Higher-order networks: Exploring air traffic dynamics

Bradley Dice
bdice@bradleydice.com
University of Michigan
Department of Physics
Ann Arbor, Michigan

Daniel McCusker
dmccuske@umich.edu
University of Michigan
Applied Physics
Ann Arbor, Michigan

Shannon Moran
moranse@umich.edu
University of Michigan
Department of Chemical Engineering
Ann Arbor, Michigan

1 MIDTERM SUMMARY

In this project, we proposed studying trajectory data to explore higher-order network representations. We considered available data sets and settled on an air traffic data set, which has previously been used as a model system in the literature [4]. As of this midterm report, we have developed a pipeline for data extraction and transformation into parsable trajectories, and have built and analyzed both first order and higher-order networks (HONs) for a subset of the data (Q1 2011). We detail this progress in the Experimental section. We will next extend this work to the full data set from 1993-2018. The core piece of our proposal was implementing a community detection algorithm on the HONs. In a slight expansion on the original proposal scope in response to feedback, we plan to implement multiple community detection algorithms to compare the resulting communities identified in HONs with each method. Overall, we are on track to deliver the project scoped in our proposal.

2 DATA: AIRLINE TRAFFIC DATA

2.1 Data choice and collection

We chose data from the Airline Origin and Destination Survey (DB1B), collected by the Office of Airline Information of the Bureau of Transportation Statistics. We made this choice due to the quality of the data, consistency of collection (1993–2018), and its previous use as a model data set by other papers in the field [2]. This database contains three primary data views: *Coupon*, *Market*, and *Ticket*, aggregated on a quarterly basis from 1993 Q1 to 2018 Q3. This includes 837 million observations of itinerary coupons, 507 million observations of origin/destination markets, and 286 million observations of airline tickets.

We successfully downloaded and imported this data into Python `pandas` dataframes (only one quarter of data per dataframe for visualization). The data appear to be clean (no `NaN` values) with few outliers (e.g. one itinerary that had over 900 passengers on it - sounds like a party!).

We determined the following, after cross-referencing with a number of airline industry sources:[1]

- **Coupons** represent individual boarding passes, e.g. one leg from LGA-DTW. This is the data set we ultimately use, as described below.
- **Tickets** represent collections of Coupons. A round trip ticket from DTW-LHR via JFK would be a ticket composed of 4 Coupons: DTW-JFK, JFK-LHR, LHR-JFK, and JFK-DTW. To get this trajectory, we need to join the Coupons data based on a ticket ID. As a result, we did not need to use the Tickets database except to spot-validate that our data interpretation was correct.
- **Markets** represent journey segments that are separated by a break in the travel. For example, a round-trip ticket to a conference and back would be composed of two markets. The first market is the series of flights (including layovers) on the way to the conference, followed by a break in travel (for the conference). The second market is the return portion of the trip, ending in the traveler coming home. One-way tickets are composed of only one market, whereas multi-city flights purchased on the same ticket could have three or more markets. We are interested in including travel irrespective of markets, so this data set is not useful to us.

The raw DB1B Coupon data gathered from the Bureau of Transportation Statistics contains 35 columns with a primary key for the coupon, foreign keys for linking coupon data to data in the Ticket and Market database, and a number of fields describing that coupon (e.g. origin and destination airports, various metadata).

2.2 Data management with `signac`

Because there are over 100 quarters of data, we needed an organizational scheme that would offer the flexibility to rapidly experiment on one quarter and automate our operations as we scale to the full data set. To do this, we use the open-source `signac` framework (https://signac.io). The project workflow script performs operations on each quarter of data using a directed graph of tasks with pre- and post-conditions. In this case, the operations graph (Figure 1) runs the entire workflow, from downloading raw data through processing with PySpark. The entire operations graph is run for each quarter of data, and each operation can be run in serial or parallel depending on memory requirements.

