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 they crossing to explore an appropriate grouping strategy in shared spaces to improve the safety and efficiency of road users.

**Keywords**: shared spaces, clustering, facility location, dynamic algorith

# 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. Last but not least, a group may gain or lose members on the fly (Coleman and James, 1961), which is called splitting and merging. The methods should consider the coexistence of users' trajectories.

Early techniques have been mainly used to simulate static or fixed-sized groups and perform group-based collision avoidance (Graciano and Chaimowicz, 2014; He and Berg, 2013; Karamouzas and Guy, 2015). Group initialization has also been addressed. (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. (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. However, the traffic scenario is more complicated. E.g. the road users who have the same origin and velocity at the beginning may split and reach different goals later. (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. However, none of these methods can efficiently simulate groups with dynamic settings.

# 2 Methodology

In this paper, we 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. A set of road users who need to be grouped during a finite time according to the similarity of their origins and destinations. The finite time is called period. Each period will contains a set of center users, each of them represents a group. The problem is to decide the best location for the group centers in each period, minimizing the total cost for reaching surrounding group members.

## 2.1 Dynamic facility location for shared spaces

The problem above can be formulated as a series of facility location problems. In a basic formulation, the facility location problem 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 problems by 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 static waiting time w before crossing, a numerical value h for initialization, and a threshold f to estimate the data points similarity. Here, the similarity metric (ODsimilarity) is the sum of Eclidean distance between the origins and destinations of a point pairs.

Our algorithm is based on the framework proposed by (Cohen-Addad, 2019), which clusters dynamic and consistent points with a sliding window. The framework is suitable for points other than trajectories. Moreover, it continuously takes single point each time regardless of the restriction in time. Therefore, the framework is modified to adapt to our application. The pseudo-code of the current algorithm is shown in Algorithm 1.

As part of the re-computation step between two periods, we run 5 independent executions of Meyerson’s algorithm, and selecting the execution with lowest cost. The updates within a period are handled by assigning to closest center if distance is less than f or otherwise open a new center at the point, and we simply remove a client if it gets deleted.

When the interval does not exceed w, a new point is taken, the transport cost from the new data point to all current centers is calculated. If the minimum distance d is larger than the facility construction cost f, a new center is built at the location of the new point (update), otherwise, the new point will be assigned to its closet centers. The update will continue until the update interval reaches the threshold of update θ/4αf (recompute again). The reason for θ/4αf is given in (Cohen-Addad, 2019).

|  |
| --- |
| **Algorithm 1**: Dynamic clustering in shared spaces |
| **Input:** A set of data points *X* , distance metrics *ODsimilarity(o1,o2,d1,d2)*  **Output:** A set of centers *F*, an assignment *B* of point to centers  **Initialize:** coarse solution *MeyersonManyTimes( first h number of data points,n )* |
| **while** *X* is not empty **do**  *Δt* ← max time interval of first h items in *X*  **if***Δt > w* **then**  remove first item from *X* until max time interval < *Δt*  recompute centers with removed items by *MeyersonManyTimes*  add centers to *F*  // recompute because of over waiting time  **else**  take the first *h* items  **if** current time - last recompute > θ/4αf **then**  recompute by *MeyersonManyTimes*  add centers to *F*  // recompute because of cost criteria  **else**  **if** min(*ODsimilarity*) *< f* **then**  **break**  // too close to open a new facil  **else**  add the *h* item of *X* to *F*  // update  **end**  remove first item from *X*  **end**  **end**  **end** |
| Procedure *MeyersonManyTimes(first h number of data points,n)*  repeat Meyerson *n* times  initial centers ← solutions with lowerest cost θ  Procedure *ODsimilarity(o1,o2,d1,d2)*  *|o1-o2| + |d1-d2|* |

## 2.2 Software and Data Availability

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

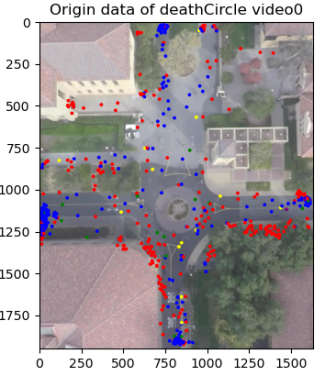
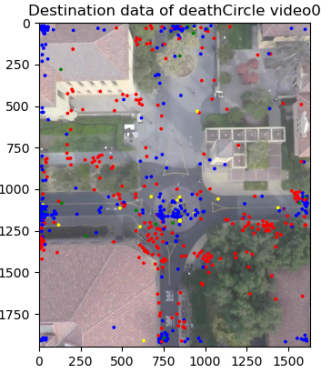


Figure 1: The OD plots of stanford drone dataset (colors represent different user types)

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](https://github.com/YaoLIII/GroupFormation.git).

# 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 and the original framework from Cohen-Addad. Our implementation follows Algorithm 1 posted in the previous section, the n to get coarse solution is set as 5.

