Zooming in on NYC Taxi Data with Portal

Julia Stoyanovich*, Matthew Gilbride, and Vera Z. Moffitt

College of Computing & Informatics, Drexel University,
Philadelphia PA, USA
{stoyanovich,mtg5014,zaychik}@drexel.edu

Abstract. In this paper we develop a methodology for analyzing transportation data at different levels of temporal and spatial granularity, and apply our methodology to the TLC Trip Record Dataset, made publicly available by the NYC Taxi & Limousine Commission. This data is naturally represented by a set of trajectories, annotated with time and with additional information such as passenger count and cost. We analyze TLC data to identify hotspots, which point to lack of convenient public transportation options, and popular routes, which motivate ride-sharing solutions or addition of a bus route.
Our methodology is based on using an open-source system called Portal that supports an algebraic query language for analyzing evolving property graphs. Portal is implemented as an Apache Spark library and is inter-operable with other Spark libraries like SparkSQL, which we also use in our analysis.

1 Introduction

Many first-time visitors to New York City are surprised by the ubiquity of yellow cabs. Are NYC taxis a luxury and a menace to pedestrians, bicyclists and other drivers? Or are they a necessity — an efficient way to supplement public transportation options in the City that Never Sleeps, but where it can take you longer to go cross-town by subway or bus than if you were to walk, and where the only practical way to get to an airport is by taxi?

We set out to answer these questions by analyzing the TLC Trip Record Data, made publicly available by the NYC Taxi & Limousine Commission [7]. We downloaded 1 year worth of yellow cab data, spanning July 2015 through June 2016. Yellow cabs pick up passengers in Manhattan or at an airport like JFK and La Guardia, and drive them to destinations in any of NYC's five boroughs. In this case study, we answer the following specific questions.

Hotspots: What are the hotspots in NYC yellow cab utilization? Which origins and destinations are the most popular? Do popularity trends persist at different levels of geographic granularity? Presence of hotspots indicates that public transportation options are insufficient in these geographic locations.

* This work was supported in part by NSF Grant No. 1750179.

Popular routes: Do there exist sets of trips that (a) share an origin and a destination, and (b) originate at the same time, or within a few minutes of each other? If so, a point-to-point ride-sharing solution can be implemented to reduce cost to the passenger and to alleviate traffic congestion.

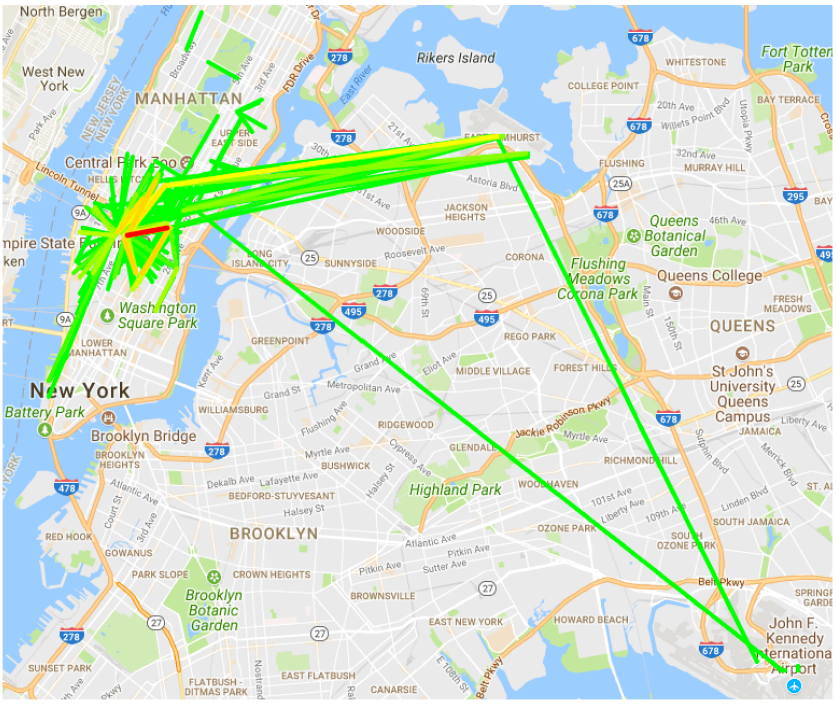

Fig. 1. Top-300 most frequent routes in March 2016, at 100-meter resolution.

