AGILE: GIScience Series, 2, 34, 2021. https://doi.org/10.5194/agile-giss-2-34-2021
Proceedings of the 24th AGILE Conference on Geographic Information Science, 2021.
Editors: Panagiotis Partsinevelos, Phaedon Kyriakidis, and Marinos Kavouras.
This contribution underwent peer review based on a full paper submission.

Group Formation in Shared Spaces

Yao Li[a] (corresponding author), Monika Sester[a]

yao.li@ikg.uni-hannover.de , monika.sester@ikg.uni-hannover.de

[a]Institute of Cartography and Geoinformatics, Appelstraße 9A, 30167 Hanover, Germany

Abstract. In shared spaces, grouped pedestrians can gain dominance and thus get the right of way from vehicles more easily; grouping can make traffic planning less complicated, e.g. it reduces the number of agents that need to be considered while traffic planning. 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 appearance time, origin, and destination of road users before crossing a junction to explore an appropriate grouping strategy in shared spaces, in order to improve the safety and efficiency of traffic.

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

1 Introduction

The concept of "shared space" is first introduced in the 70s by Dutch traffic engineer Hans Monderman by forcing vehicles to decelerate when they pass through a mixed traffic environment, to address traffic safety problems in Netherlands (Hamilton-Baillie, 2004). The term became popular, especially in Western Europe, due to the INTERREG IIIB North Sea Project (Shared Space, 2005). According to a definition provided by Karndacharuk et al. (2014), shared space is "a public local street or intersection that is intended and designed to be used by pedestrians and vehicles in a consistently low-speed environment with no obvious physical segregation between various road users to create a sense of place, and facilitate multi-functions".

Typical design of shared spaces reduces the separation between all road users by removing traffic signals (e.g. road signs, markings, and traffic lights). In this way, the intensive interactions between all road users force the drivers to drive at a limited speed, which leads to the increment of priority and safety for other road users (e.g. pedestrians and cyclists). Compared to conventional traffic designs, shared spaces create a pedestrian-friendly environment with fewer congestions (Monderman et al., 2006).

However, some studies indicate an increased risk at higher traffic volumes in shared spaces (Quimby and Castle, 2006; Reid et al., 2009). Apart from safety aspects, currently shared spaces have efficiency problems as well: the bottleneck effect happens when traffic density is high (Moody et al., 2014). Therefore, developing an algorithm to increase the safety and efficiency of shared spaces is necessary.

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 (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.

With the background above, we can conclude that the formation of road user groups before and during crossing can improve the safety and efficiency in shared spaces. Here, a group is a formation of road users moving in a coordinated manner.

However, forming groups is not a trivial task in shared spaces. Firstly, shared spaces have no road markings such as zebra lines or lane markers, thus 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 spots of road users may vary dramatically in different periods, therefore, a dynamic clustering algorithm for group formation is needed.

Different approaches to data stream clustering have been proposed since the 90s (Aggarwal et al., 2003; Kranen et al., 2011). These approaches typically use a two-phase process: firstly, taking the raw data stream in real-time to produce potential cluster centers; secondly, the tentative centers are used offline to generate the clusters with all the data. This two-stage process is necessary, as there is no adaptation of identified clusters, which leads to a strong dependency on the sequence of processing. Later, the so-called fully dynamic clustering algorithm considered both insertion and deletion of data (Cohen-Addad et al., 2016; Chan et al., 2018).

Clustering a set of points in a metric space can be viewed as a facility location problem, which involves the following general setting: 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).

Our application can be formulated as a series of facility location problems, as the number/locations of group centers are not fixed and the road users are inserted/deleted continuously. However, solving such a series of problems all at once is NP-hard, so the best strategy is to use an algorithm with a provable approximation of the best solution.

Adam Meyerson presented an online facility location algorithm with O(1)-complexity (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 > 1), a new facility opens at the demand point. Otherwise, the point will be assigned to the closest open facility.

