Realistic Transport Simulation: Tackling the Small Data Challenge with Open Data

Guimu Guo
Department of Computer Science
The University of Alabama at Birmingham
Birmingham, AL, United States
guomuguo@uab.edu

Jalal Majed Khalil
Department of Computer Science
The University of Alabama at Birmingham
Birmingham, AL, United States
jalalk@uab.edu

Da Yan
Department of Computer Science
The University of Alabama at Birmingham
Birmingham, AL, United States
yanda@uab.edu

Virginia Sisiopiku
School of Engineering
The University of Alabama at Birmingham
Birmingham, AL, United States
vsisiopi@uab.edu

***Abstract*—MATSim is the state-of-the-art open source software for agent-based transport simulation, intended for use to evaluate transportation planning models. A standard approach to use MATSim is to conduct a user survey about their day-plans of travel, from which a synthetic dataset of agents' day-plans for an entire region is generated for transport simulation. The simulation output can be used for various evaluations, such as congestion conditions of road segments and their peak hours.**

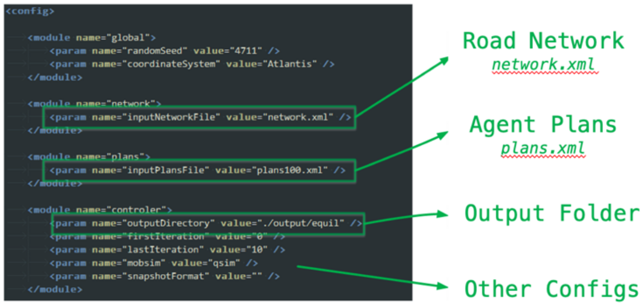
This paper aims to conduct a transportation simulation on MATSim for the region of Birmingham, AL. A traditional approach based on Iterative Proportional Fitting (IPF) is not sufficient for generating a realistic synthetic population due to the small data problem: Birmingham is a small city with limited transport data statistics, and we only have a survey of 451 people for their day-plans. To tackle the small data problem, we seek the assistance of abundant open data such as US Census data, OpenStreetMap, OpenAddresses and Birmingham Business Alliance to complete the fine details realistically. We also utilize various data science and machine learning techniques to build models that utilize these open data to generate a realistic population. Preliminary tests demonstrate reasonable accuracy of the simulation results.

Keywords*—*transport, simulation, MATSim, small data, open data

I. Introduction

MATSim (Multi-Agent Transport Simulation) [1] is a software project started around 2006 with the goal of generating traffic and congestion patterns by following individual synthetic travelers through their daily or weekly activities. It has since then evolved from a collection of stand-alone C++ programs to an integrated Java-based framework which is publicly hosted, open-source available, automatically regression tested [2]. MATSim has become the de facto standard for urban transport simulation, and has been used all over the world [3].

MATSim takes as input (1) the travel plans of a population, and (2) the underlying road network, and simulates the day-plans of the population over the road network. Figure 1 shows the job configuration file for a MATSim simulation where we can see these two data sources along with other configuration parameters, and Figure 2 shows a snapshot of the simulation using OTFVis, MATSim's Visualization Tool.



```
<config>

    <module name="global">
        <param name="randomSeed" value="4711" />
        <param name="coordinateSystem" value="Atlantis" />
    </module>

    <module name="network">
        <param name="inputNetworkFile" value="network.xml" />
    </module>

    <module name="plans">
        <param name="inputPlansFile" value="plans100.xml" />
    </module>

    <module name="controler">
        <param name="outputDirectory" value="./output/equil" />
        <param name="firstIteration" value="0" />
        <param name="lastIteration" value="10" />
        <param name="mobsim" value="qsim" />
        <param name="snapshotFormat" value="" />
    </module>
```

Figure 1: MATSim Job Configuration File

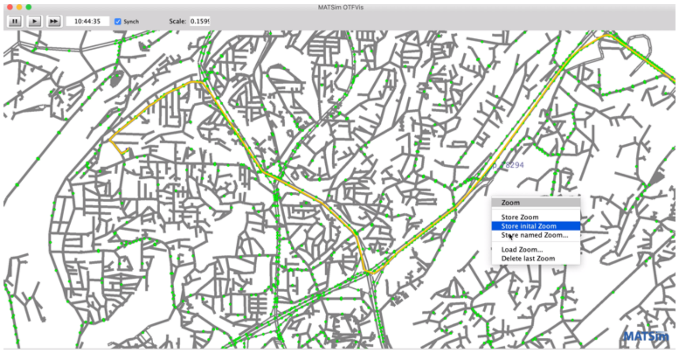
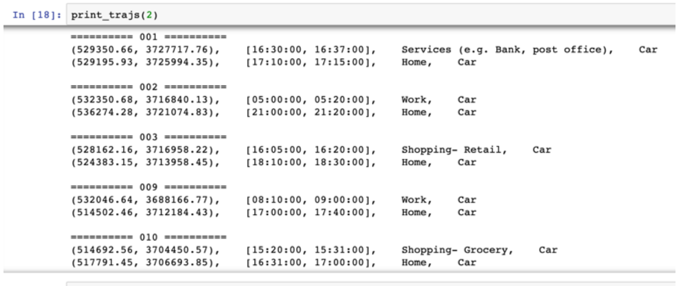


Figure 2: A Snapshot of the Birmingham Traffic in OTFVis

The road network can be obtained from OpenStreetMap [4], and so the key problem is to prepare the travel plans of a population, which is usually generated from a survey of a sample of the population, scaled by Iterative Proportional Fitting (IPF) [6] according to the distribution and correlation of the agents' attributes in the census data. However, applying IPF requires a large population sample to be statistically representative, which is often too costly to obtain.

