Automating Consistent Mapping of Temporal Regions

Geospatial Programming-Spring 2024

Maryam Torkashvand

Introduction and Goal

Most recently, studies have utilized social network analysis to examine migration, where places are seen as nodes and flows as directed edges. This network representation captures the relational linkages and internal dynamics of migration systems. By modeling the migration communities (clusters) in these migration networks across different time intervals, we can study the structural evolution of interaction between places over time. However, automotive mapping these communities in a comparable way is a challenge. Tracking the changes in communities and network structure based on the overlap between communities in different time steps can be used as a criterion to define the symbology for the communities over time.

The aim of this project is to visualize migration regions using migration networks from a large multi-generational Family tree dataset in a comparable way. The main problem I try to address is a mapping and cartographic challenge when geocoded nodes move between communities across different time intervals. When the community code changes, the symbology cannot be perceived in all maps, and this makes the visual comparison between regions problematic. In this project, I suggested and applied two solutions. The first solution is based on the geographical location of nodes, and the second solution is based on tracking overlapped communities over time. Both solutions have their advantages and limitations. Defining community IDs based on the geographical coordinates of nodes is simple and effectively works for especially small networks. However, it has limitations when the network is large with many small communities, especially when the communities are geographically sparse and change drastically between networks. On the other hand, tracking similar communities based on their overlapped percentage is more mathematically robust and applicable to a broader range of networks.

I implement these methodologies to capture the evolution of the migration structure of European settlers within the US between 1872 and 1924. In terms of modeling internal migration, geographers typically focus on geographic areas and examine net or gross migration by aggregating both incoming and outgoing flows. Even though these aggregated flows oversimplify the migration system and represent the collective migration decisions of numerous individuals and households, they can illustrate patterns in migration and population redistribution (White & Lindstrom, 2005). Additionally, migration is a temporal phenomenon, and its structures and trends need to be studied over time. In other words, migration flows between areal units form structures over space and time. To study the migration structures over time, studies have utilized the concept of "*migration regions*" by identifying those areal units with the most similarities in flow characteristics and volumes. Migration regions are the groups of spatial units that have a high degree of internal connection in terms of migration flows but low interaction with other units in other groups (Pandit, 1994).

In recent years, more studies have developed migration networks with origins and destinations as nodes, connected by flows, and employed network analysis methods to identify migration regions. The Louvain (Blondel et al., 2008) and Leiden (Traag et al., 2019) algorithms are two well-known methods for network community detection. In this project, I used the Leiden

algorithm to extract the migration regions (communities) over three temporal steps (1872_87, 1887_1901, 1901_24) and applied two suggested workflows to map these regions over time.

Specific Objectives

In this project, my specific objectives are as follows:

- Making an aggregated network of nodes and links from flow data.
- Using *igraph* library in Python to extract communities from the migration data.
- Using *geopandas* and *matplotlib* libraries in Python to map the regions over time in different partitions.
- Mapping the communities in a visually comparable way using two solutions:
    - 1. make color hues consistent across maps for each region based on the geographic locations (coordinates) of nodes.
    - Make color hues consistent across maps for each region based on the percentage of overlap between communities over time.

The main challenge I am trying to address in this project is automating the process of mapping regions in a manner that is both comparable and consistent across various maps. Since network analysis and community detection algorithms do not consider the geographical locations of units when clustering them, achieving automated mapping of multiple maps with uniformity is complex.

Data

For mapping migration regions, I use extracted migration data from the family tree dataset, which has been developed by (Koylu et al., 2021). The family tree is a network of family members stretching over many generations, containing approximately 40 million individuals. Here, the data I am using is the tables that include lists of origin and destination states and the number of families who migrated between them in specific periods. The number of migrant families (flows) was derived from the family tree dataset by tracking changes in the birthplaces of two consecutive children within families. In other words, a change in the birthplace of successive children was considered as a migration event for that family.

According to Koylu et al. (2021), family tree dataset represents only white people. Therefore, black settlers are underrepresented in the migration network extracted from this dataset. Moreover, the migration network is obtained based on the child-ladder method, which looks at the birthplace of two consecutive children in a family, so the migration of loners or families with one child is not captured.