Cohen-Addad et al. (2019) clustered dynamic and consistent points with a sliding window. The algorithm also uses a two-phase process. Firstly, the coarse solution is given by Meyerson's algorithm using the points in the window. The cost of the coarse solution is θ. Secondly, the window slides, and maintains a solution during $\theta/(4\alpha f)$ times of updates, and then recomputes from another coarse solution. As the window slides, each time a new point is inserted, and the last point of the window is deleted. The new coming data point either becomes a new facility if it is far away from all existing facilities or is assigned to the closest facility. When deletion happens, the algorithm removes the cost of the deleted point. The drawback of the algorithm is obvious: in real-world applications, one might deal with the application with arbitrary insertion and deletion, i.e. the oldest point does not leave its cluster, in other words, it still maintains the service cost (Chan et al. 2018), rather than delete it. Moreover, the algorithm only recomputes when the cost reaches a cost threshold, which may not suitable for many applications.

The main criterion for designing a dynamic algorithm is the quality of the clustering compared with the offline optimal solutions at any moment. However, maintaining a consistent clustering, i.e. a clustering with bounded recourse (the number of changes per update) is equally important in many applications. For instance, Gupta et al. (2016) considered the running time of the algorithm making updates (update time) as well as the number of updates made to the solution (recourse) for an online and dynamic set cover algorithm. Lattanzi and Vassilvitskii (2017) considered maintaining a constant factor approximate solution while minimizing the total number of times the maintained solution changes over the whole update process.

In this paper, we introduce a hypothetical system, which is aimed at encouraging road users to form groups when it is viable. The whole process works as follows: before crossing a shared space, a set of road users who have similar origins and destinations during a variable time range will become a group. This time range is called a *period*. A period can contain several 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 s/he is spatially or temporally far away from the others, s/he will become a single-member group.

We are searching for a dynamic clustering algorithm to group road users according to their origins, destinations, appearance time, and also their planned path, to improve the safety and efficiency of shared spaces. To this end, we extend the approach of Cohen-Addad et al., (2019) by adding parameter waiting time and changing the cost function.

2 Methodology

In the following, we present a definition suitable for shared spaces. Assuming a stream of road users X who come continuously and independently from all 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. Besides, there are two global parameters, namely batch size h and iteration time n to determine the coarse solution (see subsection 2.1), and a cost f to establish a new group center. The total cost C of each period consists of the construction cost of all opened group centers and the cost to integrate surrounding group members to their corresponding centers (the distance between each group member points to its center, see subsection 2.2).

