Bixi Bike
Complex Network Analysis

as part of
the course

MATH80627A
(Complex Networks Analysis)

presented to
Gilles Caporossi

by

Mohammad Abbas Meghani - 11266035
Adrien Hernandez - 11269225
Andreea Firanescu – 11274520

1. Introduction

As they have grown, bicycle-sharing networks all over the world have been the subject of extensive analysis in the past 10 years, and have seen a growing body of literature develop around them (Froelich et al., 2009; Lathia et al., 2012; Zaltz et al., 2013). Related research can be classified into three main streams, that is, i) offering models of bicycle flows to transport operators who can optimize their stations by uncovering spatiotemporal trends in bicycle usage and city data ii) servicing end-users in real-time planning of trips, and iii) supporting urban planning (Lathia et al., 2012, p. 90). The analysis of bicycle usage is indeed interesting as it partly corresponds to human movement and cultural and geographic aspects of a city. It thus helps uncover spatiotemporal trends that can partly infer the "pulse" of a city, as what was put forward by Froelich et al., the first to do so (2009), and particularly appealing to researchers who inhabit a city and can therefore better contextualize the insights gathered from their analyses.

The present project builds on the current literature by deriving insights from spatiotemporal bicycle-sharing data in Montreal. Specifically, it seeks to identify spatiotemporal clusters via community detection. The subject of community detection within a bicycle-sharing network is interesting because of i) the potential applications and the growing literature developing around it (as indicated above), ii) the ease of access to the data and the variety of community detection algorithms that can be applied to it, and iii) the use of spatiotemporal data for the application of new algorithms such as network science ones applied to time series. The approach to this project was an exploratory one as the main goal of the present project was to explore and become acquainted with community detection, one of the most important fields in network science.

2. Exploratory Data Analysis

The second reason for the selection of our topic was the ease of access and manipulation of the data. Two options in terms of data presented themselves at the start of the project, that is, using sensor data collected by the city of Montreal and bicycle usage data by BIXI. While the BIXI data offered the time and the locations of the start and end points of a journey, the city of Montreal has sensors distributed throughout the city that can help trace bicycles. Combining both datasets would have however made it difficult to consider the flows and journeys of users, an issue for researchers when studying bicycle usage networks. Using solely sparse raw sensor readings was not an option as these lack the qualitative description aspect about the context of human mobility (Lathia et al., 2012, p. 92). While using solely the BIXI dataset can have its limitations in terms of considering full journeys, it is easy dataset to build a network from and offers many possibilities in terms of algorithms that can be applied.

BIXI offers open access to its data in the form of csv files containing geolocation information about its stations (table 1) and spatial and temporal data about journeys (table 2). Each entry corresponding to an individual journey, the network built from it is spatial-temporal, and weighted. Therefore, nodes represent bike location stands, edges are bi-directed flows between two such stands, and the weight on each edge is given by the number of rides carried on edges. Excerpts from the data are shown below in the shape of pandas arrays.

	code	name	latitude	longitude
636	8046	Mairie d'arrondissement Montréal-Nord (Hébert ...	45.593945	-73.638444
637	8071	Gare d'autocars de Montréal (Berri / Ontario)	45.516926	-73.564257
638	8036	Messier / St-Joseph	45.539461	-73.576056
639	8069	Ateliers municipaux de St-Laurent (Cavendish /...	45.506176	-73.711186
640	8022	Centre des loisirs (Tassé / Grenet)	45.514734	-73.691449

Table 1. 2020 BIXI stations (tail)

	start_date	start_station_code	end_date	end_station_code	duration_sec	is_member
3264736	2020-11-16 04:49:38	7073	2020-11-16 04:56:23	6209	404	1
3264737	2020-11-16 04:51:27	6154	2020-11-16 04:55:17	7035	230	1
3264738	2020-11-16 04:53:24	6108	2020-11-16 05:02:51	6017	566	1
3264739	2020-11-16 04:55:21	6104	2020-11-16 05:05:05	6119	584	1
3264740	2020-11-16 04:55:37	6148	2020-11-16 05:13:17	6370	1059	1

Table 2. 2020 BIXI journeys (tail)

There are 619 stations across Montreal, which is a large number for its population size compared to other cities (Zaltz et al., 2013). In 2019, around 5.5 million rides were registered. Some of these are self-loops, single-edge journeys which start and end on the same node. These were set to 0, as is accepted practice for community detection in the study of bicycle-sharing networks (Zaltz et al., 2013, p.2).

To visualize the data and to start identifying strong patterns, we did a preliminary spatiotemporal data exploration. The months present in the data are April to October as these are the only months the bicycles are in use. As would be expected might be explained by Montreal climate, usage is the lowest in April and peaks in the middle of the summer (figure 1). Interestingly, when trips are aggregated, usage is higher during the week than on the weekend, which could infer

higher usage of BIXI by commuters (for work use) rather than leisure users, which is consistent with findings for other North American cities (figure 2) (Zaltz et al., 2013). Indeed, BIXI usage is highest during rush hour, between 8am and 9am and between 4pm and 7pm (figure 3).

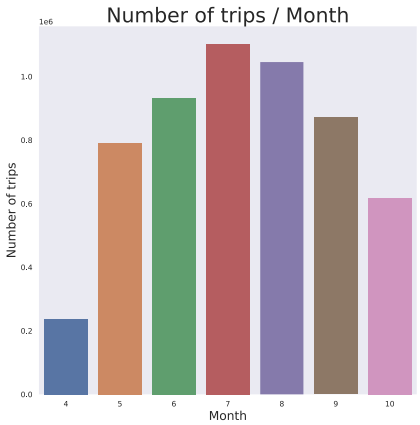


Figure 1

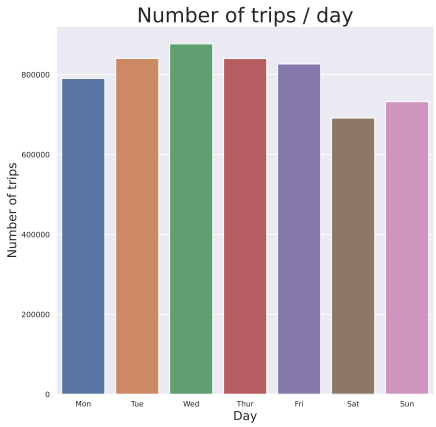


Figure 2

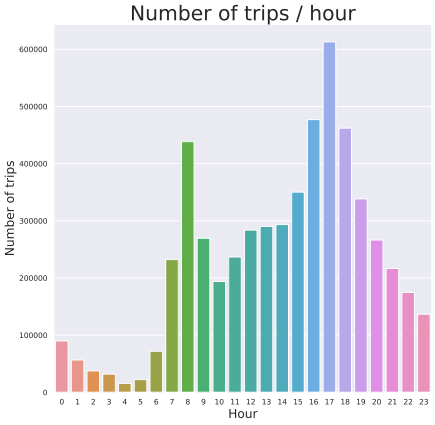


Figure 3

The data for the Montreal network is in keeping with previous studies, such as that of Zaltz et al. (2013), who have also conducted community detection in various cities, which helped us