Summary of results: We analyzed TLC data and found that there are indeed taxi utilization hotspots, and that although many popular origins and destinations persist at different levels of granularity, some do not. For example, Penn Station, JFK airport and La Guardia (LGA) airport are among the top-5 locations by both out-degree (origin) and in-degree (destination) at 10-meter and 100-meter location resolution, and are clearly hotspots at 100-meters, pointing to two unsurprising facts — that much of the taxi utilization in New York City is by tourists or by locals who travel in and out of the City, and that public transportation options serving these locations are insufficient. Perhaps more surprising is the proportion of the total number of cab rides in NYC with JFK, LGA or Penn Station as their origin, destination or both. Further, while Penn Station remains a hotspot at 1000-meter resolution, JFK and LGA airports are no longer among the top-5.

We found that there exist many popular routes, some with as many as 11 simultaneous trips during a particular month in 2016, and with only 1 or 2 passengers per car — a clear ride-sharing opportunity. Figure 1 gives an at-a-glance view of the results of our frequent route analysis. This figure shows the total number of trips between pairs of locations in March 2016, for 300 most frequent such pairs. One immediate insight is that LGA participates in many more taxi trips than JFK, despite being a much smaller airport.[1] Another insight is that the single most frequent route is between Penn Station and Grand Central — two train stations in Midtown Manhattan.

Both findings can be explained by the lack of convenient public transportation options that connect these out-of-town transportation hubs: Unlike JFK, La Guardia is not reachable by subway from Midtown Manhattan. It takes 2 trains (with a connection at the very busy Times Square station) and 20 minutes to travel between Grand Central and Penn Station by subway.

Context and contributions: While taxi data analysis is of independent interest, the main contribution of our work is a generalizable methodology for asking the kinds of questions we discuss in the remainder of this paper, and many others, over large graphs that evolve over time. Our methodology is based on using a system called Portal, which implements efficient representations and principled analysis methods for graphs whose topology (presence or absence of nodes and edges) and attributes of nodes and edges change over time.

Portal is available in the open source at `portaldb.github.io`. It is implemented on top of Apache Spark and is inter-operable with other Spark libraries like SparkSQL. Portal implements a set of algebraic operations over evolving property graphs [6], which allow to model and interrogate changes in network topology and in the values of vertex and edge attributes. In our experience, answers to the kinds of questions we ask in this case study can be computed in minutes for graphs with millions of edges on a cluster of modest size.

The data analysis methodology embodied by Portal and presented here is complementary to that of TaxiViz [4], a system for visual exploration of transportation data that was applied to (an earlier version of) the NYC TLC dataset. TaxiViz supports query types in Peuquet's spatio-temporal framework [8], interrogating data along three dimensions: space (where), time (when) and objects (what). In contrast, the representation and query mechanisms in Portal are based on evolving graph models, rather than on spatio-temporal models. In this paper, we make use of a particular aspect of this functionality — temporal and structural zoom.

2 Materials and Methods

2.1 Data

We analyzed 12 months worth of yellow cab data, spanning July 2015 through June 2016, that makes part of the TLC Trip Record dataset[7]. This dataset lists

[1] `https://en.wikipedia.org/wiki/LaGuardia_Airport`

pick-up and drop-off locations for each trip at 1-meter precision. In our analysis, we consider locations at different levels of geographic granularity, zooming out to 10 meters, 100 meters and 1 kilometer spatially. In addition to pick-up and drop-off locations, the dataset also lists the pick-up and drop-off times, the number of passengers, and the fare amount.

We represent this dataset of time-stamped trajectories with an evolving graph. In this graph, nodes correspond to locations, and directed edges correspond to trips. Latitude and longitude of the location is stored as an attribute of the node, while trip duration, fare and passenger count are stored as edge attributes. We record each trip from location a to location b as a single directed edge from a to b, and so our representation is technically a directed multi-graph, since multiple edges can connect a given pair of nodes.