In this project, I use the migration flows between states in the United States in different time periods to construct migration regions and implement the Leiden community detection algorithm. The periods are obtained based on the maximization of the similarity of migration rates within the period. Therefore, in each period, we expect to have a relatively constant structure, while differences in the migration structure across periods. The three periods are 1872_87, 1887_1901, 1901_24. To map these migration regions, I used the shapefile for the United States at the state level in 1920 as the base map for these communities. The source for the shapefiles is the National Historical Geographic Information System (NHGIS). Table 1 is a few rows of flow data, including origin state, destination state, and flow; Figure 1 shows examples of U.S shapefiles.

Table 1. Migration flow data

orgState	destState	flows
117	151	5
121	124	26
121	123	1
121	122	9
124	152	97
117	148	15

Figure 1: Historical U.S. states boundaries.

Methodology and Workflow Model

Network analysis

To map migration regions, we need to first extract them from the migration flow matrix. I used the Leiden network community detection algorithm, which detects clusters based on the strength of connections between them. Leiden community detection is a method that uncovers community structure in networks. The algorithm works by optimizing a modularity function, which measures the density of links inside communities compared to links between communities. This method is particularly effective for large networks. It is widely used in fields such as social network analysis, biology, and computer science to identify natural divisions or groups within complex systems. Figure 1 shows the workflow for this step, which includes reading the flow data as inputs into the pandas DataFames, and next, building migration networks using the *graph* library. In the migration network, the nodes are the states of the US, and the edges are the volume of families who moved between states. Using migration networks, including origin, destination, and the volume of families who migrate between states in different intervals and applying the Leiden algorithm, the regions (communities) for each network are obtained. The result of this step is a set of csv files that contain the nodes and assigned partitions resulting from Leiden algorithm and text files that include the optimum modularity measurement for each network structure. The modularity measurement shows how strongly the states within these optimum communities are connected through migration flow.

The LeidenCommunityDetection.py code is the code for this step.

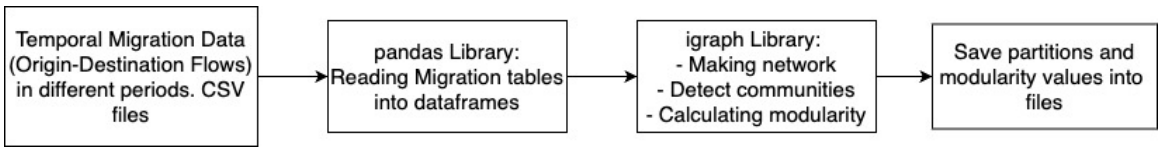


Figure 1. The workflow for community detection (step1)

Solution1

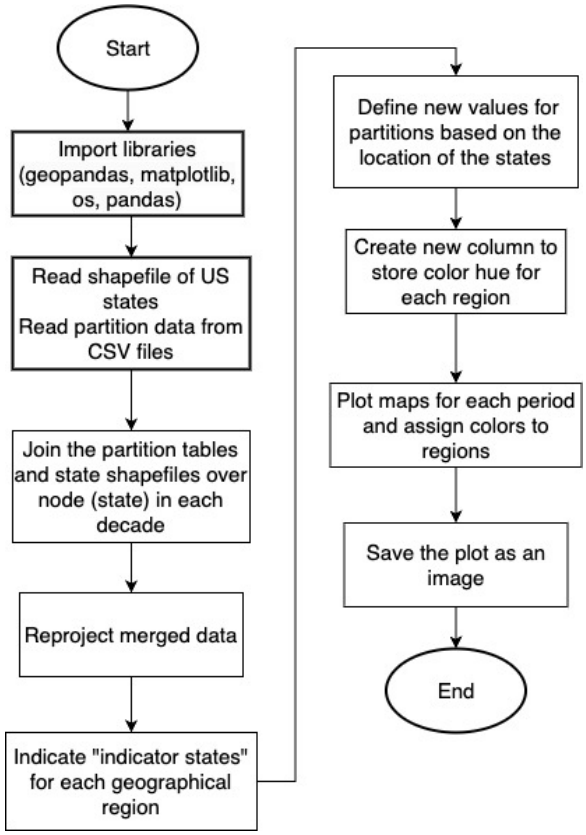
I designed and implemented two workflows to map the regions visually consistently. In the first workflow, I used the geographical location of nodes and defined a constant partition ID for those

that represent a specific geographical location. I defined "*indicator state*" in different geographical locations and assigned the color for regions based on the indicator states. For example, in our case study, I assume four main geographical directions, which states within them usually form a region, are Northeast, West, Midwest, Southeast, and South. I select a state that represents each of these locations in the United States and assign an immutable region ID across different networks over time. Then, change the initial community ID resulting from Leiden for all nodes with the same ID based on the ID of their communities' indicator state. In this way, the region colors are constant geographically across different periods.