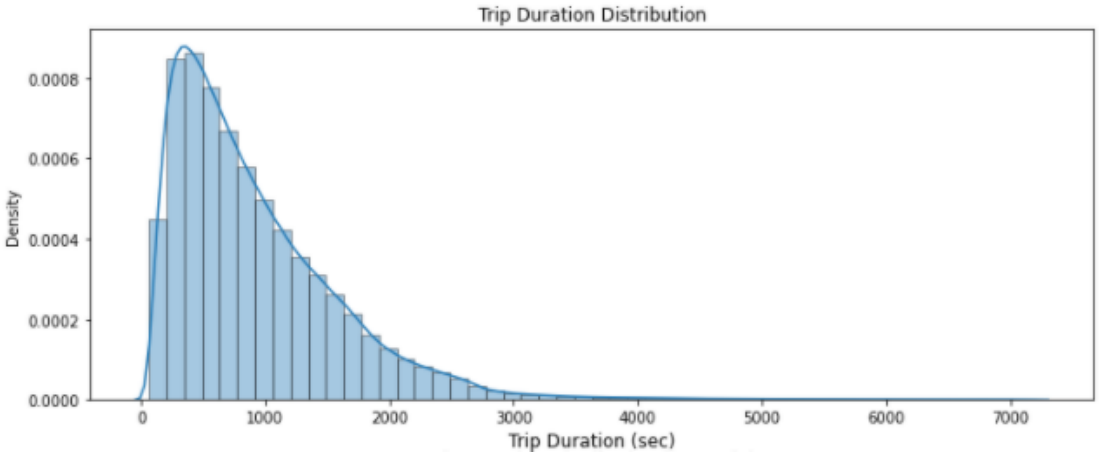


Figure 4. Montreal trip duration (data normalized by area under the curve)

validate our findings further on. As such, we have found similar trip duration as what they have found for the city of Washington D.C (figures 4 and 5).

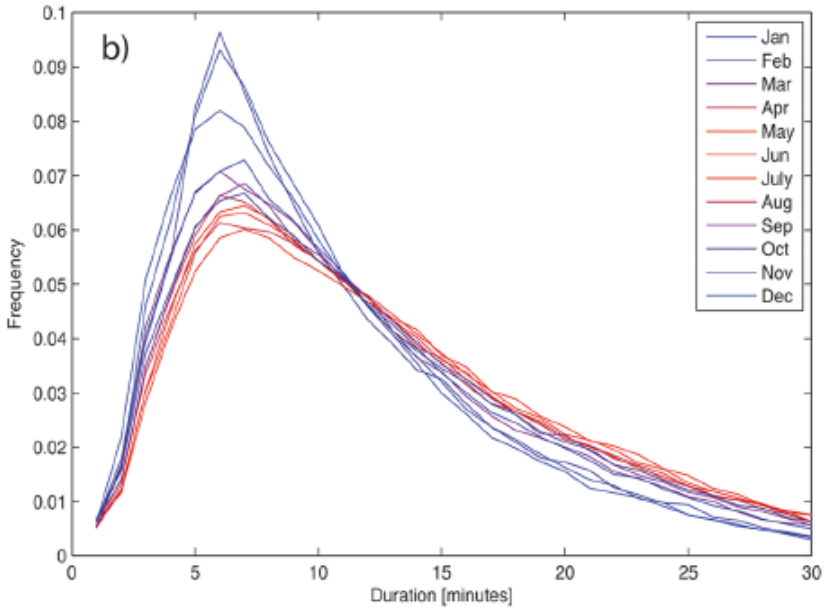


Figure 5. Washington D.C. trip duration (data normalized by area under the curve), from Zaltz et al., 2013, p. 4

In order to better understand the specifics of the data, we start by applying node descriptors using Python's NetworkX package, which give us an insight into the role of a node in a network. An important descriptor is centrality which considers the number of neighbours. When computing degree centrality (figure 6), which indicates the fraction of nodes one node is connected to, on part of the network (for computational time reasons), the highest centrality was associated to nodes in the center of the city, which is to be expected. Interestingly, the node with the highest centrality, the one situated at BANQ (Berri/de Maisonneuve) also matches the location of Montreal's busiest metro station, Berri-UQAM. The next two nodes with the highest centrality were situated in the neighbourhood of Plateau Mont-Royal and the other next two in Downtown:

Top 5:
BAnQ (Berri / de Maisonneuve) : 0.95146
Métro St-Laurent (de Maisonneuve / St-Laurent) : 0.92718
Métro Mont-Royal (Rivard / du Mont-Royal) : 0.91748
du Président-Kennedy / Jeanne-Mance : 0.91586
de Maisonneuve / Aylmer (ouest) : 0.91424
Bottom 5:
Argyle / de Verdun : 0.35275
Métro Monk (Allard / Beaulieu) : 0.39644
de l'Église / Bannantyne : 0.42557
Place du Commerce : 0.43204
Métro Georges-Vanier (St-Antoine / Canning) : 0.47896

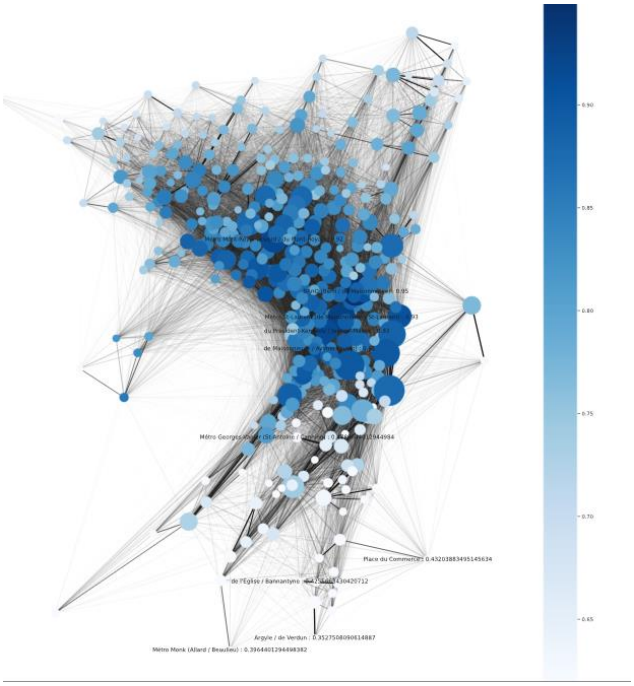


Figure 6. Degree centrality

Different centrality measures were computed, and visual representations were made to be able to visualise the topography and identify important nodes. Most global descriptors offered similar findings: harmonic centrality and closeness centrality, had the same results for the nodes with the highest centrality, and the eigenvector centrality descriptor (which can also be considered as a local descriptor), that helps identify influencers, interchanged only the 4th and the 5th nodes. However, another global descriptor, betweenness centrality, offered very different results, finding nodes in the Old Port to be more central (figure 7). Indeed, these nodes would be the ones with the

most shortest paths going through them. This is an interesting finding as it can be indicative of bridging communities, but results are not always reliable in that sense. Load centrality, which is the fraction of all shortest paths that pass through that node, also found the same top five most central nodes to be in the same vicinity.

