Group Formation in Shared Spaces

Abstract. Forming groups is natural and essential in modern traffic. E.g. pedestrians form groups to gain dominance from vehicles before they cross; grouping can reduce the number of agents that need to be considered, which makes traffic planning less complicated. However, grouping is not well investigated in shared spaces given the dynamic environment and interactions in mixed traffic. In this paper, we apply a dynamic facility location algorithm based on appearence time, origin and destination of road users before crossing to explore an appropriate grouping strategy in shared spaces, in order to improve the safety and efficiency of road users.

Keywords: shared spaces, clustering, facility location, dynamic algorithm

1 Introduction

In urban traffic, road users are often found moving in groups. These groups can be formed for different reasons. For instance, social connections (e.g. friends, couples, families) between pedestrians; mixed groups formed by traffic regulations, i.e. road users who follow the same phase of traffic lights, etc. The members of the same group interact differently to other road users in comparison to individuals (Aveni, 1977), and they tend to keep similar speed and appropriate distance (Yamaguchi et al., 2011).

An obvious benefit that comes from grouping is safety. Being in a group creates a buddy system where people can look after one another on the streets. Jacobsen (Jacobsen, 2003) found that people walking and bicycling in larger groups are less likely to be injured by motorists because the motorists are more cautious with groups. It will also have a beneficial effect on traffic planning: if groups are formed, this leads to a reduction in the number of road users that have to be included in computations, thus leading to a decrease in computational complexity for later applications, e.g. for traffic simulation and pedestrian navigation.

Although forming groups are common and natural for road users in normal traffic scenarios, there are only a few pieces of research on how to form groups in the shared space, where the traffic features (e.g. curbs, zebra lines and traffic lights) are removed to minimizes demarcations between vehicles and pedestrians. Some studies suggested an increased risk at higher traffic volumes in shared spaces (Quimby and Castle, 2006; Reid et al., 2009). Later, Holmes (Holmes, 2015) launched a survey to find out about people's experiences of using shared spaces in towns and cities. Pedestrians felt strongly that drivers did not recognize a shared space and were not slowing down to allow people to cross. Problems were pronounced in areas with high volumes of traffic or through traffic. Apart from safety aspects, currently shared spaces have efficiency problems as well: the bottleneck effect happens when traffic density is high. With the background above, we can conclude that the formation of road user groups before and during crossing will improve the safety and efficiency in shared spaces.

However, how to form groups is not a trivial task in shared spaces. Firstly, the traffic signs are removed from the road surface, therefore, the location and number of groups should be decided, which is different from most area traffic management methods (Qadri et al., 2020; Ramadhan et al., 2020). Secondly, in shared spaces, the coming and leaving slots could be different in different periods, therefore, a dynamic algorithm is needed.

Different approaches to data stream clustering have been proposed since the 90's (Aggarwal et al., 2003; Kranen et al., 2011). These approaches use a two-phase process - taking the raw data stream in real time to produce some templates and use the templates offline to generate the clusters. Later, the so called fully dynamic clustering algorithm considered both insertion and deletion of data (Cohen-Addad et al., 2016; Chan et al., 2018).

Group initialization has also been addressed. (Huang et al., 2014) considered the dynamic group behaviors via specifying the group shape as a queue and give a deformation penalty, which is effective but cannot be generalized to other group shapes. (He et al., 2016) clustered the original groups by the pairwise similarity metric defined over agents based on their starting positions and velocities. This works for the simulation application because the agents who are together at the beginning will keep coherent until the end of the experiment. (Szkandera et al., 2017) simply used a threshold based on the distance between the team leader and members to group the pedestrians with similar OD. However, this approach is sensitive to the order of the input data because the algorithm is greedy - once the first possible solution is accepted, other solutions will never be reconsidered.

In this paper, we will concentrate on the following application scenario: road users can appear from random locations around the shared space, then pass through, finally leave to their destinations. We are searching for a dynamic clustering algorithm to assign road users to several groups according to their origins, destinations and appearance time.