2.1 Coarse solution

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 // upper/lower bound of current data batch
while X is not empty do
  Δt ← t_ub - t_lb
  if Δt > w then
    // compute new centers because of waiting time
    remove first item from X until max time interval < Δt
      (corresponding upperbound is new ub)
    compute centers with removed items via
      MeyersonManyTimes
    add centers to F
    add construction cost and intergration cost to C
    lb = lb + 1, ub = new ub
  else
    take the first h items
    if current time - last recompute time > θ/4f then
      // recompute because of cost criteria
      recompute centers by MeyersonManyTimes
      add centers to F
      add construction cost and assignment cost
    else
      if min(ODsimilarity) < f then
        // integrate to the closest center
        add integration cost to C
        break
      else
        // center addition
        add the h^th item of X to F
        add f to C
      end
    end
  lb = lb + 1,  ub = ub + h
  end
end
```

Similar as the Cohen-Addad's algorithm, before any computation, our algorithm will take the first h data, shuffle them, and run Meyerson's algorithm (Meyerson, 2001) independently n times. The solution of group centers with the lowest cost will become the coarse solution (hereafter 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 distances between the two origin points and the two destination points (hereafter referred to as *ODsimilarity*, see Eq. (1)).

$$ODsimilarity = \sqrt{(ox_1 - ox_2)^2 + (oy_1 - oy_2)^2} + \sqrt{(dx_1 - dx_2)^2 + (dy_1 - dy_2)^2} \quad (1)$$

2.3 Dynamic facility location for shared spaces

Our algorithm works as follows: starting with the first h items of the dataset, if the time interval of the 1st and hth item is less than the waiting time w, the coarse solution is calculated with *MeyersonManyTimes* (*centers compute*); if the time interval is larger than w, the initial points are split to make sure the new time interval is smaller or equal to w, then *MeyersonManyTimes* is applied.

The construction cost of calculated centers and the integration cost are accumulated to the total cost C. As the coarse solution (a set of group centers) is ready, the new coming users are inserted one by one. The distance between a 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 (*center addition*), the cost C increases by one center's cost f; otherwise, the new point will be assigned to its closest center, and the integration cost d is added to the total cost C. Any operation (centers compute/ center addition/ user integration) is called *update*. Centers update will continue until the update time since the last recomputation reaches the threshold of $\theta/(4f)$[1] or the time interval is larger than w. The cost C is eliminated when current period ends. Then *MeyersonManyTimes* is run and the whole process is started again. More details are shown in Alg. 1.

[1] see Cohen-Addad et al. (2019), for explanation of this value.

2.4 Software and Data Availability

The first video of the scenario "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. The frame rate is 30 fps. After being filtered via labels to make sure there are no lost or occluded trajectory points, 663 valid trajectories are left. The origin and destination (Fig. 1) and appearance time are extracted as input for the experiment.

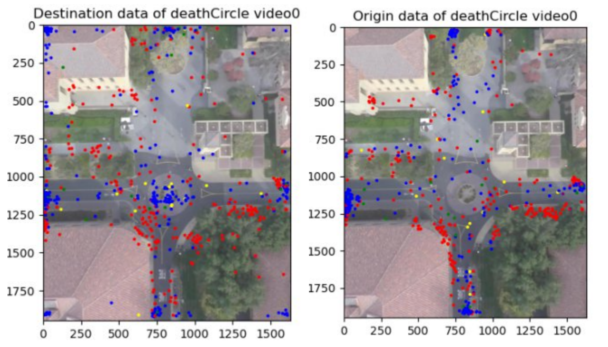

Figure 1: The OD plots of the Stanford Drone Dataset (colors represent different user types: red: pedestrian, cart: green, biker: blue, skater: yellow, x and y axis are relative image coordinates)