The advantage of using an evolving graph representation for TLC data is that we can assign periods of validity to nodes and edges, and have the data manipulation operators supported by the Portal system handle these periods of validity implicitly. We will discuss this further when explaining our analysis methods in Section 2.3. We assign periods of validity to nodes and edges as follows. An edge is valid for the duration of the trip that it represents. A node becomes valid when the first trip originates from or arrives at that node, and persists until the last incoming or outgoing trip.

We cleaned TLC data, removing trip records that did not appear to be formatted properly, for which latitude or longitude was set to 0, or for which trip duration did not appear valid, such as when arrival time was later than departure time, or when the trip took longer than 2 hours. This resulted in removing less than 1% of all trips from our dataset.

2.2 Software and execution environment

To analyze TLC data and answer our research questions we used Portal, a system for analysis of evolving property graphs. Portal is implemented in Scala as a library of Apache Spark. We now give some technical background on Apache Spark and Portal.

Apache Spark[2] is a distributed open-source system similar to MapReduce [3], but based on an in-memory processing approach [9]. Data in Spark is represented by Resilient Distributed Datasets (RDDs), a distributed memory abstraction for fault-tolerant computing. All operations on RDDs are treated as a series of transformations on a collection of data partitions, such that any lost partition may be recalculated based on its lineage.

Apache Spark provides a higher-level abstraction over MapReduce, making it easier to use for data scientists and developers. It is open-source, and has attracted a vibrant developer community. Spark includes a variety of data processing libraries, including SparkSQL [1] — a distributed implementation of SQL, GraphX [5] — a library for analysis of static graphs, as well as streaming and machine learning modules, among others.

[2] `http://spark.apache.org/`

Portal[3] is an open source system built on Spark that implements operations of TGA, a temporal graph algebra for querying and analysis of evolving graphs [6]. The Portal API supports a variety of operations, including snapshot analytics like Page Rank, temporal and non-temporal variants of subgraph, binary operations that compute the intersection, union and difference of a pair of evolving graphs, and edge creation. All operations implicitly handle temporal information, and also allow access to time in predicates. Technically, TGA, the algebraic graph query language implemented by Portal, adheres to point-based temporal semantics [2]. The Portal API also exposes node and edge RDDs, and provides convenience methods to convert them to Spark Datasets, a relational abstraction on top of RDDs. This feature enables inter-operability between Portal and SparkSQL. In this paper we use several TGA operations and do further processing in SparkSQL.

To use Portal for our analysis, we exported TLC data into Apache Parquet format [4] for on-disk representation in the Hadoop Distributed File System (HDFS)[5], using separate files for nodes and edges. For this, we used converter and loader utilities provided by Portal. All data analysis was conducted on a 16-slave in-house Open Stack cloud, using Linux Ubuntu 14.04 and Spark v2.0. Each node has 4 cores and 16 GB of RAM. Spark Standalone cluster manager and Hadoop 2.6 were used.

2.3 Analysis methodology

In this section we give an overview of the TGA operators that are implemented in Portal and were used in our analysis. We refer the reader to our recent work [6] for a full technical description of TGA.

Structural zoom. We analyzed TLC data at three levels of geographic resolution (distance): 10 meters, 100 meters (roughly 1 NYC street block) and 1 kilometer, using the *node creation* operation of TGA. This operation can be viewed as a generalization of the SQL `group by`, applied to evolving graphs: it partitions graph nodes into non-overlapping groups based on the value of a node attribute, or on a value returned by a function that is invoked over the node. For example, we can partition nodes that correspond to employees based on the value of the attribute `department` — all employees who work in the same department are assigned to the same group. Or we can partition nodes that represent taxi pick-up and drop-off locations by rounding up their latitude-longitude coordinates to three digits after the decimal point — all locations that fall within the same 100m by 100m block will be assigned to the same group.

