Pick-me: A Carpooling Clustering System for Alternative Transportation

Kareem Mohamed, Amr Aziz, Belal Mohamed, Khaled Abdel-Hakeem, Mostafa Mostafa
Faculty of Computer and Information
Helwan University, Egypt
fcih.kareem@fci.helwan.edu.eg

Ayman Atia

Faculty of Computer Science
October University for Modern Sciences and Arts
Faculty of Computers and Information
HCI-LAB, Helwan University, Egypt
ayman@fci.helwan.edu.eg

Abstract—Many travel systems were built to facilitate user travel experience while these systems flourish in some cases, it fails in other cases as explained in [5]. **Pick-me** is a clustering system that provides an alternative travel method on a regular basis and cheaper than the previous solutions. The system aims to reduce traffic and fuel consumption by minimizing the number of vehicles and maximizing the number of passengers per vehicle. The system introduces a new algorithm Dynamic Distance Minimizer, the algorithm chooses the points of interest through a given route and decides based on learning patterns the best route. The system has proven its efficiency by testing it on 16 routes with a Recommendation accuracy of 98%.

I. INTRODUCTION

Overcrowded roads and high fuel consumption are one of the core issues facing the world. The number of driven cars on roads exceeds 1.1 billion cars and expected to be increased to 2 billion cars by the year 2040 [1]. Many customers travel to work daily on a regular basis, many of them don't share their cars because they are not aware enough by whom share the same common routes. Using AI, Pick-me focuses on reducing fuel consumption and overcrowded roads by using multiple algorithms allowing it to cluster the largest number of users with the same interests and sharing common key-points on their destination. Finding a common path from users riding traffic daily and then cluster similar groups according to common features was never introduced to the best of our knowledge. Moreover, using alternative systems including private taxi or similar systems are money consuming so it can't be used on regular basis, on another hand our system focuses on using it on regular basis like going to work every day
or going to school with the largest number of users to minimize the cost, and since the working days differ from a worker to another the system should be able to learn the patterns and provide alternatives. In order to achieve that, we implemented many algorithms to facilitate the clustering process and we were able to reduce the complexity to O(n). Many systems were implemented to improve the user travel experience but these systems lack in some core features like clustering, regular basis and fully autonomous, which was proven by our result to be core features required

for maximizing user benefits and satisfaction.

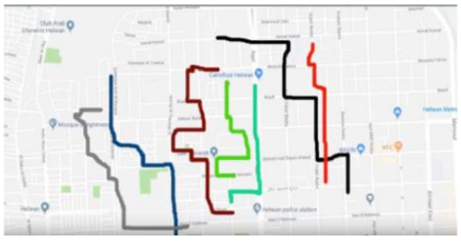
II. BACKGROUND RESEARCH AND RELATED WORK

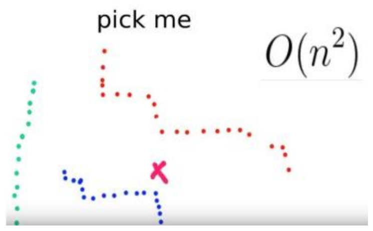
Ride-sharing applications gain large popularity in the last few years. [6] which is a dynamic ride-sharing system, which supports large scale real-time ride sharing with service guarantee on road networks and focuses on finding (1) the fastest shortest path algorithm (2) fast dynamic matching algorithms to schedule ride-sharing [7] which introduce two heuristic algorithms based on greedy method and the time-space network for the case of one origin to many destinations [8] which offers a system for ridesharing for college students and faculty members with sustainable mobility. [9] which offer activity-oriented carpooling service. the mobile app collect users data using GPS then apply a clustering method to suggest carpooling opportunities and the driver can offer a ride to the user also user can set a plan specifying the path and the time of the trip and the system will filter the results with a ranking system