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 varying and adjusting facility opening cost *f*, waiting time *w*, and batch size *h*. We also compare the algorithm against two other algorithms, i.e. Meyerson's algorithm and the original framework from Cohen-Addad. 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 the lowest cost.

Three aspects are considered as criteria, namely calculation time for the whole dataset, average cost per update, and the total number of centers that ever opened. The smaller the value of these criteria, the better the effect of the algorithm.

Moreover, we calculated three intuitive indicators, namely average group size, maximum group size, and the space-saving rate for a better understanding of the results. Average group size calculates the average number of members per group, the closer this value is to 1, the worse the team effect is; the maximum group size is the number of members of the largest group; space-saving rate is defined as the reduction in size relative to the ungrouped size, the larger this rate, the more the users in the group.

3.1 Parameters and results

The facility opening cost *f* restricts the distance between two road users. A 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 (image coordinates, roughly corresponding to 4 m in reality). To group the road users from the same lane, f is set to 200, which allows a maximum distance between starting and endpoints of 100, each.

Waiting time *w* decides how long users wait for others to form a group. In Tab. 1, batch size *h* is set to 30, the value of *w* is varied from 5s, 30s, to 60s, respectively to check the influence on the formed groups. From the intuitive indicators, we can see that higher values of *w* can form fewer groups with more group members, which leads to a higher cost and computation time, but fewer opened centers.

w [s]	average group size / maximum group size / space-saving rate	# centers ever opened	calcula-tion time [s]	average cost per update
5	1.22 / 4 / 15%	517	0.43	2099.9
30	1.87 / 21 / 43%	320	0.62	10519.8
60	2.27 / 26 / 55%	314	0.74	17367.5

Table 1: Effect of waiting time *w* (*f*=200 and *h*=30)

The batch size *h* determines the number of users to be selected for recomputation between two periods. For large or adversary dataset which has complex ground truth, *h* should be large enough to avoid bias. In an extreme case, i.e. *h* = 1, there will be no coarse solution before each period, which means the road user who comes first will become the center of the group. It can bring benefits to time efficiency and average cost, but the recourse will increase sharply. Typically, *h* is also related to the layout of the scene, e.g. shape of the lanes and potential access points to it. For instance, in the DeathCircle, most road users follow the shape of the lanes or exit from buildings, thus the entering and exiting locations are rather static, which can help to decide the value of *h*. At Tab. 2, we fixed *f* to 200 and *w* to 30 seconds, as they hold group size with a relatively small number of opened centers and average update cost. As a result, the average group size and space-saving rate appeared to be unaffected by the value of *h*. However, as expected before, the larger the value of *h*, the more effort is needed to calculate a coarse solution, which leads to higher calculation time and update cost, but less recourse.

h	average group size / maximum group size / space-saving rate	# centers ever opened	calcula-tion time [s]	average cost per update
10	1.85 / 11 / 44%	416	0.32	4271.8
20	2.02 / 17 / 49%	363	0.53	7787.6
30	1.88 / 19 / 43%	320	0.88	10566.3

Table 2: Effect of batch size *h* (*f*=200 and *w*=30)