While we were able to load one quarter of one year of data into a `pandas` dataframe, analyzing multiple quarters (as we intend!) required expanding to use PySpark[2], as we planned in our proposal. PySpark is compatible with Hadoop (and all analysis has been run on `flux-hadoop` to date) but is easier to program than Hadoop because it has the concept of `DataFrames`. Staying in Python also means that it is easier to rapidly prototype algorithms and utilize Jupyter notebooks for processing data. Operations were prototyped interactively in Jupyter notebooks, configured to interface with the Spark cluster in a real-time session, and then ported into the `signac-flow` project workflow script. Smaller pre- and post-processing tasks utilized `pandas`. All jobs (`bash`, Python, `pandas`, PySpark) were run using `signac-flow`. The entire workflow (downloading input data, parsing and pre-processing, Spark submission,

[1] https://www.iata.org/policy/Documents/coupon-use-paper.pdf

[2] https://spark.apache.org/docs/latest/api/python/index.html

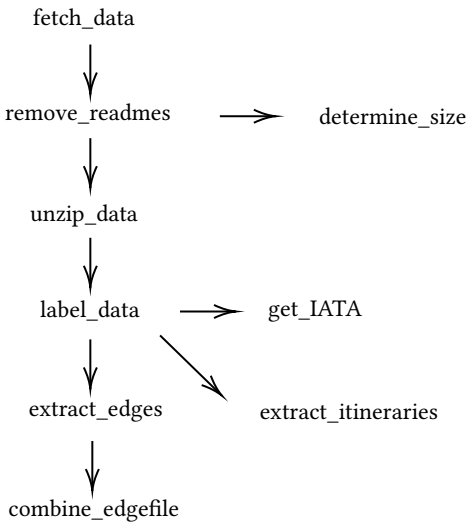

Figure 1: Operations graph implemented in `signac-flow`.

and analysis) can be reproduced or re-run by calling one terminal command.

2.3 Data transformation into networks

Transforming this data from massive CSV files into first-order and HON networks required two different operations. For the first-order network, we represented the data as a weighted directed graph. The coupons were parsed to extract origin and destination airports, and the total number of coupons grouped by each origin/destination pair was summed to give a weight for the (origin, destination) directed edge in the graph. For the higher-order network, coupons were grouped by their ticket's unique identifier so that a series of airports could be constructed for each ticket. This led to two PySpark operations with different output formats.

3 PROPOSED METHODS

Our project relies upon the following methods, which increase in order of complexity.

(1) Build first order network from coupon data
(2) Build a higher-order network (HON) from compiled trajectory data
(3) Implement community detection on HONs: InfoMap versus motif-based community detection

3.1 Build first order network from coupon data

We build weighted directed first order networks from the coupon trajectories compiled from the raw data as outlined in the "Data" section. As a note, we could also weight by the total number of *passengers* on each edge, but since most (>75%) of coupons are only one passenger, we decided this was an incremental update we did not need.

3.2 Build a higher-order network (HON) from compiled trajectory data

First, coupons were merged by their itinerary's unique identifier to form trajectories. These trajectories are the input for the `BuildHON+` algorithm. Previous work on DB1B data by Rosvall et al. made some assumptions about paths' endpoints (e.g. choice of memory nodes) that make the problem simpler and capture helpful dynamics [4]. Specifically, the first and last airports on a ticket are included twice to enforce the inclusion of self-memory nodes that reinforce the importance of a home city over transfer traffic. This will also make our analysis more comparable to the literature result.

We use the improved `BuildHON+` algorithm of Xu et al. [5] to build a higher-order network of this data. There is an open-source Python implementation[3] that we used for analysis[5, 6]. For now, we have allowed the algorithm to dynamically choose its own cutoff order k for each trajectory history by setting a large maximum value ($k = 99$). The algorithm's time complexity is not strongly dependent on k after a certain point, as seen in Table 1 of the HON paper where the run time of $k = 3$ and $k = 5$ differ by only 8% [6]. This is because there are typically far fewer correlations of high order than low order, so testing correlations of order k for extension to order $k + 1$ gets cheaper as the number of correlations decreases.

3.3 Implement community detection on HONs: InfoMap versus motif-based detection

We'd like to follow up on the proposal feedback, which asked how we were going to evaluate the quality of the communities we find. Our first metric would be "accuracy" relative to the communities identified by Rosvall et al. [4]. However, we do not have explicit information about the airport-to-community mappings identified in that work; we have only a few representative samples.

Instead, we've decided to compare the relative "performance" of different community detection methods. This exercise will thus give us an opportunity to explore the relative strengths and weaknesses of these methods.