The indicator states that I defined are:

Maine is for the Northeast, California is for the West, Georgia is for the Southeast, Texas is for the South, and the last indicator is the remaining community, which is not within the other regions.

Figure 2 shows the overall workflow of the first proposed method to handle inconsistency in the dynamic mapping of geographical regions results from the network community detection algorithm. First, we read the partitions constructed in the first step into a list of pandas DataFrames. Each partition DataFrame corresponds to a specific time interval. Also, we read the state shapefile into a GeoDataFrame using the geopands package. The code iterates through these DataFrames, joining partition data with the state shapefile and reprojecting the shapefile into Equal Areas projection. Then, we assign new values to partitions based on the states' locations. For example, Maine represents the Northeast (Region 1), Georgia the Southeast (Region 3), Texas the South (Region 4), and California the West (Region 5). Any non-string partition values are labeled as Region 2. Finally, we plot the maps for each time interval, using predefined colors for each region and generating a legend for clarity. The resulting visualization provides a comparative view of migration regions over time. The Solution1_RegionsByLocation.py code is the script for this part.



Solution2

Despite the simplicity and effectiveness of defining indicator states and mapping based on location proposed in solution 1, this method cannot be implemented on all varied network structures or needs many adjustments based on the study area. Therefore, I tried to design and implement a more global approach inspired by community evolution tracking and community similarity. I simplified the method …… used to track communities over time. ….. used more complex criteria and functions for tracking communities, such as the birth or death of a community and merging or splitting communities over time. For this project, I begin with the overlap as an indicator of similarity between two communities in two steps of time to use it for constantly mapping the communities over time. In the future, more work can be done to modify this method and make improvements to consider all different types of community evolution.

The process of tracking and analyzing community evolution over time is to map the communities that change, merge, split, or persist over time. A method for this mapping is based on calculating the similarity between communities at different periods using a metric such as the Jaccard coefficient. This coefficient is a common choice for comparing the similarity of sets and is calculated as the ratio of the intersection to the union of two sets. When this similarity exceeds a predefined threshold, the two communities are considered a match, thereby enabling the tracking of community evolution.

The Jaccard similarity index is calculated as follows and basically shows how much two communities are overlapped.

$$sim(C_{ta}, F_i) = \frac{|C_{ta} \cap F_i|}{|C_{ta} \cup F_i|}$$

Where C_{ta} represents a step community identified at a specific time step *t,* F_i *is* a front from the set of dynamic communities. A front is essentially a snapshot of a dynamic community at a previous time step. $|C_{ta} \cap F_i|$ is the number of nodes that C_{ta} and F_i have in common. It quantifies the overlap between the two communities. $|C_{ta} \cup F_i|$ is the total number of unique elements contained in both C_{ta} and F_i and represents the combined scope of the two communities.

Essentially, this similarity calculates the ratio of overlap between two communities in two times and ranges between 0 and 1. The higher the ratio, the more similar the two communities are. Using this index, we can define a threshold. If the similarity exceeds a certain threshold, it indicates that C_{ta} can be considered as continuing or merging into the dynamic community represented by F_i so we can map both communities with the same symbology. In defining the threshold, I tried different values and got different results, which I discuss them in the results and discretion part.

Figure 3 shows the workflow I used to map the evolution of migration structures in the US from 1882 to 1930 in three periods. The workflow begins by importing necessary libraries such as geopandas and matplotlib. Similar to solution 1, we read the U.S. state boundaries shapefile and reprojected it. Then, the code iterates through a directory containing CSV files of partitions and builds a list of these community structures. For each partition, the script merges the community data with the geographic data of states, mapping nodes in the community to the corresponding states. Then, we define a function to calculate the Jaccard Index, which quantifies the overlap between the current and previous partitions as a measure of similarity. If the overlap exceeds a

defined threshold, the communities are considered similar, and the regions are mapped accordingly to maintain visual consistency across time steps. Once all partitions are processed, the script visualizes these mapped regions over the specified time intervals in a series of maps.

The Solution2_RegionsByOvelap.py code is the script for this part.