## 3.1 Parameters

Cost f restricts the sum of Eucliean distance between origins and distinations of two road users. The higher value of f produces bigger groups and speeds up the computation of the groups (Tab. 1). However, the value of f depends on the dataset as well. For instance, in Deathcircle, the lane width is about 100 (image coordinate). To group the road users from the same lanes, f is set as 200, including the offsets from origin and destination.

|  |  |  |  |
| --- | --- | --- | --- |
| w=300, h=100 | f = 100 | f = 500 | f = 1000 |
| time (s) | 0.584 | 0.430 | 0.366 |

We restricted the number of points considered due to time constraints. Since larger data sets typically have more complicated ground truths, we used a larger windows for them containing more samples. To avoid overfitting, we also adjusted the cost of opening a facility depending on the window size, i.e. for larger windows a lower opening cost per facility. For

Waiting time w decides how long can users wait for the others to form a group. The results are shown in Table 2.

Numerical value h takes the users for recomputation between each period. For dataset which has complex groud truth, h should be large to avoid adversary, meanwhile, h shouldn‘t be too large to be overfitted. The results of h is shown in Table 3.

## 3.2 Results

Figure 3 shows a period of center sets (period 31, from frame 5226 to frame 5432) in "DeathCircle\_video0", the trajectories with the same color belong to the same group. The orange circles are generated centers. All road users are clustered into a group with the maximum internal similarities.

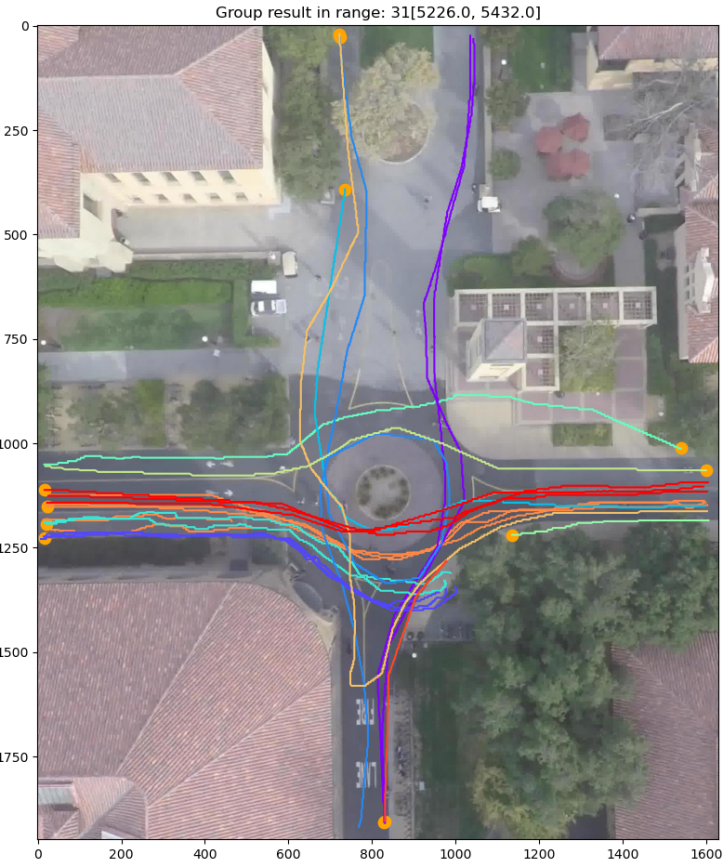


Figure 3: Group result of one period

Another possible situation is the road users who have similar OD data but diverse waypoints. We import symmetrical Hausdorff distance as a distance metric to avoid wrong clustering in this case (see Figure 4).

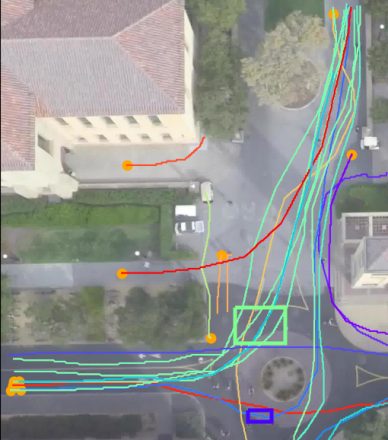
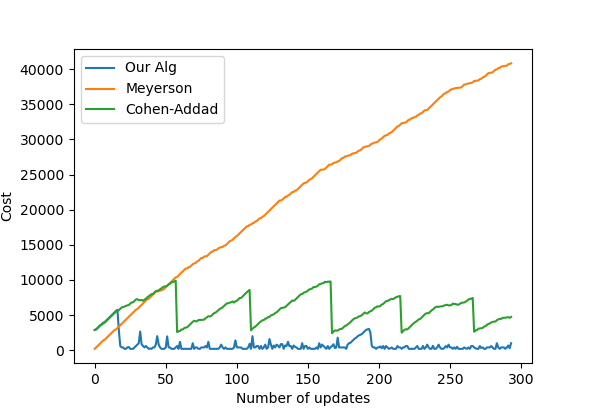


Figure 4: Benefit from Hausdorff distance (marked by rectangles: the dark blue road user is separated from the green group even they have similar OD)

## 3.3 Comparision with other algorithms

Used the parameters calculated above, we generate the plots for cost and run time per update against Meyerson and Cohen-Addad, see Figure 2.