First we plan to implement InfoMap, a random walker-based (MapEquation) community detection algorithm, to detect overlapping communities in the airport nodes [3]. This will allow us to directly compare our results to the fixed second-order airline representation and expand it to k-order representations [4]. An additional advantage of this approach is that there is already an available implementation in `C++` and Python, which we expect to be able to leverage in part.[4]

A method that has been used to identify *clusters* in higher-order networks is based on motif detection [1]. In this algorithm, clusters are built to minimize the number of a given motif that are cut when an edge is removed. Code is available in MATLAB and will need to be translated to Python for our project[5]. While this method has been applied to higher-order networks, it has never been applied to data of this scale and to transportation data specifically. There is a simple example of traffic data that was analyzed, though is not directly analogous. In addition, the authors assert that this can also

[3] https://github.com/xyjprc/hon
[4] https://github.com/mapequation/infomap
[5] https://github.com/arbenson/higher-order-organization-matlab

Figure 2: Shown is a best-possible visualization for the first-order network built using the trajectories compiled from the available coupon data for Q1 2011. This is a low-diameter network (average shortest path length of 2.36), and it is apparent from this representation that an alternative (in this case, higher-order) approach is needed to glean meaningful insights from the data.

be used to identify communities – but we need to explore further whether this claim is justified.

There is a **medium-risk** that this approach simply will not work for our data, but we are excited to explore it as this work has come up repeatedly in our review of the literature. We want to be very clear that this is an incremental "bonus" on top of our already-proposed work, and is a risky endeavor. We are going to make a good faith effort to complete it in four weeks but are not confident that it will be a clear success.

One might ask why we are not using another community detection algorithm discussed in class. In part, this is because performance for InfoMap is already benchmarked against the Louvain method and we were looking to do something more "novel" than replicating this particular work on the airport traffic network [4]. Additionally, the motif-based method had been discussed at several points in our literature review (and Benson, the lead author, has a PhD thesis which was very helpful introductory material for us), and we are excited to try implementing it in Python and applying it to our networks.

4 EXPERIMENTS

Here, we have framed each stage of our project as an "experiment." Our implicit hypothesis, unless otherwise stated, is that we will be able to generate findings that are reasonable – e.g. found in the literature on comparable data sets [4], or explainable by cross-referencing with common-sense analyses of airline data.

4.1 Experiment 1: Creating networks from flight trajectories

Hypothesis: We can represent airline flights as a first-order network using the coupon data. These networks will demonstrate a hub-and-spoke model, but will not preserve trajectory information.

First-order		Higher-order	
Airport Code	PageRank Score	Airport Code	PageRank Score
ATL	0.088	ATL	0.107
CLT	0.051	ORD	0.060
DFW	0.047	CLT	0.059
DEN	0.047	DFW	0.059
ORD	0.046	DEN	0.054

Table 1: Top five PageRanked airports for first-order and variable (k) higher-order networks.

Contrasting our initial hypothesis, we do not actually see a hub-and-spoke model. Instead, the system is highly connected and shows an average shortest path length of 2.361, as illustrated in Figure 2. We hypothesize that individual airlines may look more like a hub-and-spoke but because this data reflects the combination of many airlines, we observe a dense core with very few nodes at the periphery.

4.2 Experiment 2: Comparing higher-order network representations with PageRank

Hypothesis: A higher-order network will change the PageRank scores of some airports relative to the first-order representation. This will indicate that there is additional information embedded in the higher-order representation.

We visualize the results of our PageRank analysis for a first-order network in Figure 3. The result verifies our intuitions about which airports are major hubs: Atlanta, Chicago, Denver, LA, Philadelphia, etc. PageRank values for the five airports with the largest PageRank values calculated for the first-order and k-order representations are shown in Table 1.

In Figure 4, we highlight the ten airports that see the largest increase in PageRank when comparing k-order networks to first-order networks, as well as the ten airports that see the largest decreases. Both of these populations are composed of "hubs", while smaller airports are largely unchanged after accounting for higher-order dependencies (not shown in this report, but analysis is available).

We can take away from these results that a k-order HON representation is adding some amount of value and information to these network analyses. However, we will need to do more work to understand and build our intuition for why hubs are the airports whose PageRanks are most impacted by this improved representation. Our current hypothesis is that these hubs have the largest absolute PageRanks, and therefore even a minor percentage change in the PageRank taking into account higher-order dependencies would be a larger absolute change for hubs than minor airports.

