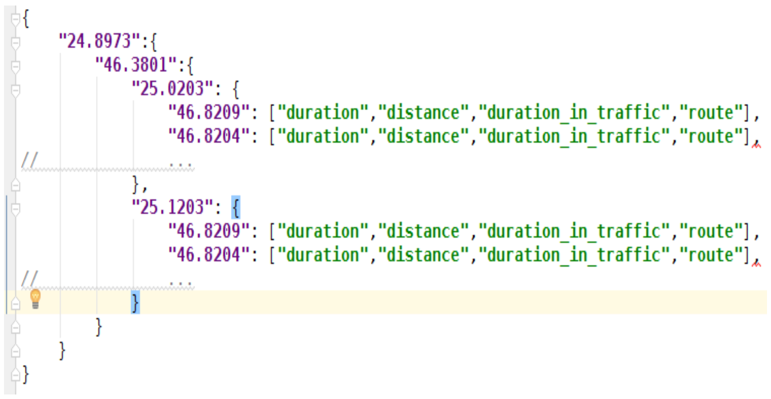


Fig. 2. Block diagram of proposed approach

The source and destination latitudes are used as key and all the information like distance, duration, and other routing information are stored as values. The latitude and longitude are stored up to 4 decimal places because the distance changes are less significant after 4 decimal places. The structure of dictionary (cache) has been shown Figure 3.



```
{
    "24.8973":{
        "46.3801":{
            "25.0203": {
                "46.8209": ["duration","distance","duration_in_traffic","route"],
                "46.8204": ["duration","distance","duration_in_traffic","route"],
//              ...
            },
            "25.1203": {
                "46.8209": ["duration","distance","duration_in_traffic","route"],
                "46.8204": ["duration","distance","duration_in_traffic","route"],
//              ...
            }
        }
    }
}
```

Fig. 3. Structure of Cache.json file

b) **RAM file:** RAM file is also similar to cache data structure, but the only difference is that it contains old records. The records that exist in cache for more than 10 minutes are transferred to RAM file structure. This data is maintained for finding patterns from data to apply machine learning algorithms for fast, optimized and real time navigation services in web-based systems or mobile applications.

Listing 1: Pseudo-code describing the working of proposed scheme

```
cache: a data structure (dictionary) used to store fresh records
ram_ file: data structure used to store old records
OSM: Offline map, Location information
grid: Location Information

check request in cache
if record in cache
    return estimations
else if you have Google Hits
    return estimation
    add records to cache
    move records after some specific time
else
    check record in RAM
    if record in RAM
        return estimations
    else if record in OSM
        return estimation
    else
        return estimation from Grid
```

c) **Open street map (OSM) file:** In the OSM layer, the information is saved into grid structure. The only problem with OSM is that we can find route between source and destination, but we cannot get time required to travel between source and destination. For this purpose, we have used the route points to get distance between source and destination by using Euclidean Formula as follows:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

To calculate the time required to travel from source to destination considering the vehicle type, following formula is applied:

$$t = s / v$$

As OSM is off-line version of Maps, so may be outdated, and will provide slightly old directions that are obsolete now. But these are rare cases as highways and motorways and trunks changes very rarely.

d) **Grid structure:** If the information can be obtained from cache, Google map or OSM, the last resort is to use a Grid like structure. In this strategy, the whole map is divided into grid. Information related to grid like distances between grids, center of grids, and the compiled distances between grid centers and the time required from one block to another is already stored. Upon request for a particular source and destination, source and destination blocks are identified and then compiled distance and time are returned.

4 Results and Discussions

The proposed methodology has been tested on different test cases. Figure 4 shows the grid created on Riyadh city. In the first experiment, the average Google map API hit time is calculated for 20 hits that were made one after another from source *A* (24.8973,46.3801) to destination *B* (24.1708, 47.2773). Table 1 shows the results. The Google map API took 1.77855806351 sec to process these requests. After these hits, the records were stored in cache and the request is made again but this time, they are served from cache which takes just 0.136595964432 seconds. It is concluded that the average time from cache is found to be very less time as compared to Google map API hit. In this scenario, the time and distance between source and destination were 106 minutes and 148 km respectively.

