# Implementation & Application of Origin-Destination Flow Data Smoothing & Mapping

**BDSS IGERT Team, Cohort 2014-2016**
Pennsylvania State University
State College, PA 16801

## Abstract

An implementation of the visualization algorithm developed in [1] is presented. This procedure is then applied to an extended dataset of migration flows ranging from the years of 1978 to 2011. Preliminary results, primarily in the form of visualizations, are presented. We also discuss several next steps in improving or using this algorithm for visualization of flow maps.

## 1 Overview

Often, visualization of time-based network data is difficult, and it is only getting more difficult as data-sets grow larger in size given improvements in sensing technology. Data collected regarding locations across time can prove to be invaluable, allowing for studies of important phenomena that can be viewed as networks of various kinds, ranging from time-varying social networks [3] to temporal protein networks that govern organ development [2].

In particular, geographic mobility data is one domain where the particular challenge of visualization is clearly present, where massive bundles of visually over-lapping connections are present and map space with which to display network/connection information is limited. When data is plentiful, flow maps, which are useful for observing directed pathways between origin and destination locations, often suffer from the "curse of density" (as we coin the problem). In this case, massive intersections and overlaps, grounded by fixed-point locations (necessary for context) yield a cluttered and uninformative, if not utterly incomprehensible, map-image.

The problem of visualization (for flow maps), however, has seen significant previous study, and, although still a challenging issue to address in modern day analytic problems via computational methods, can be decomposed into three objectives: (1) the cluttering Problem, (2) the modifiable area unit problem, and (3) the normalization problem. The first problem entails the issue of information loss, where bundling or re-routing approaches often simplify connections that lead to a difficult-to-interpret map render or require heavy user interaction (i.e., "human-in-the-loop"). The second problem summarizes the challenge of aggregation–arbitrary aggregation (at a non-informed scale) can lead to a degradation of spatial resolution or generate incorrect or incomplete patterns. The third and final problem deals with fundamental units–selecting a reasonable atomic geographic unit, such as the "county", means selecting a unit that varies widely in either size (i.e., population) or area (i.e., geographic coverage) creates invalid comparisons of flows/connections. The goal of any method designed to generate an interpretable visualization of a flow-map must be able to solve the above three problems while still "faithfully" preserving the critical, important flow patterns and regularities within the complex data-driven network.

While the above problem above certainly requires further research efforts to develop efficient, scalable methods for flow-map visualization, fortunately, [1] recently proposed a novel approach that attempts to address all the three critical objectives of flow-map visualization. The approach essentially combines a kernel-density estimation method with generalization method for removing spurious variance in the data and smooth and normalize the flows to a controllable neighborhood

size. More importantly, the method is able to preserve and bring out the key "high-level" patterns in the data, useful for facilitating exploratory analysis.

We decided to implement the approach in [1] and test it on actual IGERT-held data (from previous hackathon events). In Section 3 we will present some basic, preliminary results depicting the capability of the model.

## 2 Implementation of the Flow-Map Visualization Algorithm

In this section, we describe the original algorithm we chose to implement (in the R statistical programming language).

### 2.1 Method Design

As mentioned earlier, we implemented the approach developed by [1] to extract (hard-to-detect) underlying patterns in large-scale geographic mobility data to generate informative visualizations of such data. These visualizations are ultimately useful in developing a meaningful understanding of complex systems and their essential space-time dynamics, such as complicated flow trends or migration patterns of freshly graduated college students to areas of work.

The approach in the paper works, at a high-level, in a multi-step fashion. First, flows (defined via their origin and destination points) are re-estimated via a controlled neighborhood-based smoothing approach (which can then be used to extract underlying regularities as well as be used in flow-map rendering). A kernel-based approach, where the choice of kernel varies perhaps depending on context (i.e., fixed bandwidth, or fixed geographic distance for all locations, and adaptive bandwidth, where bandwidth is defined via some attribute threshold such as number of units), is taken to conduct flow estimation and smoothing.