Considering the results and discussion above, the parameters should be adjusted according to different needs of road users and urban traffic planners, e.g. how long should people wait (*w*)? How far are they willing to go to become a group (*f*)? Is my dataset adversary (*h*)? Therefore, we selected a reasonable parameter set *f*, *w* and *h* as 200, 30 and 10. It results in a minimum of 13 groups and a maximum of 33 groups, and the largest group contains 11 members. The average group size is 1.85 and 44% space was saved by forming groups. These parameters are also utilized in subsection 3.3 to compare the results from different algorithms.

Fig. 2 shows the grouping results of one example period in DirthCircle. The trajectories with the same color belong to the same group. The colored crosses indicate the centers of corresponding groups. There are 11 groups, 8 of them contain only one road user, and the largest group (the blue trajectories from the left side of the figure) has 5 members.

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, after testing multiple sets of parameters, we set *f* to 300, and *w* to 23.6 seconds to make the average group size over 2. In other words, a road user will move a maximum of 6m and wait for 23.6s to form a group with the others.

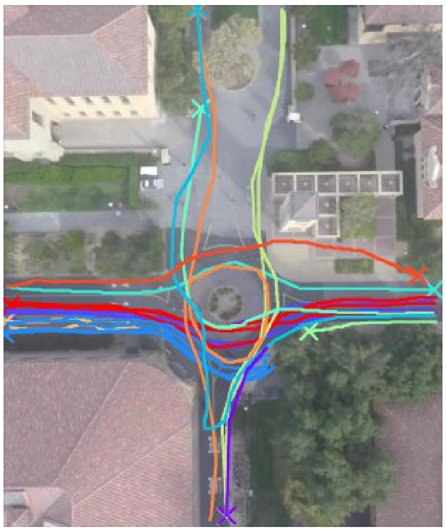

Figure 2: Grouping result of one period (21 users and 11 groups), the largest group has 5 members

3.2 Comparision with other algorithms

Using the parameters decided in subsection 3.2, we generate the plots for cost and running time per update against Meyerson's algorithm and the Cohen-Addad's algorithm, see Fig. 3.

The cost of Meyerson's algorithm has a linear dependency on the number of updates because it never deletes road users. This is also the advantage of our algorithm. By restricting 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 deletion conditions, but still keeps a similar cost per update.

Our algorithm has less running time than Cohen-Addad's, as the waiting time restricts the number of calculated points per update. The average improvement of the single update compared with Cohen-Addad's is about 35.3%. 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.

Considering the recourse, ours has a similar number of opened centers with Cohen-Addad's, and both are worse than Meyerson's, as it will never close the opened centers, new arrivals are getting more and more difficult to build a new center.

4. Discussion

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

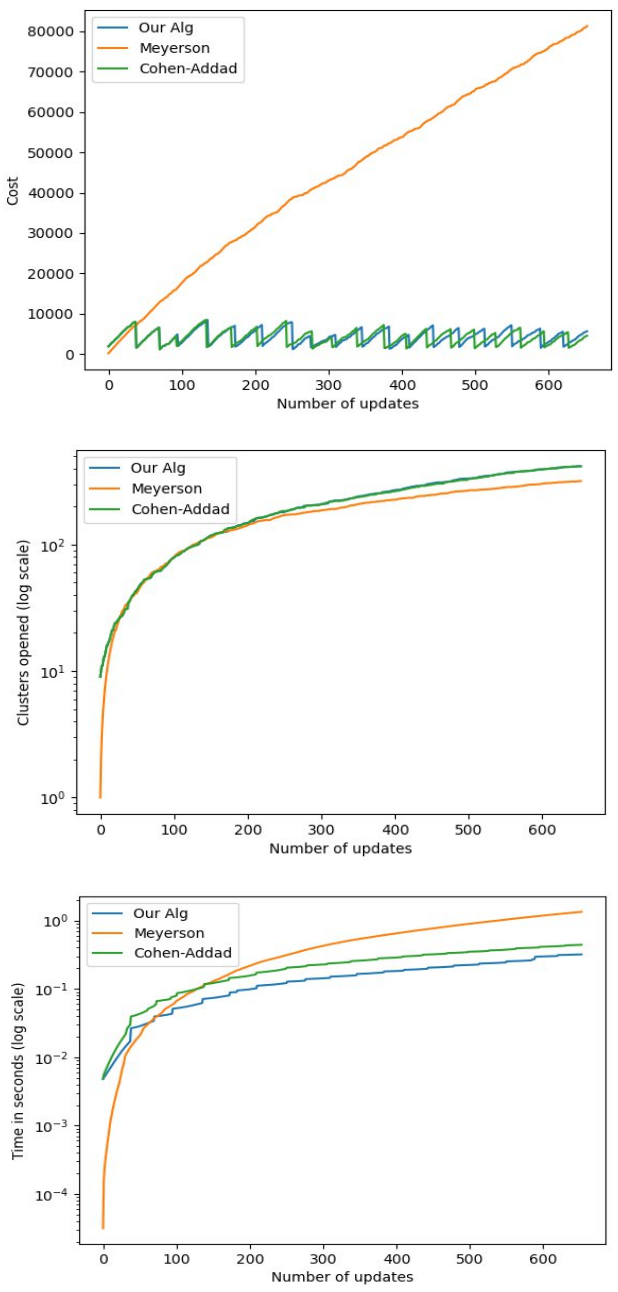

Figure 3: Comparision of cost, number of opened facilities and running time per update against Meyerson's algorithm and the Cohen-Addad's algorithm

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 need to be carefully considered according to traffic management studies.

For a sparse traffic scenario that has low traffic flow, it is not necessary to apply the method, because people need a trade-off between waiting time and forming group, and in a sparse dataset, there is no need for grouping, as conflicts between road users are less likely.

A possible extension of the approach is to also take into consideration the intermediate points of the road users in the group forming process. In the example of Fig. 4, we can replace the *ODsimilarity* with the symmetrical Hausdorff distance (Edwards, 1975) calculated from intermediate points of pairwise trajectories. In this way, dedicated, separate paths between the same origin and destination could be realized and thus adapt better to the plans of the users.

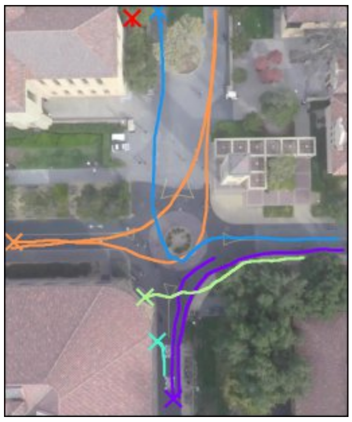

Figure 4: Benefit from Hausdorff distance (sperate the users who have similar OD but different waypoints)

One drawback of the method is that it cannot deal with the splitting and merging of groups. Recall the light blue and orange groups coming from the top direction in Fig. 2, which started from the same lane but separated after passing through the roundabout. This would involve to cluster also partial trajectories and is subject to future work.