For each group of nodes in the input, a new node is created in the output; its validity period is computed by taking a temporal union of the periods of validity of the corresponding input nodes.

[3] `http://portaldb.github.io/`
[4] `https://parquet.apache.org/`
[5] `https://hadoop.apache.org/`

Edges that were present in the input persist and keep their attributes and their periods of validity as in the input, but are re-assigned to connect the newly created nodes. For example, consider two taxi trips e_1 and e_2, both originating in the vicinity of Penn Station at locations s_1 and s_2, and both terminating at the JFK airport at locations d_1 and d_2. If zooming out geographically places s_1 and s_2 into node group s, and d_1 and d_2 into node group d, then edges e_1 and e_2 will both point from s to d in the new graph.

Temporal zoom. We analyzed TLC data at different levels of temporal resolution: seconds (as in the raw data), 10 minutes, and 1 month. This was done using the *temporal zoom* operation of TGA, which maps validity periods of nodes and edges of the input graph to coarser validity periods in the output.

For example, consider two taxi trips e_1 and e_2, and suppose that they share the origin s and the destination d (either because s and d were literally the same in the input dataset, or because the original locations were mapped to a pair of common locations as a result of node creation). Suppose that e_1 leaves s at 11:01am, and e_2 leaves s at 11:03am. We may want to state that e_1 and e_2 both left s in the 11:01-11:10am time interval. To do so, we invoke temporal zoom over 10-minute windows. Under default conditions, all input nodes and edges persist in the result, but their periods of validity are adjusted (expanded) if necessary to cover the full window. In our example, e_2 will have its period of validity adjusted to start at 11:01am.

Additionally, we can specify node and edge quantifiers of the form `exists` (this is the default), `always` and `most`, which determine under what conditions a node or an edge should be included in the temporal window. With `exists`, a node or an edge is included into the window, and its validity period is adjusted, if it existed at some point during the window, even if for only one time instant. With `always`, we only include nodes and edges that existed throughout the entire window (and so an adjustment of their validity periods is unnecessary). Finally, with `most`, presence during more than half of the temporal window is required.

Different levels of temporal resolution were considered for different analysis tasks, as we will discuss in detail later in this section.

Aggregate messages. Another TGA operation that was used in our analysis is *aggregate messages*. This operation computes the value of an attribute at a node based on information that is available at its incoming or outgoing edges, and at its immediate neighbors (nodes reachable by one edge, in either direction). For example, we can use aggregate messages to compute the in-degree or the out-degree of a node, allowing us to quantify the number of incoming and outgoing trips for a particular location, the total cost of such trips, or their average duration. As with other operations, time is handled implicitly. That is, the number of incoming edges will likely differ for a given node depending on the time period (e.g., there were 35 trips out of JFK during 11:01-11:10am, and 27 trips during 11:11-11:20am), and change in this value over time is computed and handled implicitly by the system.

Putting it all together: implementation of the data analysis tasks. As part of our analysis, we used node creation, temporal zoom and aggregate messages to accomplish two tasks that correspond to our research questions. We summarize these tasks here, and present results in Section 3.

Hotspots. We analyzed the size and the degree distribution of the TLC graph: number of nodes, number of edges, and the distribution of node in-degrees (number of incoming trips) and out-degrees (number of outgoing trips). We computed these at 10-meter, 100-meter and 1000-meter geographic resolution, with respect to the entire graph and to its monthly subsets, as follows:

1. Load the cleaned TLC dataset at 10-meter resolution.
2. Zoom out temporally to twelve 1-month windows (for per-month statistics), or to a single 12-month window (for full-graph statistics).
3. Keep original location nodes (at 10-meters), or create coarser nodes at 100-meter and 1000-meter resolution.
4. Use aggregate messages to compute in-degree and out-degree of each node in each temporal window.
5. Identify hotspots: top-k nodes by in-degree and out-degree.

Hotspot analysis took between 15 and 30 minutes to execute end-to-end on a year worth of TLC data, using a cluster of modest size (see Section 2.2).