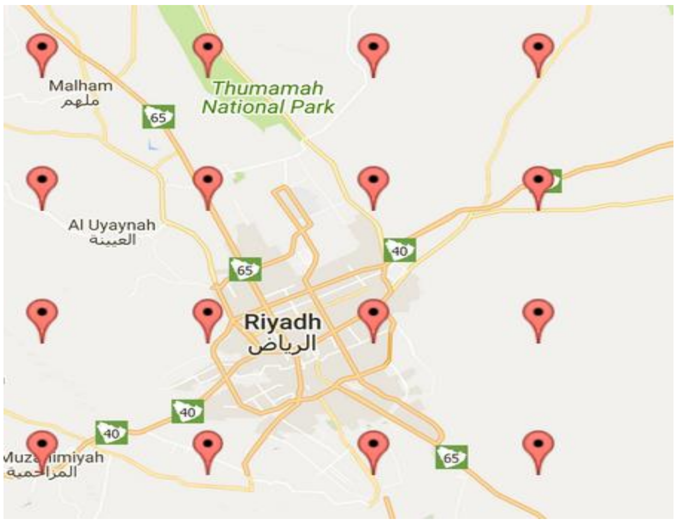


Fig. 4. Grid created on Riyadh city

Table 1. Time and distance comparison between proposed and Google map API

	Time using Google Map	**Time using proposed approach**	**Distance using Google map**	**Distance using proposed approach**
At t=0	106	106	148	148
At t=5 min	106	106	148	148
At t=10 min	113	106	148	148
At t=60 min	117	106	148	148

We continued this experiment further. After 5 minutes the statistics remains same using Google map API or using the cache. Both sources were providing similar answers. After 10 minutes the records become stale in cache, so they are transferred to RAM file. In the next step, the results were again compared to determine the effectiveness of proposed scheme. So, the paper compared statistics for the same source and destination from Google map API and the RAM. It is found that the traffic behavior changes in that time duration. Google now show time required as 113 minutes which shows the traffic increase on that route, due to which the Google API shown more time. The result from RAM file were however still acceptable with a small difference of 7 minutes; while distance remains same. After 1 hour, the estimated time changed for same source and destination, that was observed 118 minutes, which is 5 minutes more than second reading and 14 minutes more than first reading. We can identify this traffic pattern by reading different values at different times of day. These observations shown the effectiveness of cache and RAM file; hence the working of proposed methodology.

We analyzed the distance calculated by OSM and Google map. It is to be noted that the OSM returns route in form of nodes (latitude and longitude pair) as shown in Figure 5. Same request is calculated by using Google map API and that return the route shown in Figure 6. It can be observed that route returned in this case is slightly different. This problem arises due to the fact that open street map doesn't return the information in real-time. Hence, one needs to update the OSM map frequently. From map, it is cleared that OSM gives a short route. This may be due to the fact that path is temporarily not available or the path computed by OSM has become obsolete.