More specifically, with respect to the first step, given a neighborhood size threshold, $p$, and a kernel bandwidth[1], the neighborhood for a given location (such as an origin or destination location) can be found by first constructing a $p$-sized neighborhood, that contains the smallest $k$-nearest-neighbors (including the target location itself) that also satisfies the threshold constraint. A set of flows is then constructed such that their origin and destination points lie inside the neighborhoods previously constructed for the initial targeted flow. A kernel model is then defined to compute the weights for each flow in the constructed set of the previous step with the original, targeted flow. A Gaussian kernel model (as defined in more detail in the source paper) can be used to determine location weights for each point in a neighborhood which (applied to both origin-destination point pairs) is then used to compute joint probabilities for all flows. These derived probabilities or weights can then be used to smooth or re-estimate flows of the original data-set (optimized by ignoring flows shorter than a cut-off threshold *minDist*, a minimum distance meant to prune out local flows).

Second, after smoothing, to solve the cluttering problem, a flow selection and generalization method is applied to select representative flows from the results of the first smoothing step. The key is to find a representative subset of flows that capture the essence of major flow patterns and yet contain no duplicate information (which is generated by the previous step of the approach). After sorting flows by their smoothed flow values from largest to least, flows are selected (for final usage) if they do not share flow neighbors with each other and if flows are not too close to each other (ultimately yielding a "sparser" visual map). Stopping criterion for this routine include a maximum number of flows to select $l$ and the critical stopping point when there are no more flows to select.

It is important to note that the overall time complexity of the entire approach is $O(nk^2)$, and thus scales nicely to large-scale data-sets when the number of nearest neighbors $k$ is reasonably small (note that $n$ is the total number of flows in the original data-set). For an in-depth treatment of the actual multi-step approach, we recommend consulting the original source [1] and the accompanying code found at the GitHub link provided in the next section.

[1]Note that the bandwidth of a kernel, which was set to Gaussian as in the study, for a given location in a set of points, is chosen to be equal to the radius of the smallest circle that covers all points.

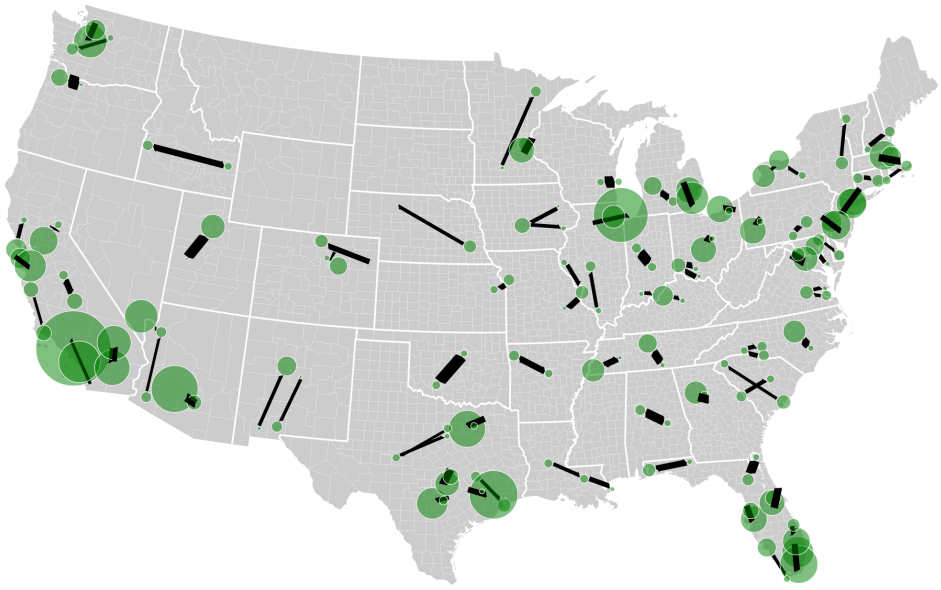


Figure 1: A flow-map generated from our implementation of the visualization algorithm.

## 3 Experimental Results

We used our implementation of the approach in [1] to generate a visualization of the large-scale County Migration data-set, which contains 1702436 samples, each with dimensionality equal to 18 (i.e., 18 covariates or features). This data is composed of samples collected across the time-range of 1978 to 2011 and contains features including ID's, string names, and geocodes for both origin and destination points of each flow (among other useful statistics that the visualization algorithm would not make use of, such as number of tax filings).