5 PROGRESS AND NEXT STEPS

Our major tasks are outlined in the Gantt chart in Figure 5. Over the next four weeks, we will primarily focus on analysis of the HONs we have now successfully built. We are currently planning on implementing two different types of community detection, as outlined in the Proposed Methods. We are confident that given our progress over the last 3 weeks, this is an achievable goal. There is some risk that the expanded scope of community detection into

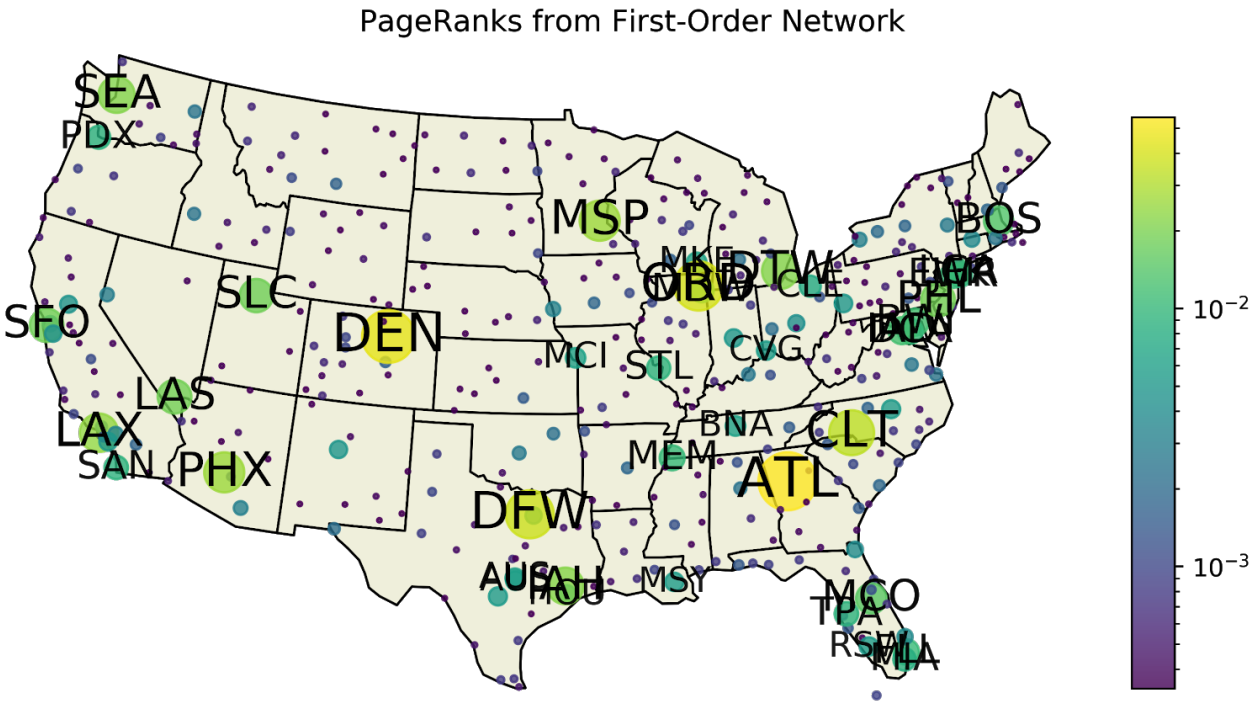

Figure 3: Nodes identified from the data set over period Q1 2011. Node size and color corresponds to the node's PageRank value based on a first order network, i.e. one that does not take higher-order dependencies into account.