Figure 7. Betweenness centrality

The measure with the most distinct and perhaps unexpected results was Katz centrality, placing some relatively central nodes in the bottom 5:

Top 5:
Métro Lionel-Groulx (Atwater / Lionel-Groulx) : 0.11945
Métro Papineau (Cartier / Ste-Catherine) : 0.06767
Ottawa / Peel : 0.06616
BAnQ (Berri / de Maisonneuve) : 0.05689
Chabot / du Mont-Royal : 0.05176

Bottom 5:
St-Urbain / de la Gauchetière : -0.06351
Metro Jean-Drapeau (Chemin Macdonald) : -0.06089
Milton / du Parc : -0.05292
St-André / Ste-Catherine : -0.04895
Villeneuve / St-Laurent : -0.04858

An issue encountered with the data is that it indicated flows in terms of origin and destination but did not trace the full journey of the rider, a fairly common issue, as mentioned above. To visualize the journeys, we then used Dijkstra's shortest path algorithm to identify potential routes. We imported from OpenStreetMap the Montreal network and used it to identify which closest road intersection would correspond to a node. Then, based on the route of each

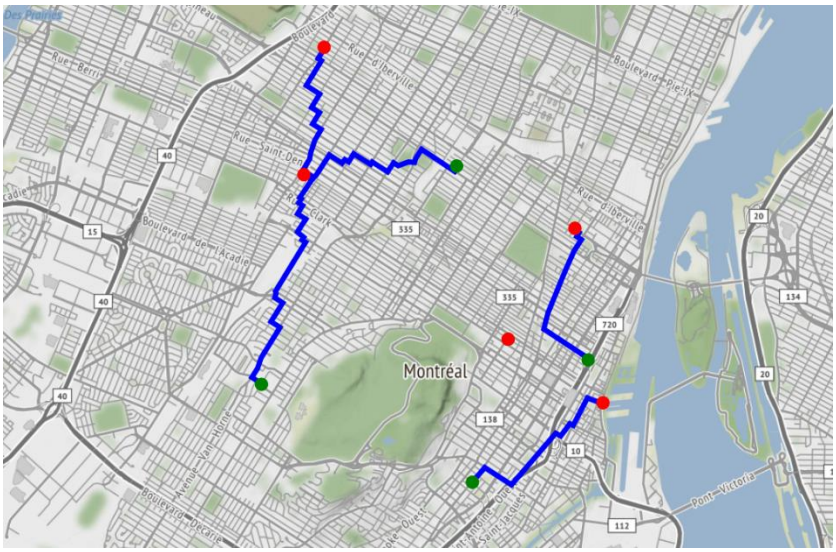


Figure 8. Djistra's shortest path between BIXI stations

section, the shortest path was found. While this might not represent true journeys, there are other potential applications, such as identifying next nodes where to place a station.

3. Methodology

3.1. Community Detection Algorithms

3.1.1. Greedy Modularity

The first community detection we tried was greedy modularity proposed by Newman in 2004 as an agglomerative hierarchical clustering method (Newman, 2004). The algorithm is interesting to try as a first one as it is a simple heuristic meant to find partitions with high modularity with reasonable computational time. It is also the first greedy algorithm proposed (Li et al., 2020) and results can be compared further on with those of more sophisticated algorithms. It initializes by having each node belong to its own community then repeatedly merges pairs of communities together and chooses the merger for maximizing modularity. Modularity maximization has two opposite yet coexisting issues; it can split true communities into smaller clusters and in other cases, identify large clusters from what would be smaller communities (Chen et al., 2014).

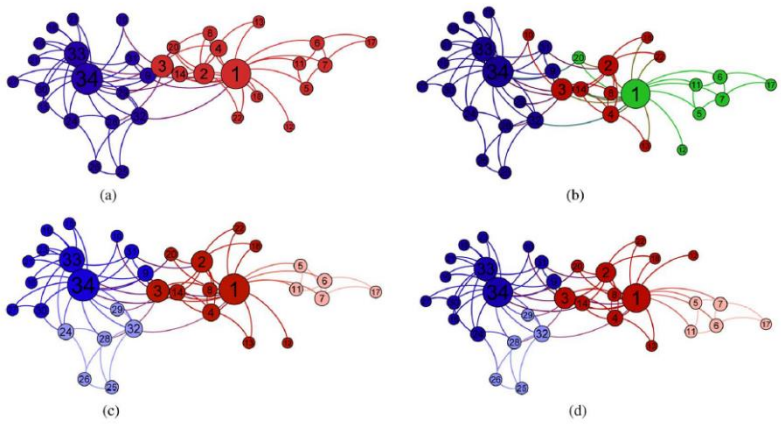


Figure 9. Misidentified communities by greedy modularity (Chen et al., 2014, p.59)

3.1.2. Louvain

Another greedy modularity optimization we tried was the Louvain algorithm as it is one of the most popular community detection algorithms, described as “simple and elegant” (Traag et al., 2019, p. 2). The Louvain algorithm works in two steps. The first one, similarly to the previous algorithm, sees each node at first be its own community, and be moved from one community to another in a way which maximizes modularity. The step creates an aggregate network, where each community from the local moving phase becomes a node in the network. The steps are then repeated. This algorithm is also interesting as the number of iterations is usually small and the first iteration takes most computation time. It is therefore easily scalable to large networks. However, the final result is highly impacted by the order in which the nodes are merged in the first step (Traag et al., 2019).

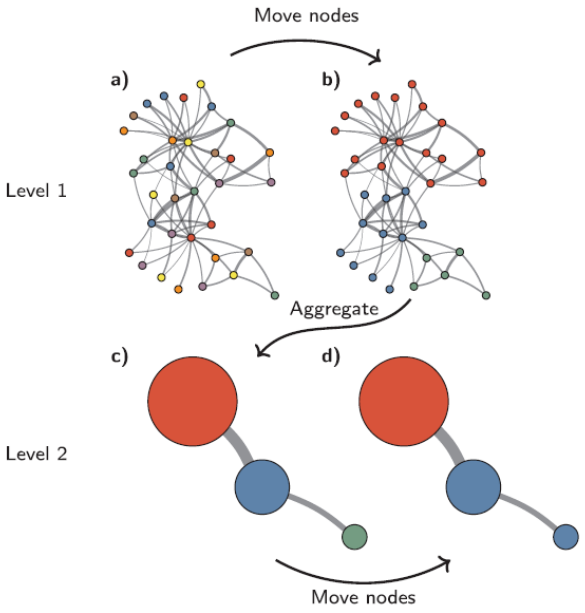


Figure 10. Louvain algorithm (Traag et al., 2019, p.2)

3.1.3. FluidC

The next algorithm we decided to try was a new type of Label Propagation Algorithm, namely, Asynchronous Fluid Communities. The algorithm presented interest as it offers the advantages of LPAs in terms of low computational cost and scalability yet presents a novel approach to community detection. Additionally, it has been integrated into the NetworkX package. It presents significantly different results to other methods and it is the first LPA to identify a variable number of communities in a network. It was first introduced in Barcelona by Parés et al. in 2018 with the objective of providing high quality partitioning of communities for large networks. They based themselves on the "idea of fluids interacting in an environment, expanding and contracting as a result of that interaction" (Parés et al., 2018, p.1).