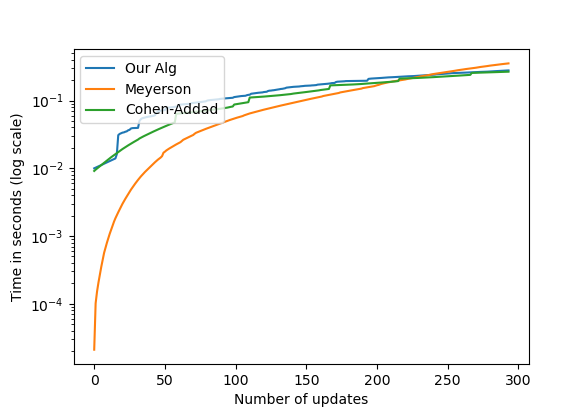


Figure 4: Benefit from Hausdorff distance (marked by rectangles: the dark blue road user is separated from the green group even they have similar OD)

Our algorithm has similar running speed as Cihen-Addad. Compared against Meyerson algorithm, ours is slower initially, but as the number of processed road users becomes more, the running time of Meyerson deteriorates comparatively, as it never removes a facility once it has been opened: the time to compute the distance to the set of facilities is therefore increasing.

The cost of MeyersonSingle generally has a linear dependency on the number of updates, though the slope is very gentle. This is also what our algorithm takes advantage off, broadly speaking by approximating the curve with a step function (adapted to handle insertions and deletions). The cost of our algorithm and MeyersonRec is basically indistinguishable, and in certain cases our algorithm fares even slightly better. The recourse of our algorithm is expectedly much better than MeyersonRec by a wide margin, and significantly worse than MeyersonSingle.

Finally, we ran our algorithm with multiple choices of facility cost f, and we observed that the recourse is almost independent of the both cost and running time of the algorithm, and only depends on the number of updates. This is consistent with tracking evolving data in time, where the underlying ground truth clustering also evolves in time.

## **Discussion**

In this section, we will discuss some interesting parts for optimizing the parameters w, f, h and potential aspects for improvement. In the previous section, we have shown that waiting time and search area 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 of waiting time and forming group, and sparse dataset take advantages of being alone.

One drawback of the method is that, if you check the red and dark blue trajectories that start crossing from left to right in Figure 3, they can be good groups at the beginning, but split to different destinations in the end. When the splitting distance is not far away, Hausdorff distance works well, but for instance, we both come from the north, then split to west and east direction, the current algorithm cannot help.

## **Conclusion and future work**

In this paper, we propose 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 the help of type labels and average velocities.

# References

## 3.4 Figures and Tables

Figures and tables can be placed in a single column, or span over both columns if required. The figure caption names the number of the figure as indicated in the example. Reference to figures in text shall use abbreviations like Fig. 1, where a non-breaking space is added before the number. The caption of the figure spells out Figure 1.

Larger figures, e.g., full page graphics, should be added in the appendix.



Figure 1: The logo of AGILE – the Association of Geographic Information Laboratories in Europe.

Tables shall be referenced in the text with Tab. 1. where a non-breaking space is added before the number; the caption of the table spells out Table 1.

## 3.5 Equations

Equations should be prepared with the Microsoft Word equation editor or LibreOffice Math. They are numbered sequentially and referenced in the text with Eq. (1). The numbers of the equations shall be next to the equation to the very right in parentheses.

(1)

## 3.6 References

References in the text are using the author date format as the following example (Author and Author, 2019). In case of more than two authors, the list authors is abbreviated (Author et al., 2020). Use a non-breaking space in “et al.”.

The section providing the references is indicated with the term references as heading 1, without numbering. The references are sorted alphabetically on first author surnames. The information provided covers author names, title of the publication, publication outlet, volume and issue, page numbers, DOI and year.

The references section on the website of Copernicus Publishers for manuscript preparation provides detailed examples for referencing various types of publications. In addition, download links for Endnote, Bibtex and CSL (for Mendeley, Zotero etc.) style files: <https://publications.copernicus.org/for_authors/manuscript_preparation.html>.

# References

Author, F. and Author, S.: The test article, International Journal of GIScience., 12, 135–147, <https://doi.org/10.1234/56789>, 2019.

Author, O., Author, T. and Author, F.: More test articles, Transactions in GIS., 35, 13–28, <https://doi.org/10.2345/67890>, 2020.