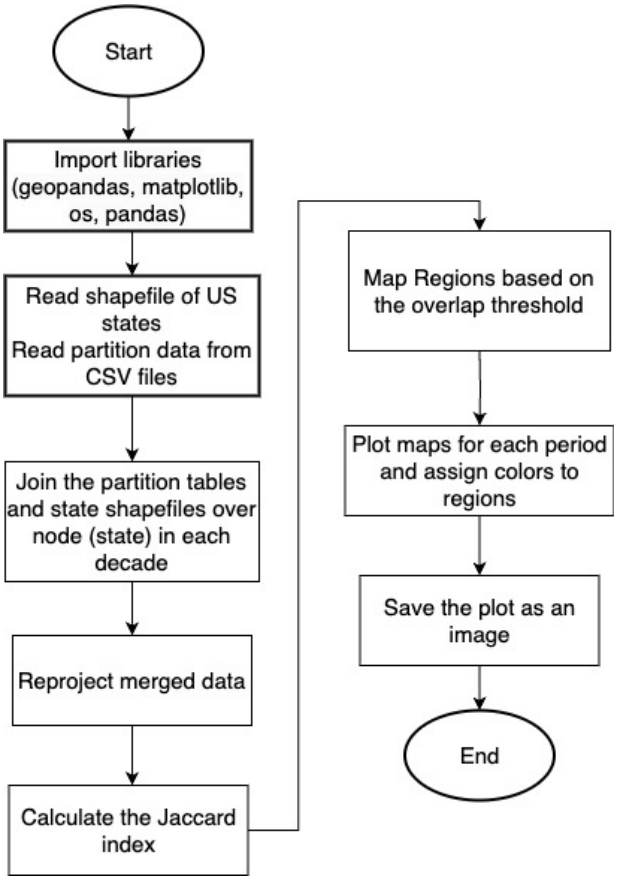


Figure 3. Workflow for mapping communities over time based on their similarity.

Results and Discussion

I implemented the Leiden community detection algorithm on the internal migration network in the United States in three periods, including 1872_87, 1887_1901, and 1901_24, and mapped the community structure evolution using two strategies. Figure 4 shows the results from the first strategy of mapping regions over time with a comparable symbology. It shows that defining indicators based on geographical location effectively maps the communities in similar geographical locations with the same symbology and we can track changes in communities over time using the maps.

However, it should be noted that this strategy has a major limitation. This code cannot be generalized and applied to different networks and different geographical regions. The logic may be capable of being implemented in other studies at the course level, such as the state in a country or counties within a continent, with some refinements and adjustments, but may not be useful in finer scale visualizations such as counties. In networks with a higher number of nodes and a higher

number of communities, finding and defining geographically indicator nodes is difficult, if it's not impossible. Therefore, although this solution is an effective, automated process of mapping regions for this specific case study and can save a large amount of time when we want to make more regional maps in different periods, it cannot be implemented in other networks without refinements and adjustments.

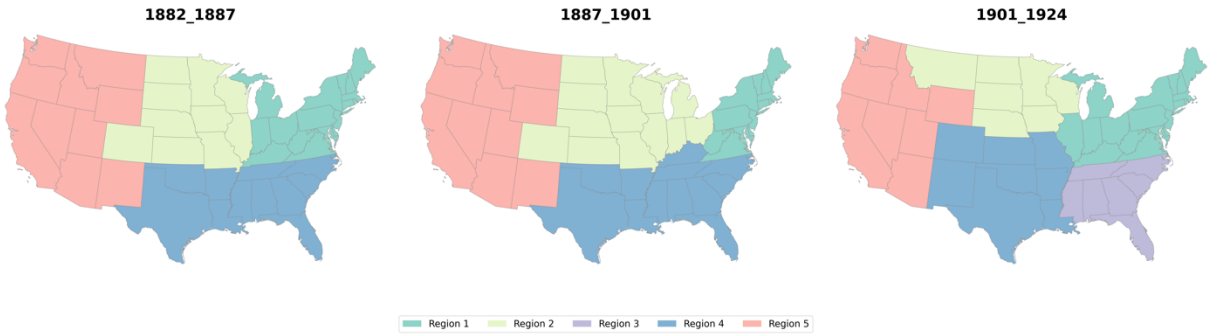


Figure 5. The US migration regions over time; mapped using solution1 workflow.