Popular routes. We analyzed the frequency with which multiple trips originate from the same location at the same time, and arrive at the same location. We computed these at 100-meter and 1000-meter geographic resolution, and at 10-minute temporal resolution. At 100-meter geographic resolution, two locations are considered the same if they fall (approximately) within one NYC city block, or about 1 minute walking distance — a reasonable distance to walk to a shared ride or to a bus. Analysis at 1000-meter resolution was done to get a sense of the general per-neighborhood trends of taxi utilization, with high utilization pointing to lack of availability of convenient public transportation options.

Our choice of 10-minute temporal resolution means that a passenger would wait at most 10 minutes, and on average 5 minutes, for a shared ride or for a bus — a reasonable waiting time if it leads to significant cost savings. This analysis was conducted as follows:

1. Load the cleaned TLC dataset at 10-meter resolution.
2. Zoom out temporally to 10-minute windows. The effect of this operation is that all trips that start (have their pick-up time) within a given 10-minute window will have their pick-up time adjusted to the beginning to the window.
3. Execute a SQL group-by query that computes, for a pair of locations *source* and *dest*, and for a trip start time *start*, the number of originating trips, the total number of passengers, total cost of all trips, and combined trip duration. This step is done by instantiating a Spark Dataset from the Edge RDD of the evolving graph, and passing the result to SparkSQL to compute the result, using the following SQL query:

```
SELECT source, dest, start,
       count(*) as num_trips,
       sum(passengers) as total_passengers,
       sum(cost) as total_cost,
       sum(duration) as total_duration
FROM   edgeDF
GROUP BY source, dest, start
ORDER BY num_trips DESC
```

Popular routes analysis is computationally demanding because of the fine temporal granularity of the data (10 minutes). For this analysis, we used a month worth of TLC data, for to March 2016.

3 Results

Hotspots. Manhattan is dense, and the number of distinct pick-up and drop-off locations varies by about an order of magnitude as we coarsen geographic resolution by a factor of 10: There are 1,957,882 distinct locations at 10-meter resolution, 193,676 at 100-meter resolution, and 11,540 at 1,000-meter resolution. Our cleaned 12-month TLC dataset contains a total of 132,765,961 taxi trips between these locations. The number of trips does not depend on the geographic resolution, and remains the same in all cases.

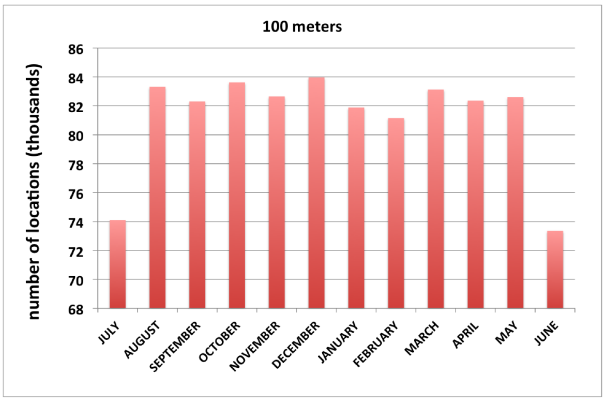

Fig. 2. Number of unique locations per month, in thousands, at 100-meter resolution.

Figure 2 shows the monthly break-down of the number of distinct locations (in thousands) at 100 meter resolutions. (The trends for 10-meter and 1000-meter resolutions were similar.) We observe that the number of locations does not vary significantly month-to-month; it ranges between 882,883 and 1,023,119 for 10-meter resolution, between 74,099 and 83,115 for 100-meter resolution, and

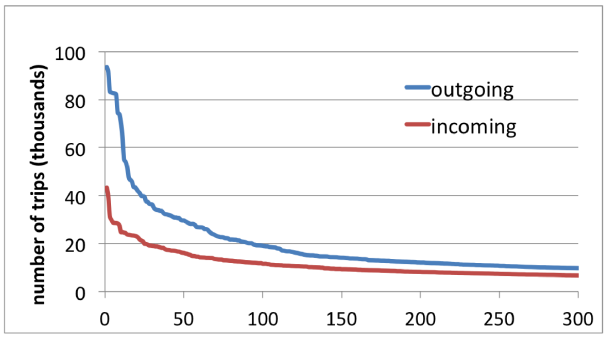

Fig. 3. Degree distribution for top-300 locations, at 10-meter resolution.