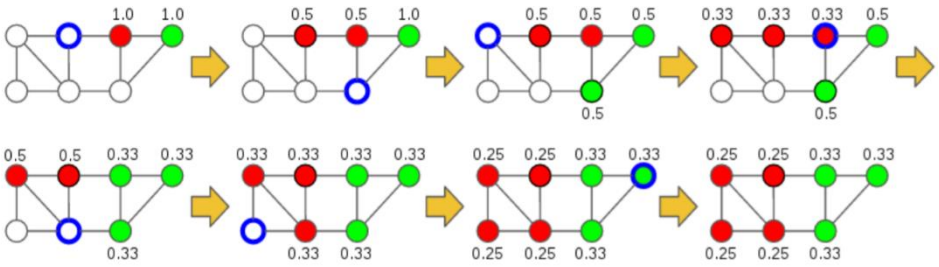


Figure 11. Fluid Communities algorithm with *k*=2 (Parés et al., 2018, p.3)

The FluidC algorithm proceeds as follows: first, each of the initial *k* communities is initialized in a random vertex *v* in the graph. Then, using an update rule (equation 1) the algorithm iterates over all vertices in a random order, updating the community of each vertex based on its own community and the communities of its neighbours, returning the communities with maximum aggregated density. This process is performed several times until convergence. At all times, each community has a total density of 1, which is equally distributed among the vertices it contains. If

a vertex changes of community, vertex densities of affected communities are adjusted immediately. When a complete iteration over all vertices is done, such that no vertex changes the community it belongs to, the algorithm has converged and returns the final communities.

$$\mathscr{C}_v' = argmax_{c \in \mathscr{C}} \sum_{w \in \{v, \Gamma(v)\}} d(c) \times \delta(c(w), c)$$

$$\delta(c(w), c) = \begin{cases} 1 \ , if \ c(w) = c \\ 0 \ , if \ c(w) \neq c \end{cases}$$

Equation 1. Update rule of FluidC (Parés et al., 2018, p.3)

FluidC is asynchronous, meaning that it is possible that some vertices update their labels and others do not. This ensures that a community does not lose all its vertices and be removed, which would happen if the algorithm were synchronous, and a community would lose a vertex, but density would not be increased immediately. Conversely, FluidC does not permit the creation of monster communities as large communities can only keep its size by having many intra-community edges keeping its density lower.

3.2. Time Series Networks

We found with our exploratory data analysis and community detection algorithms that there is an effect of time on the demand of bikes in the complex network. This gave us another appealing direction to analyze and explore our data and use methods yet unknown to us, which is building networks using time series. We used two main methods which aim to apply network science to temporal data mining. To convert time series data into a network, we first used a visibility graph and then time series clustering.

3.2.1. Visibility Algorithm

The visibility algorithm, as presented in by a fellow classmate, was introduced by Spanish researchers in 2008 as a simple and computationally fast method of mapping a time series into a network. This algorithm sees time series data converted into a network is by using a test of visibility. Each bar in the histogram is a series data representing a demand, which then becomes a node in the network. The edges are connected to the demand or nodes which can be directly observed without obstruction.

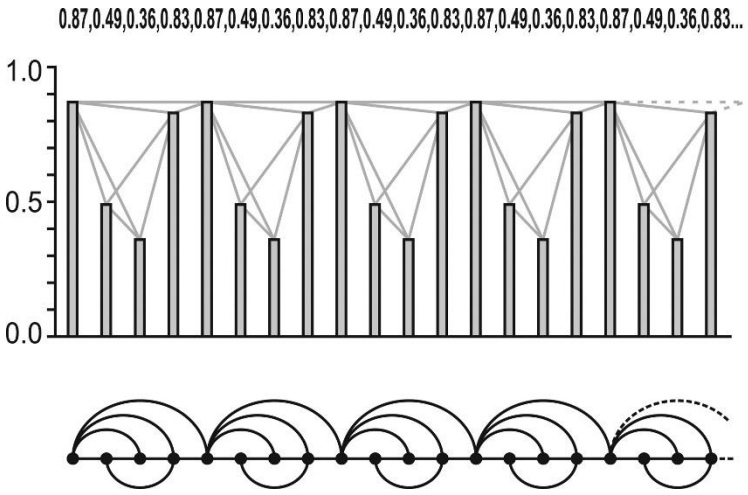


Figure 12. Example of a time series and the associated visibility graph (Lacasa et al., 2008, p.9)

The method is compelling as it considers the time series as a landscape. In the resulting graph, two nodes are connected if there is “visibility”, or a straight line connecting the series data, if this line does not intersect any intermediate data height, following the equation:

$$y_c < y_b + (y_a - y_b)\frac{t_b - t_c}{t_b - t_a}$$

Equation 2. Visibility criterium (Lacasa et al., 2008, p.2)

The resulting network is connected, undirected and invariant under transformations. Indeed, apart from being computationally favourable, this method presents the advantage of being retains its features against transformations. Indeed, the constructed graph inherits several properties of the series, so that periodic series convert into regular networks, random series into random networks, and fractal series into scale free networks.

3.2.2. Time series clustering

Another method was presented by Ferreira and Zhao in 2016 using time series clustering. This method is of particular interest to us as it applies community detection techniques for time series clustering, which had not been reported in the literature before. The main objective for the authors' proposed method was the "transformation of time series from time-space domain to topological domain" (Ferreira & Zhao, 2016, p.227). It uses the topological structure of the underlying network that is constructed during its clustering process to detect shape patterns in the time series.

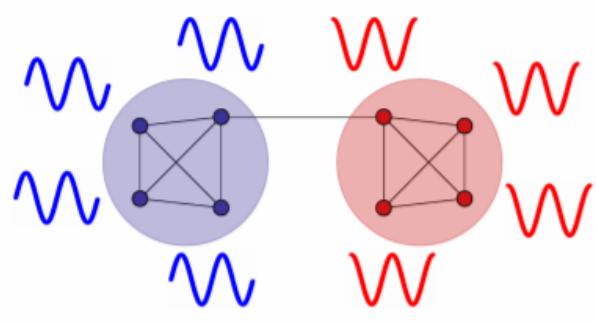


Figure 13. Various time series clustered into two communities (Ferreira & Zhao, 2016, p.235)

This method is unique in that each node is a time series, connected to others, which are then used to form clusters, with the goal of dividing the set of time series into clusters of similar ones. The algorithm starts on a dataset of time series with a data normalization process, then applies a distance measure or similarity metric to obtain the differences between time series. The network is then constructed using either a minimum threshold of similarity $\mathcal{E}$ between two networks (which is different for different metrics) or a predefined number of *k* nearest networks. After the network is constructed, several community detection algorithms can be applied, giving us insight into the time series similarities.

Algorithm 1: Time series clustering

```
input: dataset, k or ε
begin
    normalization(dataset);
    D ← distanceMatrix(dataset);
    G ← netConstruction(D, k or ε );
    C ← communityDetection(G);
end
```

Figure 14. Time series clustering algorithm (Ferreira & Zhao, 2016, p.236)

4. Results

4.1. Community detection

The package NetworkX was used in Python to construct an aggregate graph with journeys form all months from the year 2019 and a separate graph for each month, then to draw a preliminary graph of the nodes. It was then used to implement greedy modularity on the aggregate graph, clearly identifying two large communities, separating the city between North-West and South-East. As previously discussed, this algorithm poses the risk of merging real communities smaller than a certain threshold to form large clusters.

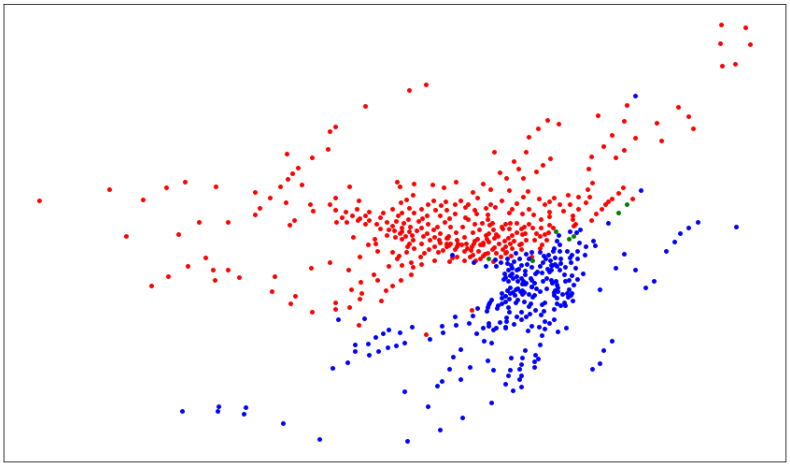


Figure 15. 2019 communities identified through greedy modularity

When applying asynchronous fluid algorithm from NetworkX, we then sought to identify more meaningful communities and set *k* larger to 2. Such meaningful communities were identified when $k = 5$, even accounting for the randomness of different seeds.