2 Methodology

Before crossing a shared space, a set of road users who have similar origins and destinations (OD) during a finite time will become a group. The finite time is called period. A period can contain serval groups, each group has one user as its center. A road user can wait for some time to form a group with the others. But if it is spatially or temporally far away from the others, it will become a single-member group. The problem is to decide the best location for the group centers in each period, minimizing the total cost for assigning group centers and reaching surrounding group members.

The problem above can be formulated as a series of facility location problems. In a basic formulation, the facility location is the following: giving a set of demand points and a set of candidate facility sites with costs of building facilities at each of them, the goal is to select a subset of sites where facilities should be built. Each demand point is then assigned to the closest facility, incurring a service cost equal to the distance to its assigned facility. The objective is to minimize the sum of facility costs and the sum of the service costs for the demand points (Charikar and Guha, 1999). However, solving a series of facility location problems at once is NP-hard, so the best hope is to use an algorithm with a provable approximation of the best solution.

In the following, we present a definition suitable for shared spaces. Assuming a stream of road users who come continuously and independently from all the directions of a shared space. For each road user, the coordinates of its origin *(ox,oy)*, destination *(dx,dy)*, and appearance time t are known. All road users have a maximum static waiting time w before crossing, two numerical value h and n for coarse solution (see subsection 2.1), and a cost f to estimate the spatial similarity between two road users with their OD data (see subsection 2.2).

2.1 Meyerson's algorithm and coarse colution

Adam Meyerson posted an online facilty location algorithm with O(1)-competitive (Meyerson, 2001) The algorithm is straightforward - when a new demand point arrives, the distance d between the point and the closest already-open facility is measured. The opening cost for a facility is f. With probability d/f (or probability 1, if d/f is more than 1), a new facility opens at the demand point. Otherwise the point will be assigned to the closest open facility.

Before any computation, our algorith will take the first h data, shuffle them, and run Meyerson's algorithm independently n times. The solution of facilties with lowest cost θ will become the coarse solution (hereinafter referred to as *MeyersonManyTimes*).

2.2 Similarity measure

The distance metric to estimate the spatial similarity between two road users is the sum of the Euclidean distance between two origin points and the Euclidean distance between two destination points (hereinafter referred to as *ODsimilarity*).

2.3 Dynamic facility location for shared spaces

Our algorithm is based on the framework proposed by (Cohen-Addad et al., 2019), which clusters dynamic and consistent points with a sliding window. Starting with a solution given by Meyerson's algorithm of cost θ, and maintain a solution during $\theta/(4\alpha f)$ updates, and then recompute from scratch. A point is inserted/deleted when it enters/exists the window, respectively. The new coming data either becomes a new facilty if it is far away from all existing facilities, or be assigned to the closest facilty. Each time when deletion happens, the algorithm removes the cost from the deleted point to its closest facilty.

Since the sliding window continuously takes single point each time, the framework doesn not fit the dataset with temporal gaps. Meanwhile, the road user data does not have the characteristics of first-in-first-out (FIFO), i.e. the user who comes earlier with lower speed may leave late, thus, the cost reduction of deletion make no sence either.

We modified the framework with the new parameter w, new distance metric *ODsimilarity* and changed the cost function, so it can adapt to our application.

Our algorithm works as follows: Starting with the first *h* items of dataset, if the the time interval of the 1st and hth item is less than *w*, the coarse solution is clculated with *MeyersonManyTimes* (centers recompute); if the interval is larger than *w*, the initial points are splited to make sure the new time interval smaller or equal to *w*, then apply *MeyersonManyTimes*. As the coarse solution (a set of group centers) is ready, the new coming users are inserted one by one. The distance between the new user to all existing group centers is calculated via *ODsimilarity*. If the minimum distance *d* is larger than the center construction cost *f*, a new center is built at the location of the new point (centers update), otherwise, the new point will be assigned to its closet center. Centers update will continue until the update time since last recomputation reaches the threshold of *θ/(4f)* or the time interval is over *w*, if so, run *MeyersonManyTimes* and start whole process again. In this way, the cost cancels when a group of users left. More details is shown in Alg. 1.