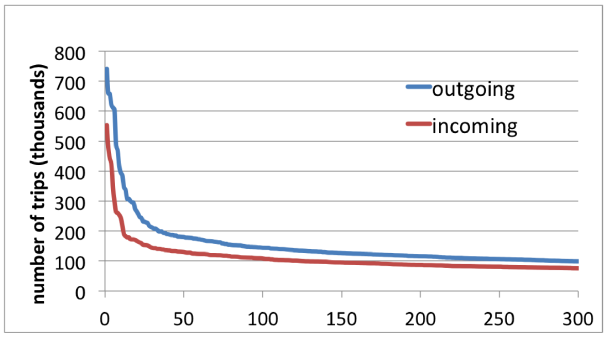

Fig. 4. Degree distribution for top-300 locations, at 100-meter resolution.

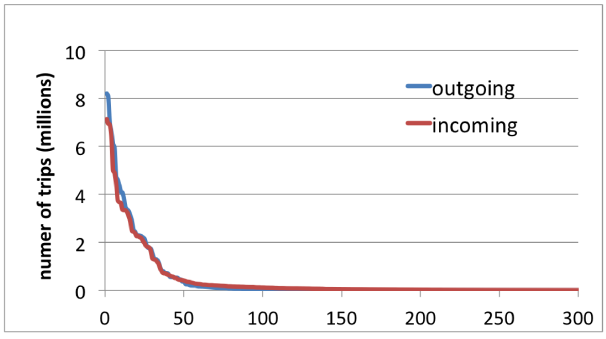

Fig. 5. Degree distribution for top-300 locations, at 1000-meter resolution.

between 4793 and 5734 for 1000-meter resolution; is lowest in June and July, and highest in March for 10 and 100 meters, and in December for 1000 meters. Month-to-month variability in the number of trips follows a similar trend as the monthly variability in the number of distinct locations.

Figures 3, 4 and 5 present the distribution of node in-degrees (number of arriving trips) and out-degrees (number of originating trips) for the top-300 such nodes, for different geographic resolutions. We observe that the distributions are power law, and that they are steeper for coarser resolution levels, as expected. Even for finer geographic granularity, locations with the highest in-degree and out-degree are responsible for a very significant proportion of the edges. At 10-meter resolution, the top-5 locations by out-degree carry 434,540 edges, or about 0.3% of all edges. At 100-meter resolution, which corresponds to 1 NYC city block and is perhaps more intuitively meaningful, the top-5 locations have combined out-degree of 3,291,470, or 2.5% of all edges. At 1000-meter resolution, the top-5 locations by out-degree are responsible for originating 36,038,808 trips, 27% of all taxi trips in NYC!

What are the hotspots in NYC taxi data? Penn Station appears at the top-5 as both an origin and a destination at all resolution levels. In fact, for 10-meter resolution, 3 of the 5 most frequent destinations are in the immediate vicinity of Penn Station. Penn Station also appears as one of the top-5 origins at 10-meter resolution, along with JFK (once) and LGA (three times) airports. The same locations appear as hotspots at 100-meter resolution, although in a different proportion. Top-5 origins at 1000-meter resolution include Penn Station, Midtown East, Rockefeller Center, Bryant Park / New York Public Library, and the Port Authority But Terminal. Top-5 destinations are the same, but Port Authority is replaced by the Flatiron district.

Interestingly, JFK and LGA airports no longer appear among the top-5 at 1000-meter resolution. This can be explained by the fact that Manhattan is dense, and while few individual city blocks receive as much taxi traffic as an outer-borough airport, they do jointly carry more traffic. While this finding is not altogether surprising, it argues for the need to consider evolving graphs such as the one we are studying in this paper at different resolution levels.