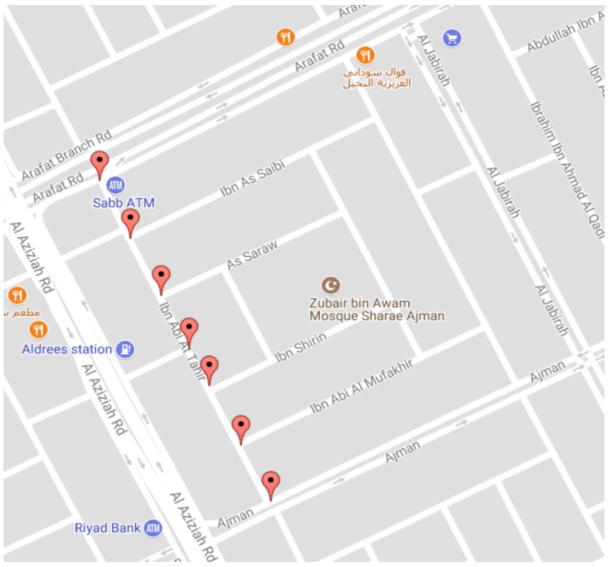


Fig. 5. Route between two points using OSM

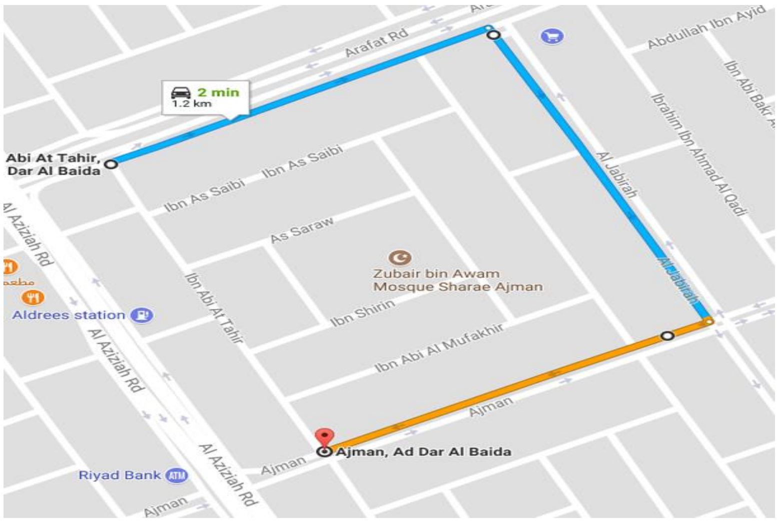


Fig. 6. Route between two points using Google MAP API

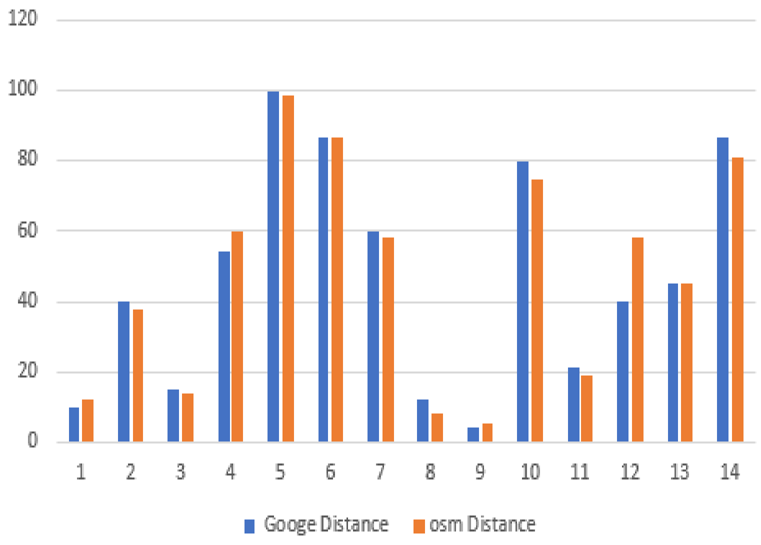


Fig. 7. Difference between OSM and Google MAP API distance

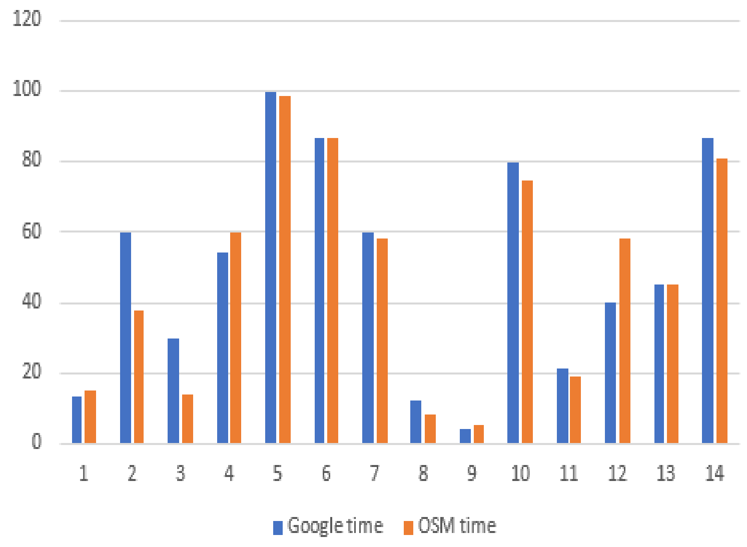


Fig. 8. Difference between OSM and Google MAP API time

Figure 7 and 8 shows the distance and time calculated between different sources and destinations using the proposed approach and Google map API. It is found out that these calculations don't differ substantially and hence, the proposed four-pronged solution can be reliably used.

Before we end this section, it is essential to mention the limitations of proposed approach. The only issue observed with the proposed approach was that open street map sometimes show outdated information in some cases. In real scenarios, the distance information doesn't change very frequently provided that the Google API doesn't remain inaccessible for months. The timing information can change based on the traffic situation that occurs at various times of the day; but that can be predicted. Hence, the proposed approach can be used for geo-calculation-based applications. But for mission critical applications, it can't be used reliably.