Considering the first solution's limitations, I designed and applied the mapping and tracking communities based on their similarity (overlap). I used two different thresholds for the similarity index, one 0.3 and the other 0.5. Figure 6 and Figure 7 demonstrate the different results for the third map using different thresholds. The smaller threshold cannot capture the new community in the third map and assign it as the same community as the fourth community. By increasing the threshold, we force more strict overlap criteria for tracking the community so it can capture the new partition in the third map. However, the symbology and colors are not consistent anymore, as the communities on the third map are not recognized as having evolved from the previous community. In conclusion, the overlap criteria can track the community evolution over time at some point, but for a better and more robust algorithm, we need to consider the birth, death, merge, split, and other functions of community evolution. This script is the first attempt at applying this logic to automate the task of mapping regions. In future works, I need to improve it and add more functions, and after that, it can be used in a variety of ranges of studying temporal network structures such as social networks and biology.

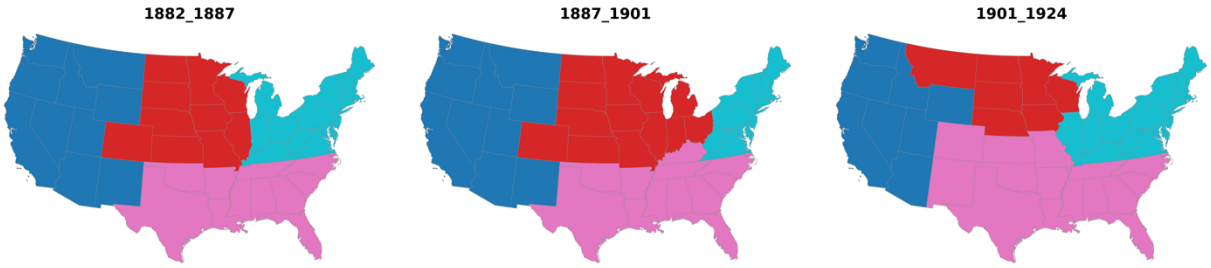


Figure 6. The US migration regions over time; mapped using solution2 workflow (threshold = 0.3)

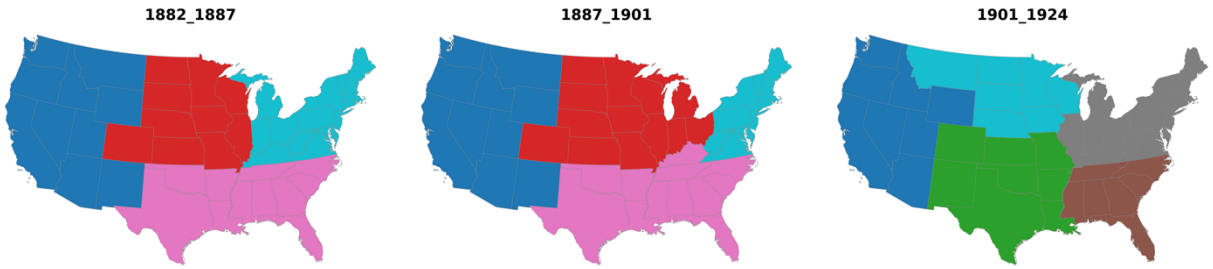


Figure 6. The US migration regions over time; mapped using solution2 workflow (threshold = 0.5)

Conclusion

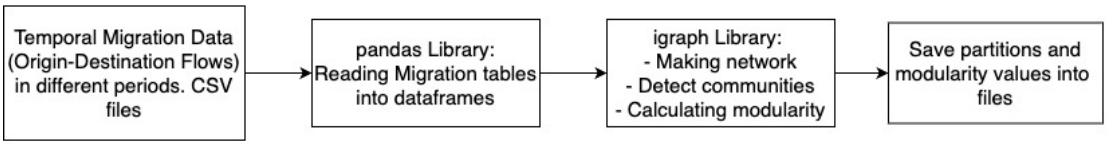
By adopting a network approach in migration studies, migration flows become the main units of analysis instead of the places themselves. Looking at migration through a network perspective can uncover the complexity of its components and highlight how migration structures and patterns evolve over time. However, studying and mapping the evolution of temporal migration structures resulting from network analysis in a geographical space is challenging as the partition (cluster) id is not necessarily geographically consistent. In this project, I addressed this mapping challenge by testing two solutions, one based on the geocoordinates of nodes and the other based on community similarity. I implemented these solutions on the US internal migration flows between states across three time periods. The results show that although the first solution is very local and the script is designed very specifically for my study, it effectively maps the regions based on their geographical directions. Using this code, I can save time in the future mappings of temporal changes in the different structures within the US and at the state level.