Algorithm 1: Dynamic facility location for shared spaces

Input: A set of data *X* including OD and appearing time t for each road user

Output: A set of centers *F* at each period, an assignment *B* of point to centers, cost per period C

```
ub = lb = 0 // manage the upper/lowerbound of data

while X is not empty do
  Δt ← t_ub - t_lb
  if Δt > w then
      remove first item from X until max time interval < Δt
      compute centers with removed items via
MeyersonManyTimes
      add centers to F
      add construction cost and assignment cost to C
      lb = lb + 1, ub = new ub
      // recompute because of over waiting time
  else
      take the first h items
      if current time - last recompute time > θ/4f then
         recompute centers by MeyersonManyTimes
         add centers to F
         add construction cost and assignment cost
         // recompute because of cost criteria
      else
         if  min(ODsimilarity) < f then
            add assignment cost to C
            break
            // too close to open a new facil
         else
            add the h^th item of X to F
            add f and assignment cost to C
            // update
         end
      end
  lb = lb + 1,  ub = ub + h
  end
end
```

2.4 Software and Data Availability

The first video of scenerio "DeathCircle" of the Stanford Drone Dataset (Robicquet et al., 2016) is used to apply all the operations. The video contains 703 road users and lasts about 7 minutes. After being filtered via labels to make sure there are no lost or occluded trajectory points, 663 vaild trajectories left. The origin and destination (Fig. 1) and appearance time are extracted as input of the experiment.

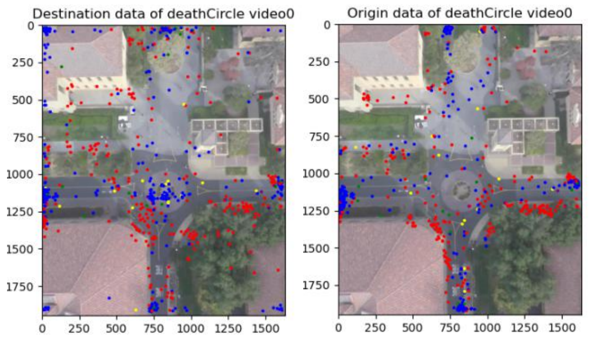

Figure 1: The OD plots of the Stanford Drone Dataset (colors represent different user types, x and y axis are relative image coordinates)

The proposed methods were implemented in python. All experiments were performed on a computer with the CPU Intel Core i5-8250U CPU @ 1.60GHz × 8 and Memory 7.7 GiB. The code is available at github.

3 Experiments and Results

In this section, we evaluate our algorithm by adjusting cost *f*, waiting time *w*, and numerical value *h*. We will also compare the algorithm against the other two algorithms, i.e. Meyerson's algorithm and the original framework from Cohen-Addad. Our implementation follows Alg. 1 posted in the previous section. As part of the re-computation step between two periods, we set n to 5, i.e. run independently Meyerson's algorithm five times, and select the execution with lowest cost *θ*.

3.1 Parameters and results

Cost *f* restricts the sum of Euclidean distance between origins and destinations of two road users. The higher value of f produces bigger groups and speeds up the computation of the groups. However, the value of *f* depends on the dataset as well. For instance, in DeathCircle, the lane width is about 100. To group the road users from the same lane, *f* is set as 200, including the offsets from origin and destination.

Waiting time *w* decides how long can users wait for the others to form a group. The higher value of *w* can form fewer groups with more group members, which leads to a higher cost and computation time.