5. Conclusion and future work

In this paper, we proposed a dynamic facility location method to cluster road users who have similar OD and entering time in a shared space. In future work, we will adopt this method for all kinds of road users by integrating average velocities to the distance metric and include trajectories to deal with the splitting and merging cases. Furthermore, we will investigate how additional constraints and requirements could be integrated into the approach, to be able to apply it for a traffic organization of a junction or a shared space. Among others, this relates to identifying persistent clusters over time or with a certain spatio-temporal pattern, which would lead to an adaptation of the identification of coarse clusters. Furthermore, the group formation has to be integrated into the traffic planning of the whole junction. Another investigation will deal with the aspect of partial clustering to allow groups to split and merge in order to maximize the benefits of groups.

Acknowledgements

The funding of this research by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 227198829/GRK1931 is gratefully acknowledged.

References

Aveni, A. F. "The Not-So-Lonely Crowd: Friendship Groups in Collective Behavior." Sociometry 40, no. 1 (1977): 96-99. Accessed May 12, 2021. doi:10.2307/3033551.

Chan, T. H., Guerqin, A., and Sozio, M.: 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: https://doi.org/10.1145/3178876.3186124

Charikar, M. and Guha, S.: "Improved combinatorial algorithms for the facility location and k-median problems," 40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039), 1999, pp. 378-388, 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.

Gupta, A., Krishnaswamy, R., Kumar, A., and Panigrahi, D.: Online and dynamic algorithms for set cover. In Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017). Association for Computing Machinery, New York, NY, USA, 537–550. DOI: https://doi.org/10.1145/3055399.3055493.

Hamilton-Baillie, B. (2008). Shared space: Reconciling people, places and traffic. Built environment, 34(2), 161-181. DOI: https://doi.org/10.2148/benv.34.2.161.

Holmes, L.: Accidents by Design: The Holmes Report on ”shared space” in the United Kingdom. (July):1, 2015.

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

Karndacharuk. A., Wilson D. J. and Dunn. R. (2014) A Review of the Evolution of Shared (Street) Space Concepts in Urban Environments, Transport Reviews, 34:2, 190-220, DOI: 10.1080/01441647.2014.893038.

Kranen, P., Assent, I., and Baldauf, C.: 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.

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.

Monderman, H., Clarke, E., & Baillie, B.H. (2006). Shared space - the alternative approach to calming traffic. Traffic engineering and control, 47, 290-292.

Moody, S., & Melia, S. (2014). Shared space research, policy and problems. In Proceedings of the Institution of Civil Engineers Transport (Vol. 167, No. 6, pp. 384-392). Thomas Telford Ltd. https://doi.org/10.1680/tran.12.00047.

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

Qadri, S. S. S. M., Gökçe, M. A. and 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 Project_Number: 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 crowded scenes. In Lecture Notes in Computer Science, volume 9912 LNCS, pages 549–565, 2016. ISBN 9783319464831. doi: 10.1007/978-3-319-46484-833.

Space, S. (2005). Room for Everyone: A new vision for public spaces. The Netherlands: Leeuwarden.

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.