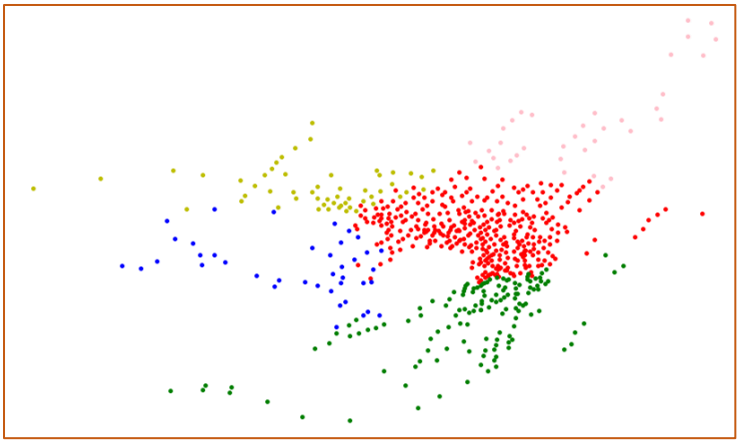


Figure 16. 2019 communities identified through FluidC

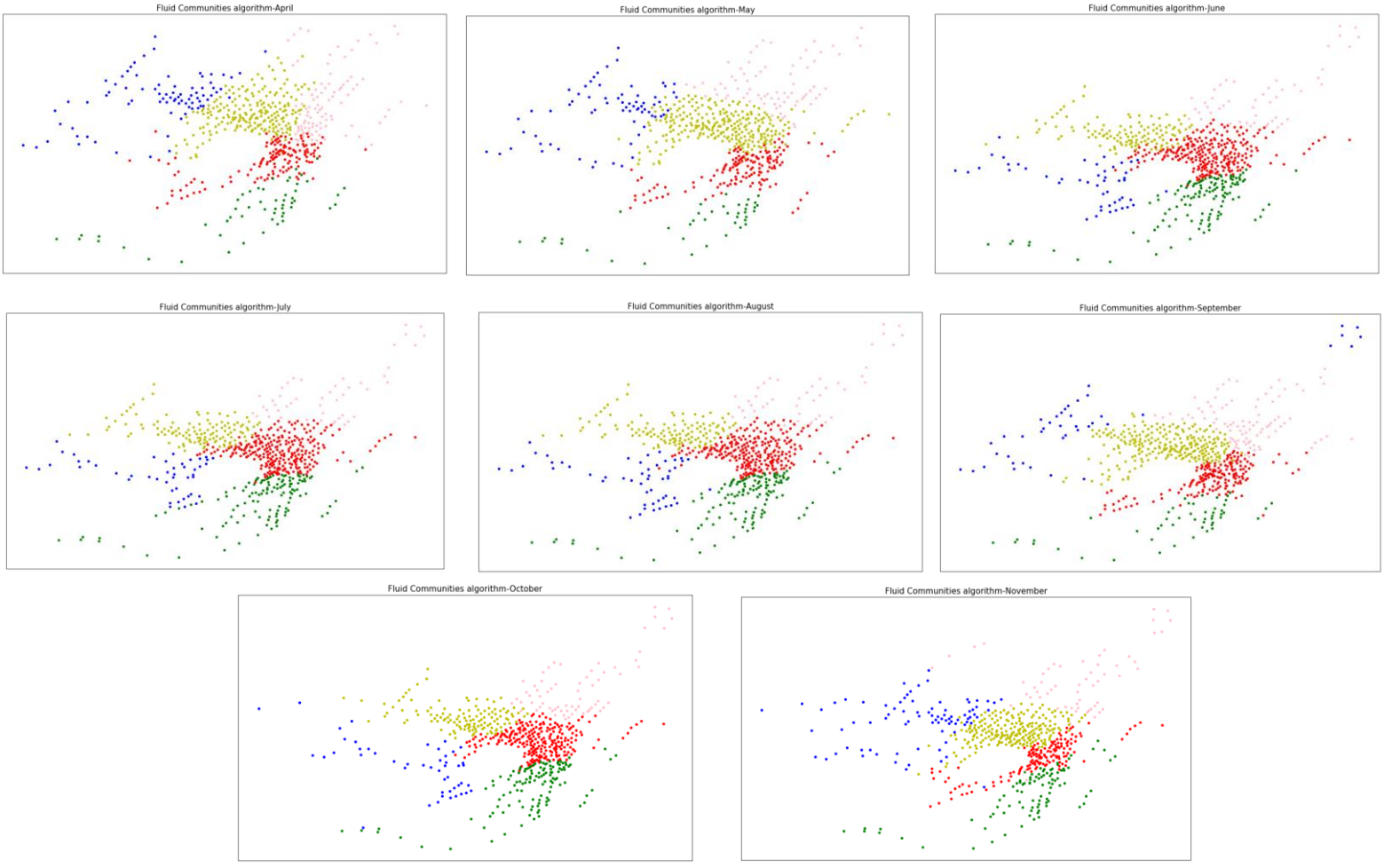


Figure 17. 2019 communities identified through FluidC/month

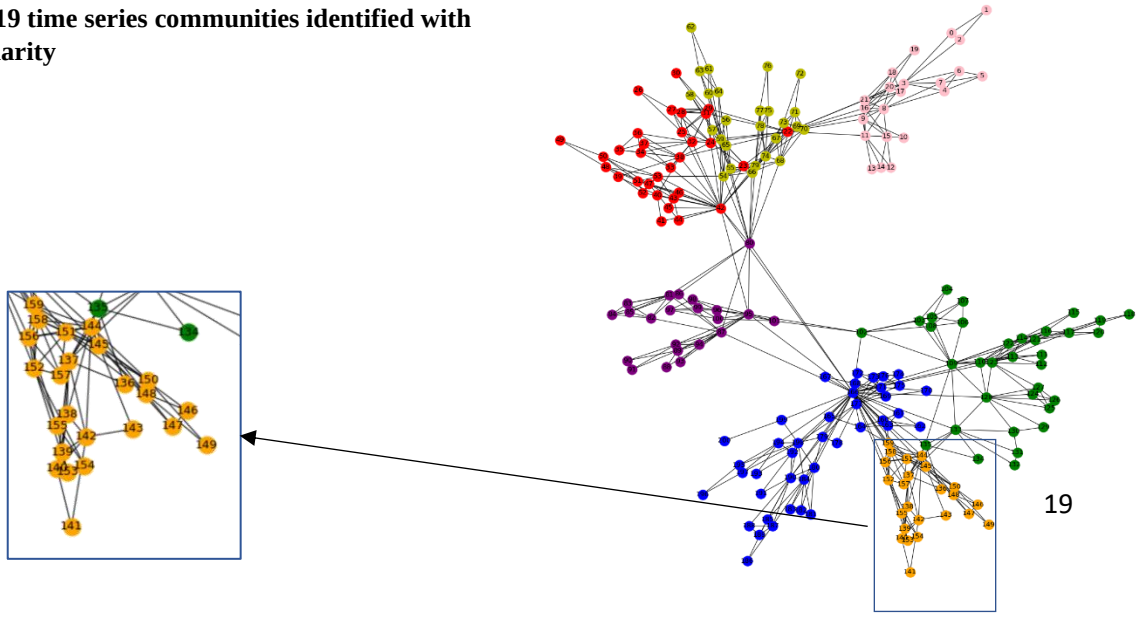
An interesting aspect of being able to set the number of clusters and the seed is applying the algorithm to different months and interpret recognizable patterns. Figure 16 indicates some communities expanding over the summer months with a more balanced community distribution in April and October.