The algorithm was run on a 40-core machine, using Parallel R, resulting in an approximate total run-time of 2 hours. Scripts (using scripting languages such as JavaScript) were written to generate the actual visualizations that appear in this paper from the flow algorithm's output. The code, scripts, and preliminary program output (which serves a sort of sample of the program's capability) are available on GitHub [2].

Figure 1 depicts a holistic view of the flow-map generated by our implementation while Figure 2 shows zoomed insets that displays the finer details of areas of interest in the map. We see that, acknowledging that these are but preliminary results, we have successfully applied the algorithm of [1] to the target county migration data-set to generate a sparse but visually useful flow map of the originally dense data.

Furthermore, we see in Figure 3 four other visualizations. The first two represent all top 50 flows and the second two represent just the top 100. For each set, one map contains population centroid scaling while the other does not. We note that it is a bit difficult to identify which counties link to which when the circles are scaled based on population. The flows still seem to be geographically closer than one would expect. Interestingly enough, perhaps unfortunately, most links are within state or between adjacent states. However, upon some rudimentary comparison with results obtained in the source paper, we observe that the vast majority of flows are also within state or one state over (even a handful of roughly the same flows in the source paper's images also appear in our results). We speculate that tweaking the original visualization algorithm might ultimately be the best route to perhaps alter results in a meaningful fashion (as well as adjust algorithmic hyper-parameters, such as the $p$ threshold, perhaps via searching a discretized grid).

[2] https://github.com/flinder/mig_flows/

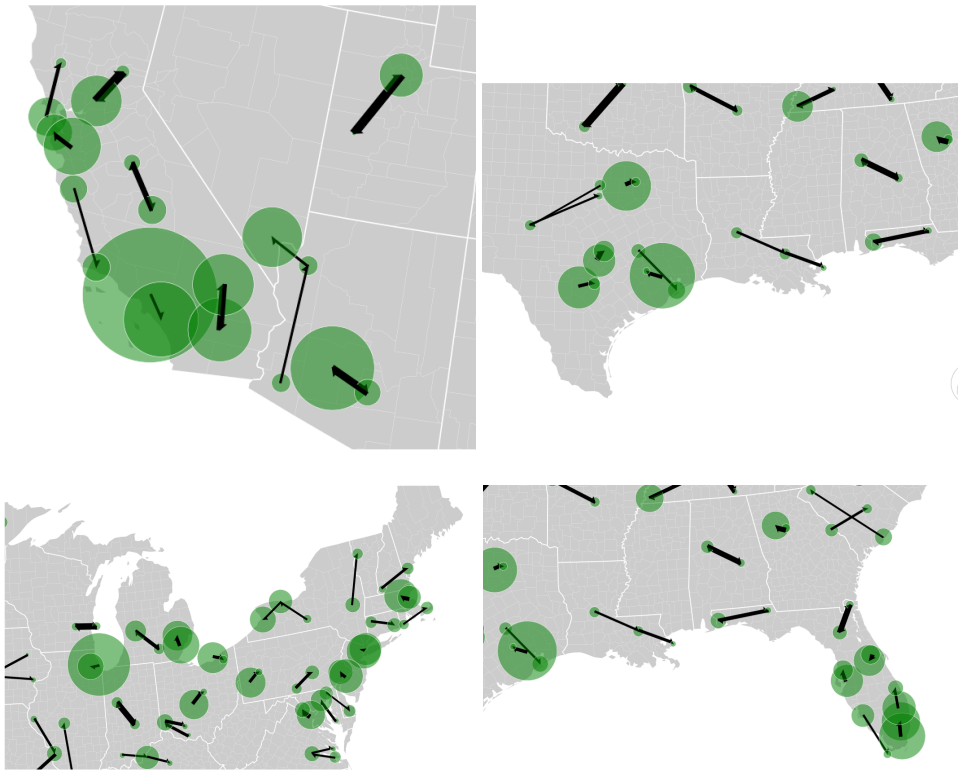
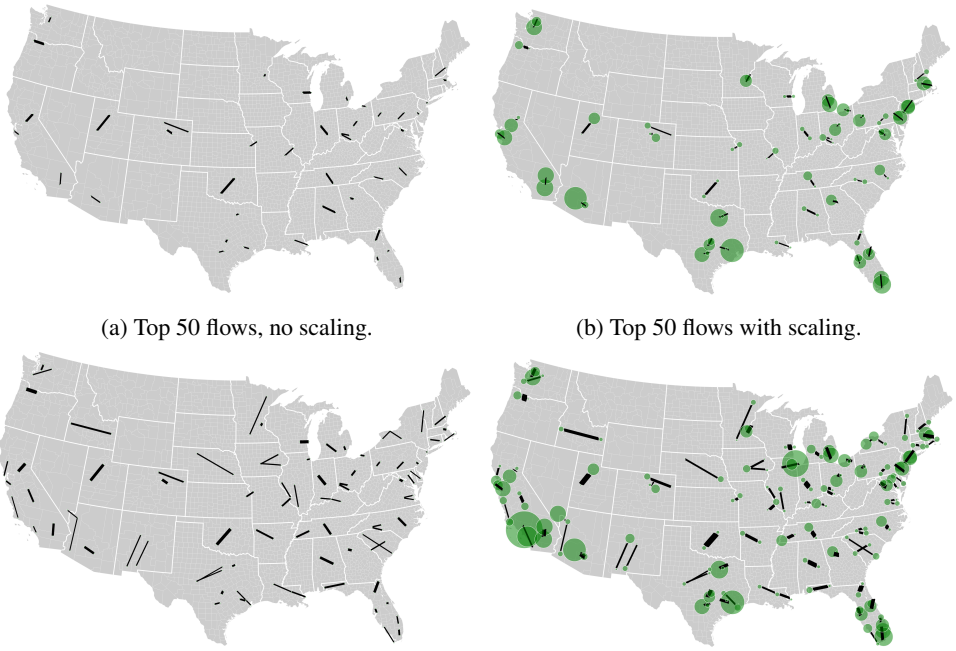