5 Conclusions and Future Work

Proposed system has shown good performance in term of response time as compared to Google map API in terms of routes estimations and calculation. This can save a lot of time in geo-calculation intensive applications as discussed earlier in paper. In these cases, we can reuse the estimations calculated at Google some time ago for the current request if the request is same or nearby. Most of the hits that will take time and may cause network congestions and load on server in case of millions of requests, can be avoided by using cached data. This cached data after sometime, can be transferred to others files (RAM file) for later use. We have applied linear regression model on Google real time data and our system cached data, and it works up to acceptable accuracy. This can be improved in future using different parameters such as traffic ratio, time etc. in some location. We can also do offline navigations using this data, in case of network failure, if this data is locally stored on device. Apart from this information, the proposed system does not deal with emergency issues that occurs during that 10 or 15 minutes where we take compiled information from cache.

6 Acknowledgement

Thanks to our team, who has worked with us to follow the vision and lead, also for them to bear our many requests of making different suggested updates in algorithm in pursuit of accuracy and performance. We are very thankful for their respect and hard-work. Also, thanks to the organization Cubix, Iqra University and Sindh Madrassatul Islam University, that gave us an opportunity to work on most pragmatic-level industry-grade problems in the field of our passion.

7 References

[1] Cohen, A. “Building a Weighted Graph based on OpenStreetMap Data for Routing Algorithms for Blind Pedestrians”, Master’s Thesis, Technion Israel Institute of Technology (2017)

[2] Hahmann, S., J. Miksch, B. Resch, J. Lauer & A. Zipf, “Routing through open spaces – a performance comparison of algorithms”, Geo-spatial Information Science (2017). https://doi.org/10.1080/10095020.2017.1399675

[3] Pascal, N., D. Zielstra, and A. Zipf. "The street network evolution of crowdsourced maps: OpenStreetMap in Germany 2007–2011."Future Internet 4, 1: 1-21 (2011). https://doi.org/10.3390/fi4010001

[4] Moritz, G. and C. Dornhege. "Realistic cities in simulated environments-an open street map to robocup rescue converter." In Fourth International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster (SRMED 2009) (2009)

[5] Graf, F., H. Kriegel, M. Renz, and M. Schubert. "Mario: Multi-attribute routing in open street map." In International Symposium on Spatial and Temporal Databases, pp. 486-490. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22922-0_36

[6] Bertsimas, D., P. Jaillet, S. Martin, “Online Vehicle Routing: The Edge of Optimization in Large-Scale Applications” (2018)

[7] Zhang, B., T. Chen, W. Su, “Optimal routing and charging of an Uber-like electric vehicle considering dynamic electricity price and passenger satisfaction”, in proceedings IEEE Conference and Expo of Transportation Electrification Asia-Pacific (ITEC Asia-Pacific) (2016). https://doi.org/10.1109/itec-ap.2016.7513013

[8] Dennis, L. and C. Vetter. "Real-time routing with OpenStreetMap data." In Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems, pp. 513-516(2011). https://doi.org/10.1145/2093973.2094062

[9] Błażej, C., R. Jacob, P. Mooney, and A. C. Winstanley. "Comparison of the accuracy of OpenStreetMap for Ireland with Google Maps and Bing Maps." In Proceedings of the Ninth International Symposium on Spatial Accuracy Assessment in Natural Resources and Enviromental Sciences 20-23rd July 2010, p. 337. University of Leicester (2010)

[10] Peter, M. and P. Corcoran. "Characteristics of heavily edited objects in OpenStreetMap. "Future Internet 4, 1: 285-305 (2012). https://doi.org/10.3390/fi4010285