4.2. Time Series Analysis

In order to create time series, we had to convert the dataset to a time series one. The data was originally in an edge list format, where each row of the data represents one bicycle journey. To convert this to a time series dataset, rides had to be aggregated for every day, that is, we calculated the number of rows (or rides) for each day and presented them as demand. This results in a list of time series bicycle demand, which can be used to create networks.

For the visibility graph, we used the daily demand data for the year 2019, that is 201 days and hence 201 nodes which comes to about 6 and a half months. The data is not normalized as the method is invariant to transformation. We then applied community detection algorithms to this network. First using greedy modularity, we find that the days closer together are more often in a group, which might reduce the amount of information we can get from such a graph.

Figure 18. 2019 time series communities identified with greedy modularity



When using the FluidC algorithms, this issue was exacerbated, where communities had all closed nodes or days clustered together. A better community detection algorithm was indicated.

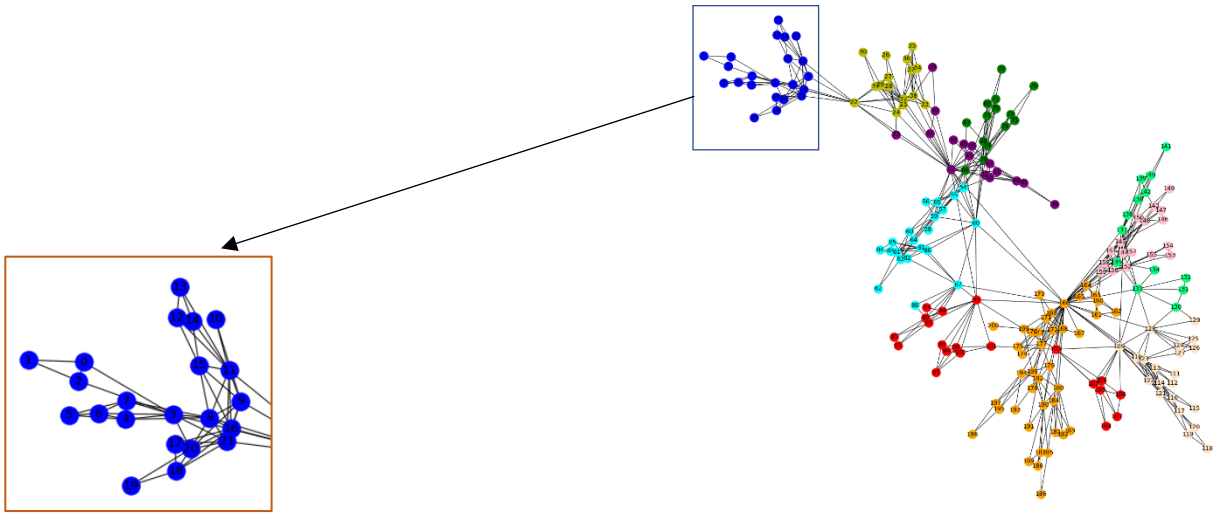
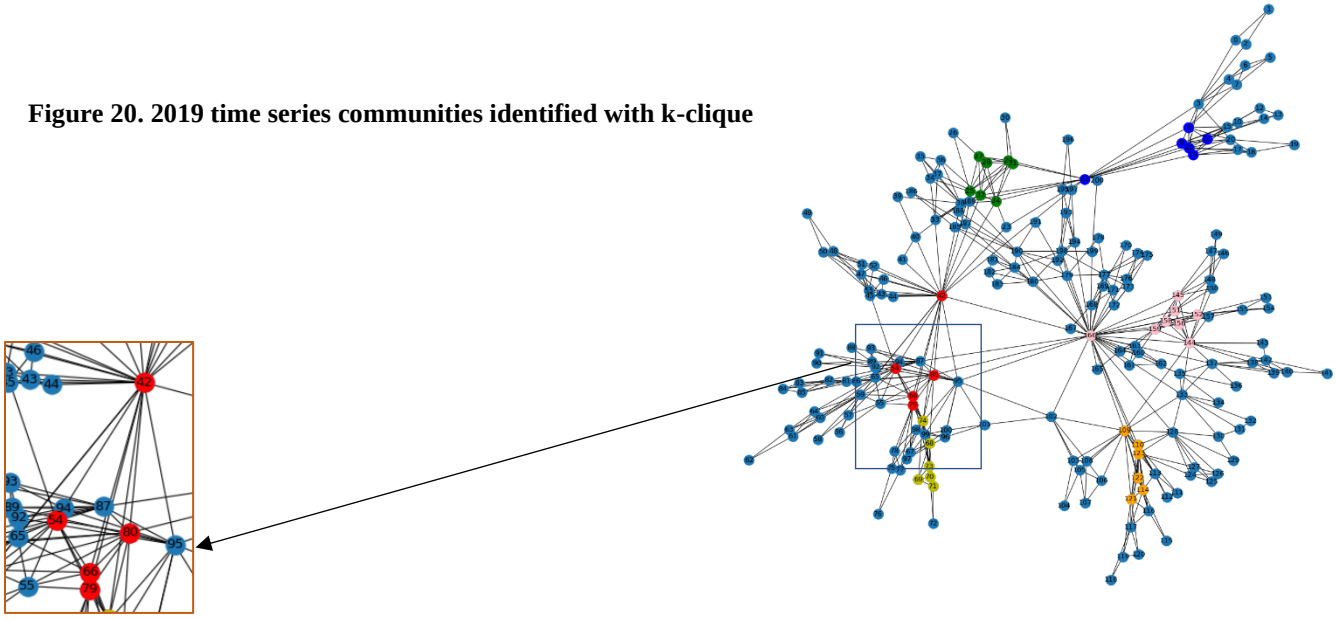


Figure 19. 2019 time series communities identified with FluidC

We then looked at the use of k-clique communities, which gave us nodes forming cliques of size *k* with other nodes. $K = 5$ was selected here. The communities were more indicative of the importance or similarity of these days.

Figure 20. 2019 time series communities identified with k-clique



To obtain more insight, we used centrality measures to see how the different days stand within them and list the days with the top five centrality measures. This diagram shows us that the same two days have the highest measures for all measures, but the days vary for different methods from the third highest position. Each of these might give a different insight into special user behaviour of said days.