Figure 2: Several zoomed-in shots of several sub-sections of our generated map. These highlight interesting regions that are effectively pulled out by the algorithm, ideally zones that contain interesting regularities within the data-set. Also note the directional nature of the flow arrows, which indicate direction of flow or migration movement.



(a) Top 50 flows, no scaling.

(b) Top 50 flows with scaling.

(c) Top 100 flows, no scaling.

(d) Top 100 flows with scaling.

Figure 3: Several zoomed-in shots of several sub-sections of our generated map. These highlight interesting regions that are effectively pulled out by the algorithm, ideally zones that contain interesting regularities within the data-set. Also note the directional nature of the flow arrows, which indicate direction of flow or migration movement.

## 4 Conclusions

We successfully implemented the flow-map visualization algorithm described in [1] and applied to the the desired target County Migration data-set. The generated visualizations appear to follow the lay-out intended by the original source paper, yielding sparser yet potentially meaningful representations of the original data to facilitate improved exploratory analysis and initiate the process of uncovering interesting general patterns within the data (i.e., high-level, critical migration patterns). Future work includes improving internal algorithmic efficiency (such as modifying the code to operate on a multi-node, multi-threaded super-computing system, efficiently exploiting hardware resources to minimize computational time) as well altering key elements within the visualization approach (such as changing the kernel types for smoothing, which were set to mimic the choices made in the original paper). All in all, we have implemented a useful tool that may play an important role in future IGERT applications and competitions, including augmenting research targeted for answering questions in the most challenging of problem domains, for example , *The Shrinking Cities Problem*!.

## References


[1] GUO, D., AND ZHU, X. Origin-destination flow data smoothing and mapping. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2043–2052.

[2] LAGE, K., MLLGRD, K., GREENWAY, S., WAKIMOTO, H., GORHAM, J. M., WORKMAN, C. T., BENDSEN, E., HANSEN, N. T., RIGINA, O., ROQUE, F. S., WIESE, C., CHRISTOFFELS, V. M., ROBERTS, A. E., SMOOT, L. B., PU, W. T., DONAHOE, P. K., TOMMERUP, N., BRUNAK, S., SEIDMAN, C. E., SEIDMAN, J. G., AND LARSEN, L. A. Dissecting spatio-temporal protein networks driving human heart development and related disorders. *Molecular Systems Biology 6*, 1 (2010).

[3] SANTORO, N., QUATTROCIOCCHI, W., FLOCCHINI, P., CASTEIGTS, A., AND AMBLARD, F. Time-varying graphs and social network analysis: Temporal indicators and metrics. *arXiv:1102.0629 [physics]* (2011).