[11] Dragomir, A. C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver. "Google Street View: Capturing the world at street level. "Computer 43,6: 32-38 (2010). https://doi.org/10.1109/mc.2010.170
[12] Ioannis K. N, K. P. Valavanis, N. C. Tsourveloudis, and A. N. Kostaras. "Evolutionary algorithm based offline/online path planner for UAV navigation. "IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 33, 6: 898-912 (2003). https://doi.org/10.1109/tsmcb.2002.804370
[13] Yogesh, G. and G. Dudek. "Offline navigation summaries." In Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 5769-5775. IEEE (2011). https://doi.org/10.1109/icra.2011.5980094
[14] Curt A. S., A. J. Fuller, M. P. Rhoten, J. Perez, and S. R. Drake. "Caching data for offline display and navigation of auxiliary information." U.S. Patent 7,577,771, issued August 18 (2009)
[15] Prithumit, D., N. Singh, S. Kumar, N. Rai, P. A. S. Naidu, and C. H. S. N. Iyengar. "Offline navigation system for mobile devices. "International Journal of Software Engineering & Application 1, 2 (2010).
[16] Joy D., and K. M. Grennan. "Integrating online navigation data with cached navigation data during active navigation." U.S. Patent 9,927,247, issued March 27 (2018)
[17] Pokorný, P. "Determining Traffic Levels in Cities Using Google Maps", in proceedings of Fourth International Conference on Mathematics and Computers in Sciences and in Industry (MCSI), Corfu, Greece (2017). https://doi.org/10.1109/mcsi.2017.33
[18] Open Street Map. Available Online: https://www.openstreetmap.org. Last checked on April 21, 2020
[19] Islam, N., Shaikh, Z. A., Talpur, S. "Towards a grid-based approach to traffic routing in VANET", in proceedings of E-INUDS (2008)
[20] Vahdat H., A. Ramazani, T. Mohammadi, W. Mansoor, "A Survey on Context-Aware Vehicular Network Applications" (2016). https://doi.org/10.1016/j.vehcom.2016.01.002
[21] Hadiwardoyo, S. A., S. Patra, C. T. Calafate, J. C. Manzoni, "An Intelligent Transportation System Application for Smartphones Based on Vehicle Position Advertising and Route Sharing in Vehicular Ad-Hoc Networks", Journal of Computer Science and Technology, 33(2), pp. 249-262. https://doi.org/10.1007/s11390-018-1817-4
[22] Goudarzi, F., H. Asgari, H. S. Al-Raweshidy, "Traffic-Aware VANET Routing for City Environments—A Protocol Based on Ant Colony Optimization", IEEE Systems Journal (2018). https://doi.org/10.1109/jsyst.2018.2806996
[23] Alvaro Paricio Garcia, Miguel A. L´opez-Carmona, "Multimap Routing for Road Traffic Management", in proceedings of International Conference on Practical Applications of Agents and Multi-Agent Systems, 2019, Spain. https://doi.org/10.1007/978-3-030-24209-1_16
[24] Giovanni De Nunzio, Laurent Thibault, Antonio Sciarretta, "A Model-Based Eco-Routing Strategy for Electric Vehicles in Large Urban Networks", Comprehensive Energy Management – Eco Routing & Velocity Profiles, pp. 81-99. https://doi.org/10.1007/978-3-319-53165-6_5
[25] Hampus Londögård, Hannah Lindblad, "Improving the OpenStreetMap Data Set using Deep Learning", Lund University, 2018.
[26] Austin Dunn, Bailey Hanson, Christopher J. Seeger, "Evaluating Walkability in the Age of Open Data: OpenStreetMap and Community-level Transportation Analysis", Journal of Digital Landscape Architecture, pp. 119-128 (2018).
[27] Nadia Pinardi, Matteo Scuro, Lorenzo Carelli, "Ship routing hazard maps", AtlantOS Deliverable, D8.10. AtlantOS (2019).

[28] Muhamad Asvial, M. Faridz Gita Pandoyo, Ajib Setyo Arifin, "Internet of Things Solution for Motorcycle Riders to Overcome Traffic Jam in Jakarta Using EBkSP", in proceedings of International Conference on Information and Communication Technology Convergence (ICTC), 17-18 October, 2018, Jeju, South Korea. https://doi.org/10.1109/ictc.2018.8539395

8 Authors

Mohammad Talha Qureshi is associated with Indus University, Karachi, Pakistan

Athaul Rai is currently research student at Sindh Madressatul Islam University, Karachi, Pakistan. Email: athaulrai@gmail.com

Noman Islam and **Ghazala Shafi Sheikh** are from Iqra University, Karachi, Pakistan. Emails: noman.islam@iuk.edu.pk & Ghazala.shafi@iuk.edu.pk. In addition, Noman Islam is also associated as post-doc research fellow at University of Kuala Lumpur, Malaysia

Article submitted 2020-04-20. Resubmitted 2020-05-18. Final acceptance 2020-05-20. Final version published as submitted by the authors.