Popular routes. We analyzed the frequency with which multiple trips originate from the same location at the same time, and arrive at the same location. This analysis was done at 100-meter and 1000-meter resolution, and trips were considered to start simultaneously if they started within the same 10-minute window. This analysis was done over one month worth of data, for March 2016.

A route is a pair of locations that serve as the origin and the destination of a taxi trip. In our analysis in the remainder of this section, we removed routes in which origin and destination correspond to the same geographic location (by latitude and longitude, at the appropriate level of geographic resolution).

Figure 1 visualizes this data at 100-meter resolution on a map of NYC, while Figure 6 zooms in on Midtown Manhattan, the area with the highest number of frequent routes. In these visualizations, red lines correspond to the highest

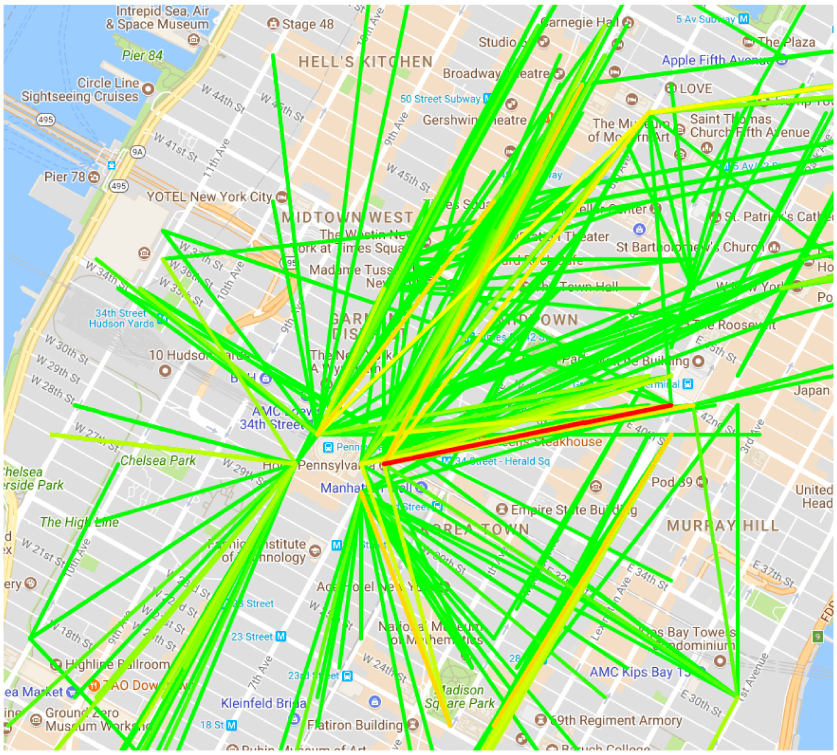

Fig. 6. The Midtown Manhattan portion of the top-300 most frequent routes in March 2016, at 100-meters. Zoomed-in version of Figure 1.

number of trips, while green is relatively lower. Note, that even the green lines connect pairs of locations with around 200 taxi trips between them.

As is apparent from this visualization, taxi trips between Penn Station and Grand Central are by far the most frequent (connected by a red line, and by multiple yellow lines). Penn Station and Grand Central are two train stations that are in close geographic proximity: they are within a 25-minute walk from each other. However, these hubs are not connected by a direct subway line, and so a subway trip between these two locations takes about 20 minutes and requires changing subway lines at the very busy Times Square station.

Another striking insight from this analysis, which can be seen on the map in Figure 1, is that there are many more taxi trips between Midtown Manhattan and LGA airport than JFK airport. This is surprising because LGA is a much smaller airport: In 2016 it handled 29.8 million passengers, compared to 59.0 million at JFK. The most likely reason for the difference is that, unlike JFK, LGA is not reachable by subway. In fact, the only public transportation option to LGA is a bus that runs from Upper Manhattan. While ride-sharing solutions and convenient public transportation options will not meet the needs