III. METHODOLOGY

Pick-me is based on two main components (Client-side as a mobile application and server-side) the client-side uses Firebase phone, Auth, to prevent spammers from using the application, upon login the system uses GPS to take a snapshot of location every 70 meters (Figure 1a), then upload the snapshots to the server-side.
The server-side analyzes each location snapshot to determine the route, start and endpoints and create a user valid route. As it is impossible to get the same points every time, even on the same route. We used DTW [2], an algorithm to check if the points belong to a pre-trained route if so, increment the route weight and if not add a new route.
During the route analysis, we needed to check if it’s a valid route. Depending on the distance covered by the user was not enough as the user can travel a large distance by walking back and forth in one location, so we had to depend on the total covered area by the user which is achievable by creating a bounding box around the route (Figure 2) then use the Euclidean distance to calculate the bounding area.

In order to fetch recommendations, the system must find the best routes to suit user need and by comparing the user location to all the points around him was proven to be resource and time consumer leading to $O(n^2)$ complexity (Figure 1b).


(a) multiple routes formed by taking a snapshot every 70 m



(b)analyzing routes by each route points

Fig.1.

To avoid looping through all the points provided by a route, we used the previous bounding box to detect early the routes of interests which reduced complexity to O(n) (Figure 3a). After detecting the routes of interest, the system must find the best meeting point which is achievable by looping through all the points, but this method requires a high computation and time-consuming loops, so we have introduced a new algorithm based on Gradient Descent [3]. Our algorithm maps multiple crawlers then uses the slope to skip the unneeded points, each crawler moves in the direction that minimizes the distance between two routes (Figure 3b), afterwards, the server responds to the client with the filtered routes and the user can choose the best for him.

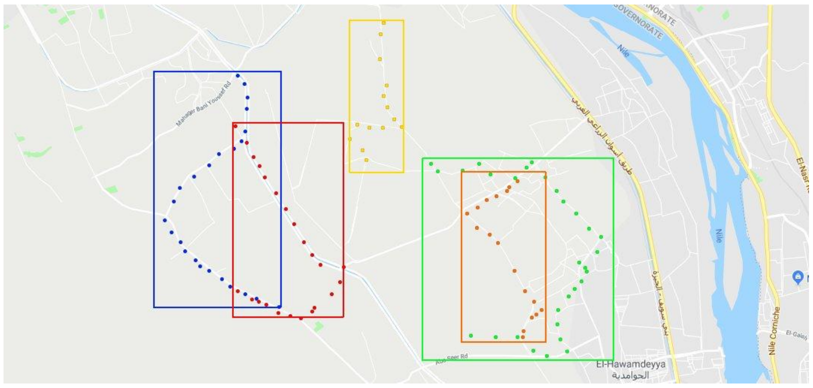


Fig. 2. bounded routes

IV. TEST AND RESULTS

A. Test preparation

The testing process wasn't a straightforward approach since the Application requires travelling the same route multiple times before making sure that the user is using this route on the regular basis, besides it requires long-distance traveling to prove its efficiency.
In order to achieve writing multiple test cases, we have implemented a manual In order to achieve writing multiple test cases, we have implemented a manual mode which allows the tester to point the interesting locations on the map, but the manual mode wasn't capable of giving the same efficiency as the system, as the system takes a snapshot of location every 70 meters which isn't achievable using only the hand as input so a 'points generating module' which allows the tester to point the interesting locations on the map, but the manual mode wasn't capable of giving the same efficiency as the system, as the system takes a snapshot of location every 70 meters which isn't achievable using only the hand as input so a 'points generating module' is a must.
The 'points generating module' creates a location snapshot for each 70m between the start and endpoint to fill the gap from the manual mode (Figure 4a, 4b). It uses the Euclidean distance $\sqrt{(lat1 - lat2)^2 + (lng1 - lng2)^2}$ to calculate the distance between the two points before dividing it by incrementor value 0.0006295 (inc=0.0006295) which is equivalent to 70m [4] in order to calculate the number of points required to fill in between the two points.