	Closeness	Betweenness	Degree	Eigenvector	Harmonic	Information	Katz	Load
1	167	167	167	167	167	167	167	167
2	43	43	43	43	43	43	43	43
3	81	23	110	145	81	81	145	23
4	96	81	145	160	96	96	110	81
5	88	96	23	110	110	145	81	96

Table 3. Days with highest centrality / centrality measure

For the time series clustering method, we first used the demand for the months for the years 2019 and 2020 and separated them into different series. Then, a dynamic time warping distance and a pearson's correlation distance was used to calculate the 5 closes edges. In pearson's case, it was 4 as each node is perfectly correlated with itself. We then tried a greedy modularity and a Louvain algorithm. Both of them had similar results in terms of clusters. We tried different types of distance results as they might have given different insights into the data. However, the results indicate similarity between different months of the demand for BIXI bikes. We particularly appreciated dynamic time warping as it had the ability to take into account delay in patterns

between series. In the paper, time series data was simulated with a specified class. A RAND index was used to measure accuracy. In our case we did not have information about classes, so we decided on a more of an unsupervised approach.

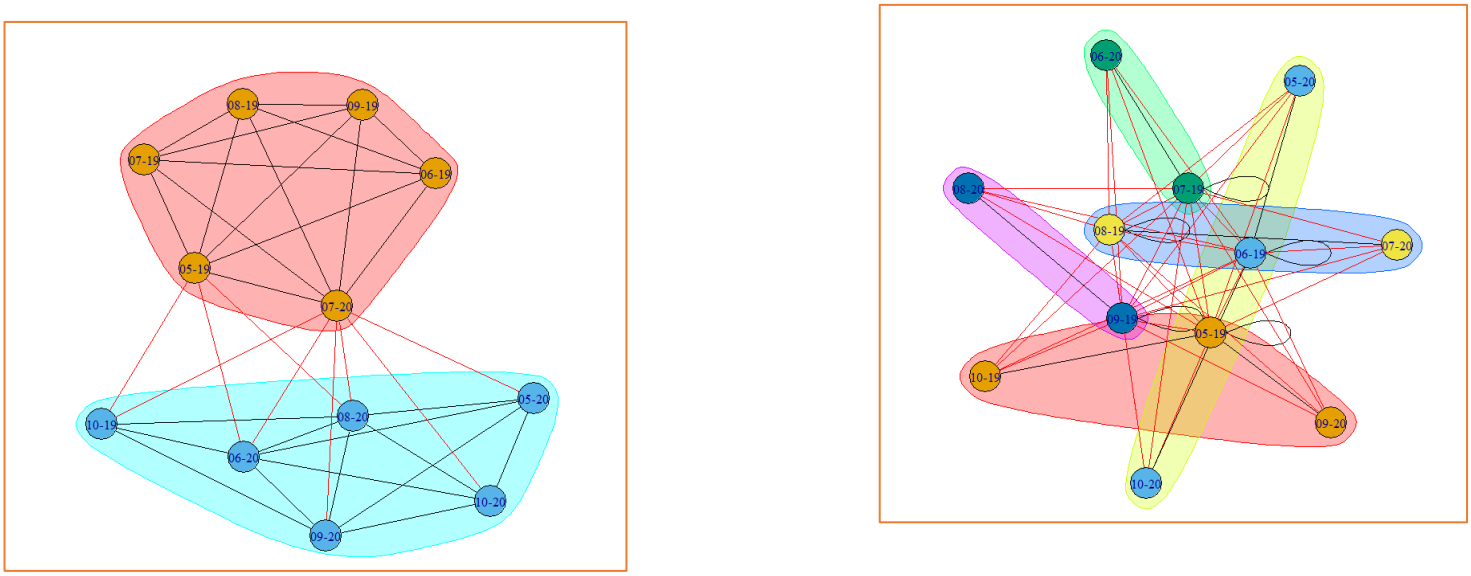


Figure 21. Month time series clustering via CD using dynamic time warping (left) and Pearson's Correlation (right)

We also used weeks for the time series nodes with the 2019 bixi data, with 27 weeks in total. This gives us insight into similarity between different weeks.

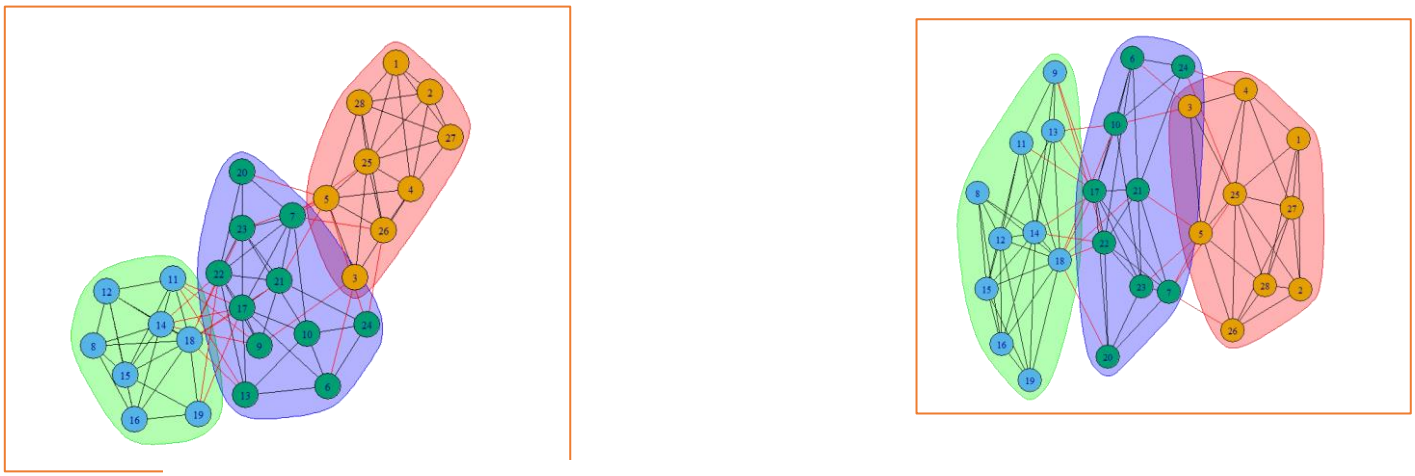


Figure 22. Week time series clustering via CD using dynamic time warping (left) and Manhattan (right)

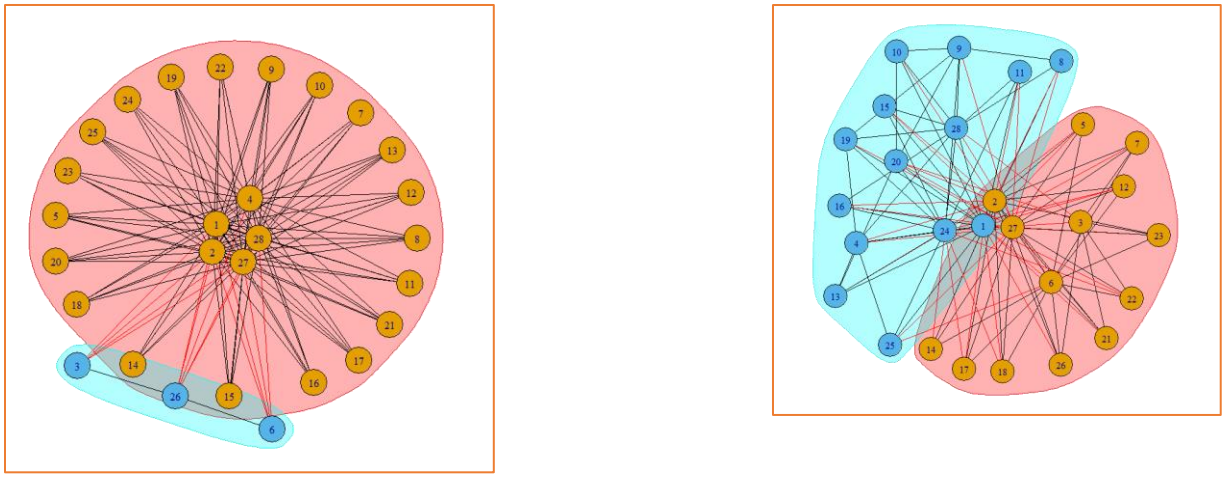


Figure 23. Week time series clustering via CD using Harmonic mean (left) and Cosine (right)

For further exploration, we also decided to make a time series data for each of the BIXI stations for 201 days of the service operation in 2019. The time series nodes represent each bike station and are labelled with their official codes. We only kept those stations which had at least one ride taken from it every day. We then used the DTW to obtain edges and applied a Louvain community detection algorithm. This plot makes it possible to compare different stations using time series complex network clustering.