Numerical value *h* takes the users for recomputation between each period. If *h* equals 1, there will be no coarse solution before each period, i.e. the road user who comes first will become the center of the group. For large or adversary dataset which has complex groud truth, *h* should be large enough to avoid bias. In DeathCircle, most road users follow the shape of lanes or come from buildings, which is not complex, so the *h* can be set to a small value.

As discussed above, we set *f*, *w* and *h* to 300, 200 and 15, respectively to get a suitable solution for the current dataset. In each period, constructing a minimum of 6 groups and a maximum of 30 groups, including those which only have a single member. The average group size is 1.41. These parameters are also utilized in subsection 3.3 to compare the results from different algorithms.

Fig. 3 shows the grouping results of a period in DirthCircle. The trajectories with the same color belong to the same group. The colored crosses are the generated centers of corresponding groups. There are 5 groups, four of them contain only one road user, and one group has 3 members. All road users are grouped with their origins, destinations, and appearance time.

Another interesting question is when almost all the road users are included in groups that contain at least two members. Since *h* does not affect users' aggregation, we changed *f* to 300, and *w* to 710 to make the average group size over 2.

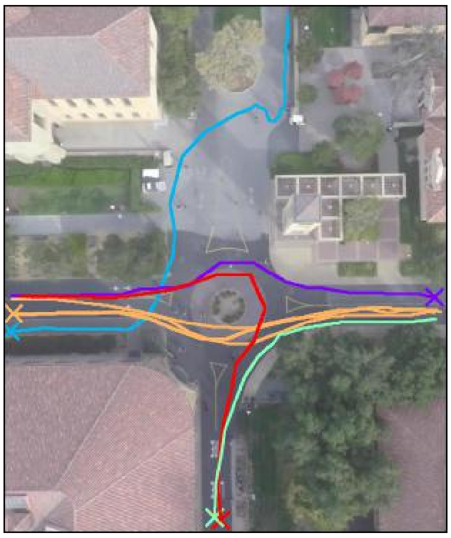
Figure 2: Group result of one period

3.3 Comparision with other algorithms

Used the parameters calculated above, we generate the plots for cost and running time per update against Meyerson's algorithm and the framework from Cohen-Addad, see Fig. 3.

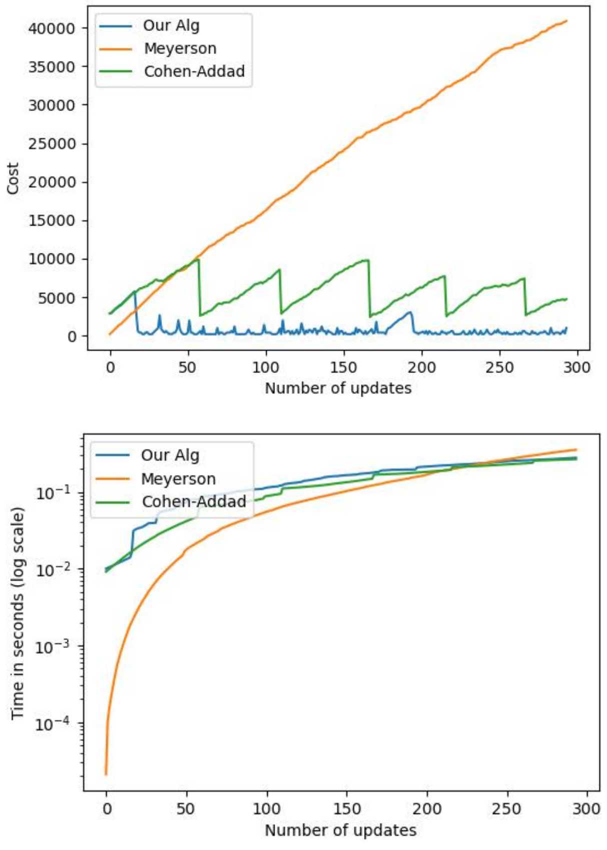

Figure 3: Comparision of cost and running time per update against Meyerson's algorithm and the framework from Cohen-Addad

The cost of Meyerson's algorithm has a linear dependency on the number of updates because it never deletes the road users. This is also what our algorithm takes advantage of, by restrict the waiting time and cost criteria, the insertion and deletion of road users happen continuously, which keeps the cost in a low range. Comparing to Cohen-Addad's, our algorithm has additional recompute criteria and natural deletion condition, which leads to less and stabler cost.