Since we are working on 2D-Axis we have to use the angle between the line and the x-axis to calculate the ratio of the incremental value. The incremental-longitude (inclng) and the incremental-latitude (inclat) are derived from the following equations (Figure 4c)

$$inclng=sin(\theta) * inc \qquad (3)$$

$$inclat=cos(\theta) * inc \qquad (4)$$

Wherefrom Figure (5a)

$$sin(\theta) = \frac{y2 - y1}{\sqrt{(y2 - y1)^2 + (x2 - x1)^2}} \qquad (5)$$

$$cos(\theta) = \frac{x2 - x1}{\sqrt{(y2 - y1)^2 + (x2 - x1)^2}} \qquad (6)$$

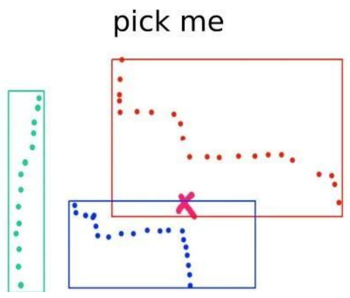
The resulted points will be perfectly aligned (Figure 6a) which will not be the case on real-life deployment so we had to feed the resulted points to a random function using Normal distribution with std equivalent to 15m as default value and mean equals to each generated point, this will generate none regular routes simulating GPS failure or small turns (Figure 6b).

B. Testing Modules

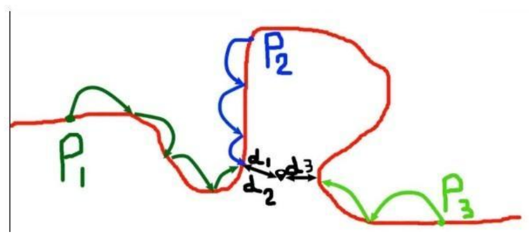
In the testing process, we have used a Laptop with Intel Core i3-6006U Processor and 4GB DDR4 Ram. We test on 16 collected routes with 4 distinct routes and we have 4 similar routes with different points for each distinct route, we used 2 main modules (clustering and recommendation) to remove redundant routes and recommend a route to the user.

1) Clustering module: We used through the test a two distinct routes daily for 4 days, the clustering for the first route was 98%, while the clustering for the second route was 90%, and the average accuracy for DTW was 84% in 7 times, while the DTW average time took 0.008s based on the number of routes in the database.

2)Recommendation module: We had 3 users to use our system, user 1 goes through route 1 and route 2 daily, while user 2 and user 3 go through route 2 daily, user 2 requested a recommendation, the system successfully recommended him to use route 2. We have iterated through the previous test 25 times with a different set of routes with an average recommendation accuracy of 98% and took an average time of 0.005s.

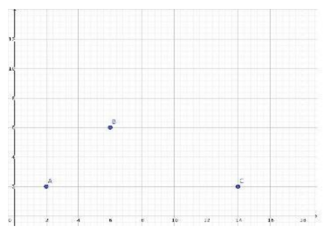


(a) search the intersected routes only

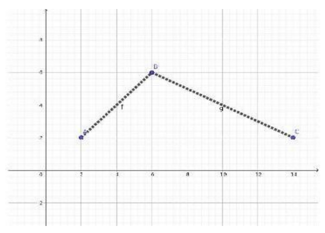


(b) multiple crawlers minimizing the distance

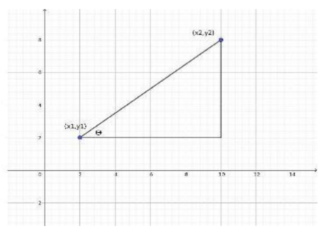
Fig. 3.



(a) Manual entered keypoint



(b) Generated points



(C) using the angle to calculate the ratio of the incremental value

Fig. 4.