The second solution is more compatible with broader case studies as it is based on the similarity of the communities between periods. I used the overlap ratio as the similarity index. However, the overlap is not the only community evolution index. Communities may emerge, merge with each other, split into more communities, and experience other evolutions. In future works, I need to consider the other. Form of community evolutions to achieve a more robust framework for mapping temporal network structures in a geographical space.

Appendix

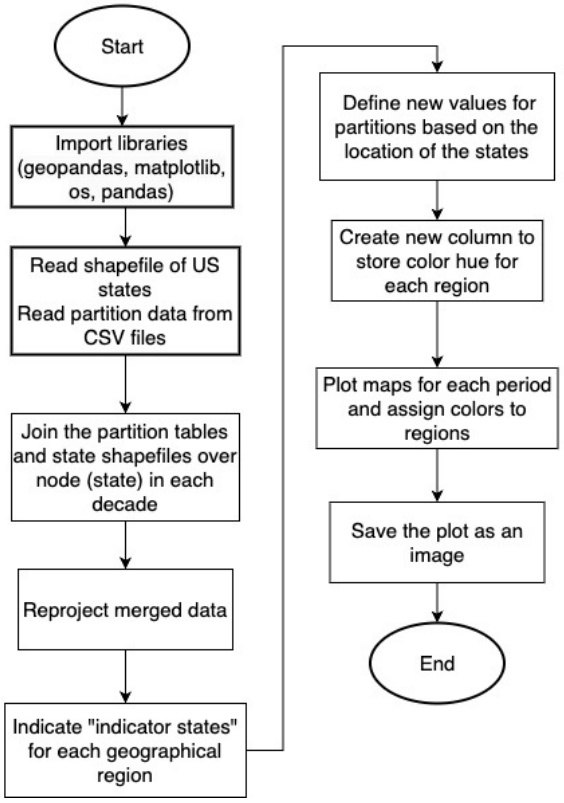
The overall workflow.

Step 1: Building the graph and extracting communities using the Leiden community detection algorithm. LeidenCommunityDetection.py

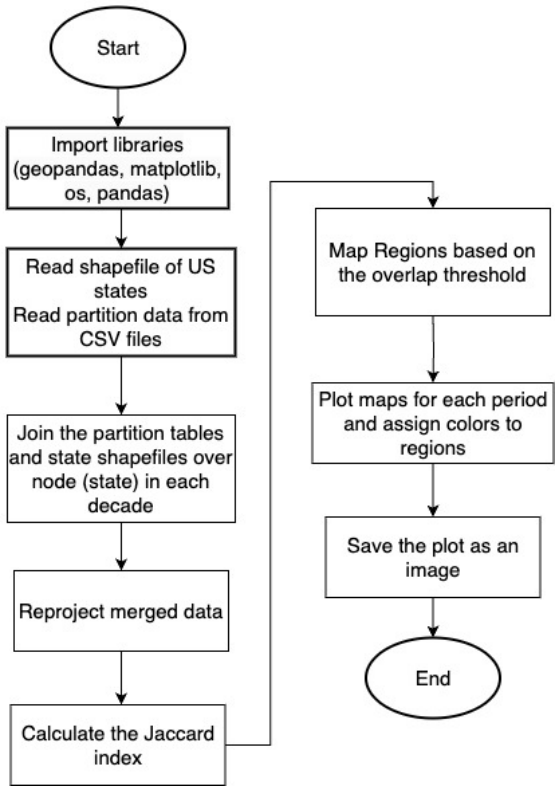


Step 2: Mapping temporal communities over time.

Solution1: Solution1_RegionsByLocation.py



Solution2: Solution2_RegionsByOverlap.py



References

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, *2008*(10), P10008.

De Haas, H. (2010). The internal dynamics of migration processes: A theoretical inquiry. *Journal of ethnic and migration studies*, *36*(10), 1587-1617.

Koylu, C., Guo, D., Huang, Y., Kasakoff, A., & Grieve, J. (2021). Connecting family trees to construct a population-scale and longitudinal geo-social network for the US. *International Journal of Geographical Information Science*, *35*(12), 2380-2423.

Pandit, K. (1994). Differentiating between subsystems and typologies in the analysis of migration regions: A US example. *The Professional Geographer*, *46*(3), 331-345.

Traag, V. A., Waltman, L., & Van Eck, N. J. (2019). From Louvain to Leiden: guaranteeing well-connected communities. *Scientific reports*, *9*(1), 5233.

White, M. J., & Lindstrom, D. P. (2005). Internal migration. In *Handbook of population* (pp. 311-346). Springer.