Our algorithm has a similar running time as Cohen-Addad's. Compared to Meyerson's algorithm, ours is slower initially, but as the number of processed road users increases, the slope of running time for Meyerson's algorithm is larger than our method, as it never removes a group center once it has been opened, the time to compute the distance metrics from new coming road users to all existing group centers is therefore increasing.

4. Discussion

In this section, we will discuss some interesting parts for optimizing the parameters *w*, *f*, *h*, and potential aspects for improvement.

A possible case is the road users who have similar OD data but diverse waypoints, as the two orange trajectories in Fig. 4. To solve the problem, we can replace the Euclidean distance with symmetrical Hausdorff distance (Edwards, 1975) as distance metric to avoid wrong grouping.

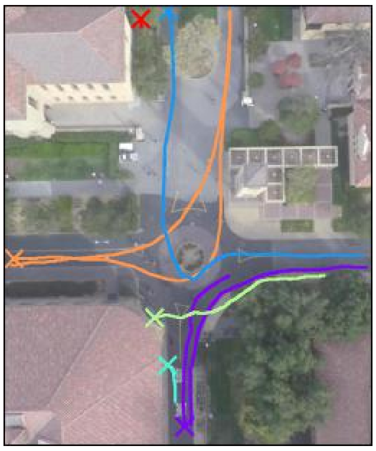

Figure 4: Benefit from Hausdorff distance (sperate the users who have similar OD but different waypoints)

In the previous section, we have shown that waiting time *w* and cost *f* are two important factors to form a group. However, in general, they may not be the same static value for all road users but highly depend on the road user‘s personality (e.g. patient or reckless). For sparse traffic scenario, it is not necessary to apply the method, because people need a trade-off between waiting time and forming group, and sparse dataset take advantages of being alone.

One drawback of the method is that it cannot deal with splitting and merging of groups. Recall the red and purple group in Fig. 2, they started from two different lanes but got closer after passing through the roundabout. If the splitting distance is not far away, Hausdorff distance may help, but for a huge gap, current algorithm doesn not work.

5. Conclusion and future work

In this paper, we proposed a dynamic facility location method to cluster road users which has similar OD and existing time in a shared space. In future work, we will adopt this method for all kinds of road users with intergrating average velocities to distance metric, and import trajectoris to deal with the splitting and merging cases.

References

Aveni, A. F. The not-so-lonely crowd: Friendship groups incollective behavior. Sociometry, 40(1):96–99, 1977. ISSN 00380431. URL: http://www.jstor.org/stable/3033551.

Charikar, M. and Guha, S. Improved combinatorial algorithms for the facility location and k-median problems. In Annual Symposium on Foundations of Computer Science - Proceedings, pages 378–388. Institute of Electrical and Electronics Engineers Computer Society, 1999. doi: 10.1109/sffcs.1999.814609.

Cohen-Addad, V., Schwiegelshohn, C., and Sohler, C. Diameter and k-center in sliding windows. In Leibniz International Proceedings in Informatics, LIPIcs, volume 55. Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, aug 2016. ISBN 9783959770132. doi: 10.4230/LIPIcs.ICALP.2016.19.

Cohen-Addad, V., Hjuler, N., Parotsidis, N., Saulpic, D.,and Schwiegelshohn, C. Fully dynamic consistent facility location. In Advances in Neural Information Processing Systems, volume 32, dec 2019. URL https://hal.archives-ouvertes.fr/hal-02360783.