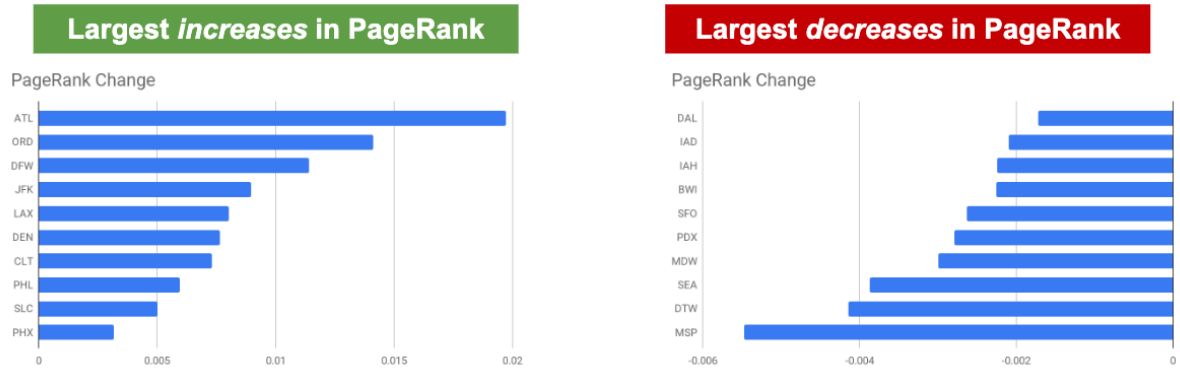

Figure 4: Airports showing the largest change in PageRank between first-order and k-order network representations built for Q1 2011 data. Discussion in the text.

motif-based approaches will be unsuccessful, but this was not part of the original proposal. We are on track to complete the work outlined in our proposal.

6 DISTRIBUTION OF WORK

Volume of the work was evenly distributed from the viewpoint of the team. For the midterm output shown here, responsibility biases were approximately as follows: Bradley focused on fetching and cleaning data, using PySpark for network extraction, and analysis of the first-order network; Dan focused on using pyHON with synthetic sample data, building the HONs from real data, and HON data analysis; Shannon investigated community detection algorithms and drove the writing of the report and presentation. All team members contributed to data visualization. This code vs.

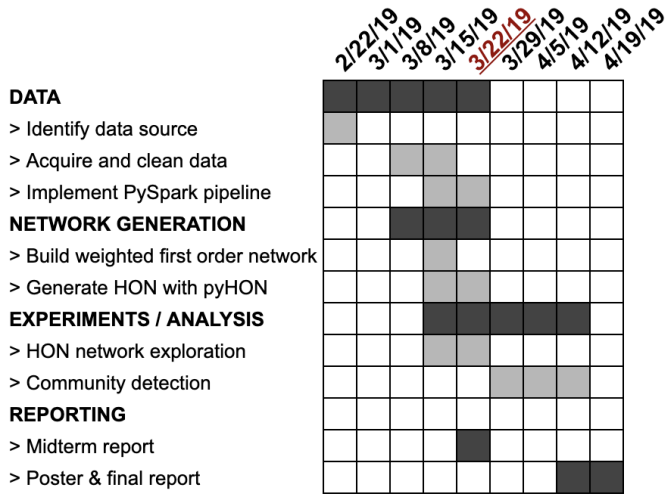

Figure 5: Gantt chart illustrating progress to date (underlined, in red) and remaining work. Our team remains on track to complete the work outlined in our proposal.

reporting load will shift over the next few weeks as we begin the community detection portion of the proposal.

REFERENCES

[1] A. R. Benson, D. F. Gleich, and J. Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (Jul 2016), 163âĂŞ166. https://doi.org/10.1126/science.aad9029

[2] Bureau of Transportation Statistics. [n. d.]. Airline Origin and Destination Survey (DB1B), 1993-2018. https://www.transtats.bts.gov/tables.asp?DB_ID=125

[3] M. Rosvall, D. Axelsson, and C. T. Bergstrom. 2009. The map equation. *The European Physical Journal Special Topics* 178, 1 (nov 2009), 13–23. https://doi.org/10.1140/epjst/e2010-01179-1

[4] Martin Rosvall, Alcides V. Esquivel, Andrea Lancichinetti, Jevin D. West, and Renaud Lambiotte. 2014. Memory in network flows and its effects on spreading dynamics and community detection. *Nature Communications* 5, 1 (aug 2014). https://doi.org/10.1038/ncomms5630

[5] Jian Xu, Mandana Saebi, Bruno Ribeiro, Lance M. Kaplan, and Nitesh V. Chawla. 2017. Detecting Anomalies in Sequential Data with Higher-order Networks. arXiv:arXiv:1712.09658

[6] Jian Xu, Thanuka L. Wickramarathne, and Nitesh V. Chawla. 2016. Representing higher-order dependencies in networks. *Science Advances* 2, 5 (2016). https://doi.org/10.1126/sciadv.1600028