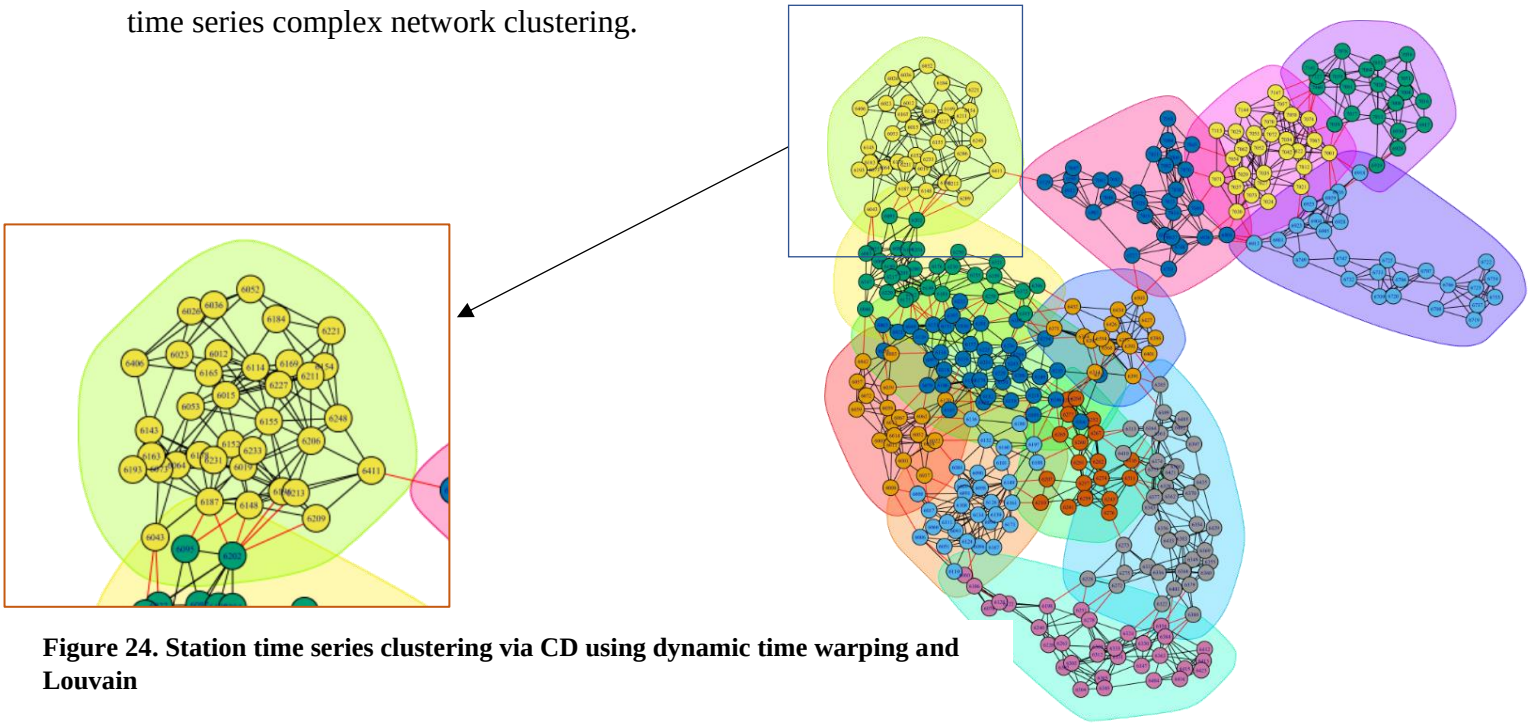


Figure 24. Station time series clustering via CD using dynamic time warping and Louvain

5. Conclusion

Using BIXI data has permitted us to explore and become acquainted with a variety of algorithms for community detection, including the novel approach of time series clustering for studying spatiotemporal data.

Our results indicated that some community detection algorithms were more appropriated for our bicycle-sharing data than others. To be more precise, evaluating communities for such densely connected network with temporal significance was a challenge. We were able to overcome this by considering time acknowledging methods for complex network creation, like visibility graph and time series clustering. We still retain the limitations of working with data that does not contain full journey information, which would make for an interesting future study.

Another interesting aspect for further study would be to separate the network into workday usage and weekend usage. Previous studies have shown distinct community structures for usage related to commuting vs for leisure. Zaltz et al. (2013) found that during the week, the network would exhibit communities analogous to industry clusters, quite different to the clusters identified for weekends. This aspect could be studied in a more focused project.

References

Chen, M., Kuzmin, K., & Szymanski, B. K. (2014). Community Detection via Maximization of Modularity and Its Variants. *IEEE Transactions on Computational Social Systems*, *1*(1), 46–65.

Ferreira, L. N., & Zhao, L. (2016). Time series clustering via community detection in networks. *Information Sciences*, *326*, 227–242.

Froehlich, J., Neumann, J., & Oliver, N. (2009). Sensing and predicting the pulse of the city through shared bicycling. *IJCAI 2009*.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, *486*(3), 75–174.

Lacasa, L., Luque, B., Ballesteros, F., Luque, J., & Nuno, J. C. (2008). From time series to complex networks: The visibility graph. *Proceedings of the National Academy of Sciences*, *105*(13), 4972-4975.

Lathia, N., Ahmed, S., & Capra, L. (2012). Measuring the impact of opening the London shared bicycle scheme to casual users. *Transportation Research Part C-emerging Technologies*, *22*, 88-102.

Li, W., Kang, Q., Kong, H., Liu, C., & Kang, Y. (2020). A novel iterated greedy algorithm for detecting communities in complex network. *Social Network Analysis and Mining*, *10*(1).

Newman, M. E. J. (2004). Detecting community structure in networks. *The European Physical Journal. B*, *38*(2), 321.

Parés, Ferran & Garcia-Gasulla, Dario & Vilalta, Armand & Moreno, Jonatan & Ayguadé, Eduard & Labarta, Jesús & Cortés, Ulises & Suzumura, Toyotaro. (2018). Fluid Communities: A Competitive, Scalable and Diverse Community Detection Algorithm.

Traag, V. A., Waltman, L., & Van Eck, N. J. (2019). From Louvain to Leiden: Guaranteeing well-connected communities. *Scientific Reports*, *9*(1).

Zaltz, M., O'Brien, O., Strano, E., & Viana, M. (2013). The structure of spatial networks and communities in bicycle sharing systems. *PloS one*, *8*(9), e74685.