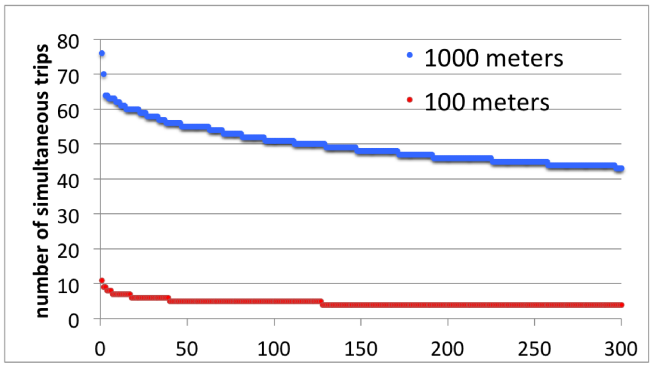

Fig. 7. Top-300 routes with highest number of simultaneous trips in March 2016, at 100-meter and 1000-meter resolution.

of all travelers, having these options will help alleviate traffic congestion and reduce transportation cost for many, as suggested by the comparatively lower JFK taxi utilization.

Finally, we analyzed the number of simultaneous frequent routes: pairs of locations that were connected by multiple simultaneous taxi trips. Figure 7 presents present summary statistics about frequent routes for March 2016, at 100-meter and 1000-meter resolution. At 100-meter resolution, there were 55,401 routes with two or more simultaneous taxi trips. Of these, 2,049 routes had 3 simultaneous trips at some point during that month, 262 had 4 simultaneous trips, and one source-destination pair had as many as 11 simultaneous trips. At 1000-meter resolution, the two most frequent routes had 76 and 70 simultaneous trips at some point during March 2016. Based on our analysis, the vast majority of popular routes are taken by cabs with 1 or 2 passengers — a clear ride-sharing opportunity. Ride sharing is warranted particularly for the routes with 3 or more simultaneous trips.

4 Conclusions and Lessons Learned

In this paper we presented an exploration of the NYC TLC yellow cab dataset, and identified hotspots and frequent routes. We analyzed this data at different levels of temporal and geographic resolution using an open-source library called Portal. We plan to analyze this data further, determining persistence of hotspots and of frequent routes over time, and considering daily, weekly and seasonal trends. We also plan to integrate taxi data with public transportation data, and to validate our methods on datasets from other cities.

Portal is, to the best of our knowledge, the only open-source library that supports analysis of *evolving property graphs*. The abstractions provided by Portal allow implementing use cases such as those presented here with a handful of lines of code, and automatically handle temporal semantics. The biggest challenge in

implementing these use cases, in terms of performance, was dealing with the high evolution rate in our dataset, since every taxi ride is technically an evolution event. We had to carefully chose the level of temporal granularity that is both appropriate for our data analysis scenario and is possible to compute within a reasonable amount of time.

References

1. M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, and M. Zaharia. Spark SQL: relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 1383–1394, 2015.
2. M. H. Böhlen, C. S. Jensen, and R. T. Snodgrass. Temporal Statement Modifiers. *ACM Transactions on Database Systems*, 25(4):407–456, 2000.
3. J. Dean and S. Ghemawat. MapReduce: Simplied Data Processing on Large Clusters. *Proceedings of 6th Symposium on Operating Systems Design and Implementation*, pages 137–149, 2004.
4. N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva. Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2149–2158, 2013.
5. J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica. GraphX: Graph processing in a distributed dataflow framework. In *11th USENIX Symposium on Operating Systems Design and Implementation, OSDI '14, Broomfield, CO, USA, October 6-8, 2014.*, pages 599–613, 2014.
6. V. Z. Moffitt and J. Stoyanovich. Temporal graph algebra. In *Proceedings of The 16th International Symposium on Database Programming Languages, DBPL 2017, Munich, Germany, September 1, 2017*, pages 10:1–10:12, 2017.
7. NYC Taxi and Limousine Commission. TLC trip record data, 2018. `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`.
8. D. Peuquet. It's about time: A conceptual framework for the representation of temporal dynamics in geographic information systems. *Annals of the Association of American Geographers*, 84(3):441–461, 1994.
9. M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012, San Jose, CA, USA, April 25-27, 2012*, pages 15–28, 2012.