Edwards, D. A., The Structure of Superspace, Editor(s): Nick M. Stavrakas, Keith R. Allen, Studies in Topology, Academic Press, 1975, Pages 121-133, ISBN 9780126634501, https://doi.org/10.1016/B978-0-12-663450-1.50017-7.

Holmes, L. Accidents by Design: The Holmes Report on ”shared space” in the United Kingdom. (July):1, 2015. URL http://chrisholmes.co.uk/wp-content/uploads/2015/07/Holmes-Report-on-Shared-Space-.pdf

Huang, T., Kapadia, M., Badler, N. I., and Kallmann, M. Path planning for coherent and persistent groups.In Proceedings - IEEEI nternational Conference on Robotics and Automation, pages 1652–1659. Institute of Electrical and Electronics Engineers Inc., sep 2014. doi:10.1109/ICRA.2014.6907073.

Chan, H., Guerqin, A., and Sozio, M. 2018. Fully Dynamic k-Center Clustering. In Proceedings of the 2018 World Wide Web Conference (WWW '18). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 579–587. DOI: 10.1145/3178876.3186124

Kranen, P., Assent, I., Baldauf, C. et al. The ClusTree: indexing micro-clusters for anytime stream mining. Knowl Inf Syst 29, 249–272 (2011). https://doi.org/10.1007/s10115-010-0342-8

Jacobsen, P. L. Safety in numbers: More walkers and bicyclists, safer walking and bicycling. Injury Prevention, 9(3):205–209, sep 2003. ISSN 13538047. doi: 10.1136/ip.9.3.205. URL: www.injuryprevention.com.

Meyerson, A., "Online facility location," Proceedings 42nd IEEE Symposium on Foundations of Computer Science, Newport Beach, CA, USA, 2001, pp. 426-431, doi: 10.1109/SFCS.2001.959917.

Quimby, A. and Castle, J. A Review of Simplified Streetscape Schemes. Technical report, 2006.

Qadri, S. S. S. M., Gökçe, M. A. & Öner, E. State-of-art review of traffic signal control methods: challenges and opportunities. Eur. Transp. Res. Rev. 12, 55 (2020). https://doi.org/10.1186/s12544-020-00439-1.

Ramadhan, S. A., Sutarto, H. Y., Kuswana, G. S., and Joelianto, E. Application of area traffic control using the max-pressure algorithm. Transportation Planning and Technology, 43(8):783–802, nov 2020. ISSN 10290354. doi: 10.1080/03081060.2020.1828934.

Reid, S., Ko-cak, N., Reviewer, L. H., and Emslie, J. DfT Shared Space ProjectStage 1: Appraisal of Shared Space Report for Department for Transport Document Control Project Title: Shared Space Research MVA ProjectNumber: C3783100 Document Type: Report Document Approval. Technical report, 2009.

Ren, J., Ma, R., and Ren, J. Density-based data streams clustering over sliding windows. In Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery - Volume 5, FSKD’09, page 248–252. IEEE Press, 2009. ISBN 9781424445455.

Robicquet, A., Sadeghian, A., Alahi, A., and Savarese, S. Learning social etiquette: Human trajectory understanding in crowdedscenes. In Lecture Notes in Computer Science (including subseries LectureNotes in Artificial Intelligence and Lecture Notes in Bioinformatics),volume 9912 LNCS, pages 549–565, 2016. ISBN 9783319464831. doi: 10.1007/978-3-319-46484-833.

Szkandera, J., Kolingerová, I., Maňák, M., Path Planning for Groups on Graphs, Procedia Computer Science, Volume 108, 2017, Pages 2338-2342, ISSN 1877-0509, doi: 10.1016/j.procs.2017.05.040.

Yamaguchi, K., Berg, A. C., Ortiz, L. E., and Berg, T. L. Who are you with andwhere are you going? In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 1345–1352. IEEE Computer Society, 2011. ISBN 9781457703942. doi: 10.1109/CVPR.2011.5995468.