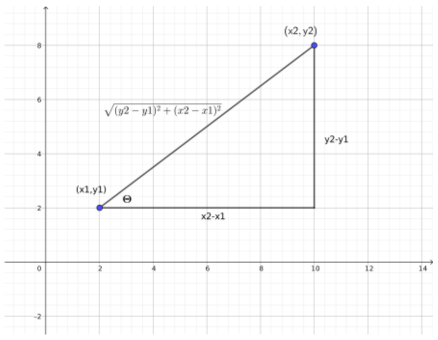
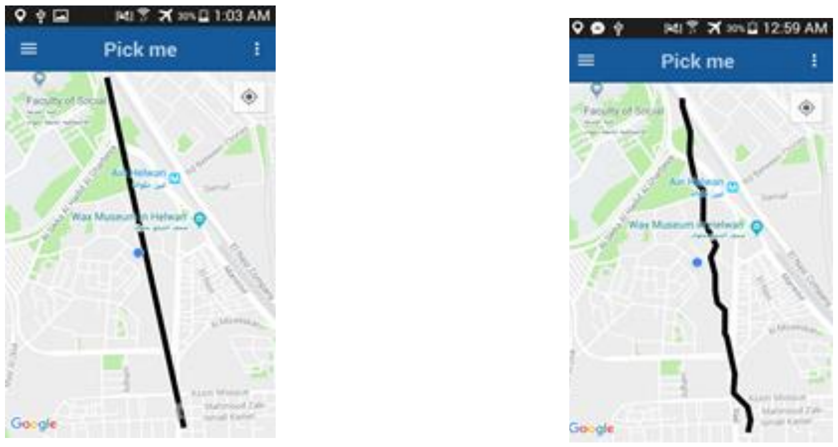


Fig. 5. calculating the coordinates of the next point



(a) perfectly aligned points (b)none regular generated routes

Fig. 6.

V. CONCLUSION

We have presented an Alpha version with a 98% recommendation accuracy, ’Our algorithm’ improved the efficiency and helped to reach this accuracy and to iterate through a large set of location snapshots with a very small time. Other solutions focused on implementing a straightforward approach, we decided to use an automated solution that minimizes the cost on the user and helps to reduce fuel consumption, traffic and pollution. We have focused on this release on efficiency and reducing server load to allow the highest number of users. We are planning to improve DTW algorithm detection accuracy on the next phase besides learning from user activity, predicting user activity according to job title, adding extra privacy layer and clustering users according to their preferences.

REFERENCES

[1] (2016) The number of cars worldwide is set to double by 2040. Matthew Nitch Smith. 7 August 2018. https://www.weforum.org/agenda/2016/04/ the-number-of-cars-worldwide-is-set-to-double-by-2040
[2] (2007) Dynamic Time Warping. In: Information Retrieval for Music and Motion. Springer, Berlin, Heidelberg, 69-74
[3] Bottou L. (2010) Large-Scale Machine Learning with Stochastic Gradient Descent. In: Lechevallier Y., Saporta G. (eds) Proceedings of

COMP-STAT’2010. Physica-Verlag HD, 177-178
[4] Calculate distance, bearing and more between Latitude/Longitude points. 8 August 2018. https://www.movable-type.co.uk/scripts/latlong.html
[5] Agatz, Niels, et al.” Optimization for dynamic ride-sharing: A review.” European Journal of Operational Research 223.2 (2012): 295-303.
[6] Tian, Charles, et al.” Noah: a dynamic ridesharing system.” Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. ACM, 2013.
[7] Tao, Chi-Chung, and Chun-Ying Chen.” Heuristic algorithms for the dynamic taxipooling problem based on intelligent transportation system technologies.” Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007). Vol. 3. IEEE, 2007.
[8] Hasan, Raza, et al.” Smart peer carpooling system.”2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC). IEEE, 2016.
[9] Monteiro de Lira, Vinicius, et al.” The ComeWithMe system for searching and ranking activity-based carpooling rides.” Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, 2016.