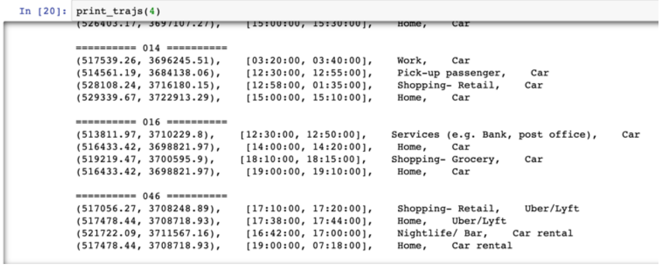
Supported by the STRIDE (Southeastern Transportation Research, Innovation, Development & Education) Center, we have conducted an online survey (with motivating rewards) where people in Birmingham report their day-plans as a sequence of legs, each associated with the destination location, the start and end time of the leg, the activity at the destination, and the travel mode (e.g., driving by car, or using Uber/Lyft). The initial location and activity of the day are also provided.

978-1-7281-0858-2/19/$31.00 © 2019 IEEE



```
In [18]: print_trajs(2)
         ========== 001 ==========
         (529350.66, 3727717.76),    [16:30:00, 16:37:00],    Services (e.g. Bank, post office),    Car
         (529195.93, 3725994.35),    [17:10:00, 17:15:00],    Home,    Car

         ========== 002 ==========
         (532350.68, 3716840.13),    [05:00:00, 05:20:00],    Work,    Car
         (536274.28, 3721074.83),    [21:00:00, 21:20:00],    Home,    Car

         ========== 003 ==========
         (528162.16, 3716958.22),    [16:05:00, 16:20:00],    Shopping- Retail,    Car
         (524383.15, 3713958.45),    [18:10:00, 18:30:00],    Home,    Car

         ========== 009 ==========
         (532046.64, 3688166.77),    [08:10:00, 09:00:00],    Work,    Car
         (514502.46, 3712184.43),    [17:00:00, 17:40:00],    Home,    Car

         ========== 010 ==========
         (514692.56, 3704450.57),    [15:20:00, 15:31:00],    Shopping- Grocery,    Car
         (517791.45, 3706693.85),    [16:31:00, 17:00:00],    Home,    Car
```

Figure 3: Examples of Persons with 2 Legs



```
In [20]: print_trajs(4)
         (526403.17, 3697107.27),    [15:00:00, 15:30:00],    Home,    Car

         ========== 014 ==========
         (517539.26, 3696245.51),    [03:20:00, 03:40:00],    Work,    Car
         (514561.19, 3684138.06),    [12:30:00, 12:55:00],    Pick-up passenger,    Car
         (528108.24, 3716180.15),    [12:58:00, 01:35:00],    Shopping- Retail,    Car
         (529339.67, 3722913.29),    [15:00:00, 15:10:00],    Home,    Car

         ========== 016 ==========
         (513811.97, 3710229.8),    [12:30:00, 12:50:00],    Services (e.g. Bank, post office),    Car
         (516433.42, 3698821.97),    [14:00:00, 14:20:00],    Home,    Car
         (519219.47, 3700595.9),    [18:10:00, 18:15:00],    Shopping- Grocery,    Car
         (516433.42, 3698821.97),    [19:00:00, 19:10:00],    Home,    Car

         ========== 046 ==========
         (517056.27, 3708248.89),    [17:10:00, 17:20:00],    Shopping- Retail,    Uber/Lyft
         (517478.44, 3708718.93),    [17:38:00, 17:44:00],    Home,    Uber/Lyft
         (521722.09, 3711567.16),    [16:42:00, 17:00:00],    Nightlife/ Bar,    Car rental
         (517478.44, 3708718.93),    [19:00:00, 07:18:00],    Home,    Car rental
```

Figure 4: Examples of Persons with 4 Legs

Figures 3 and 4 show the leg sequences for survey participants whose day-plans contain 2 and 4 legs, respectively. We see day-plans like Home-Work-Home (e.g., Person 002) and Home-Services-Home-Shopping-Home (e.g., Person 016).

Note that differentiating different mode of traffic allows more information in the simulation, as we can check how much traffic flow on each road segment is caused by ridesharing apps, and further to see whether they are helping bridge the gap of insufficient public transport, or are disturbing the traffic.

Overall, we obtained 451 people and their day-plans that are of reasonable quality. The lengths of leg sequences are distributed as shown in Figure 5. We can see that day-plans with 2 legs are the most common (e.g., Home-Work-Home), and most day-plans are within 5 legs.

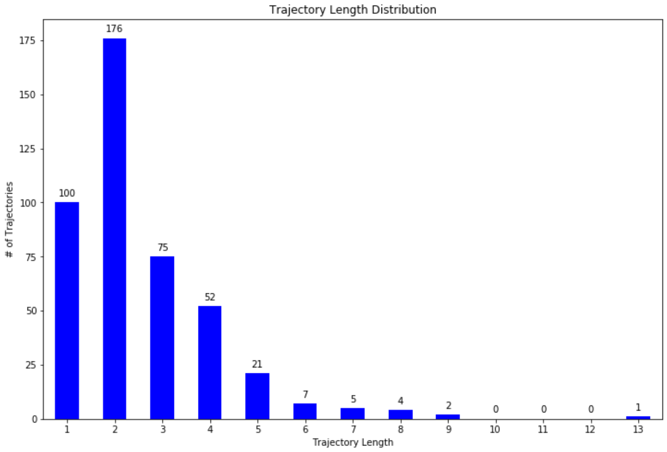
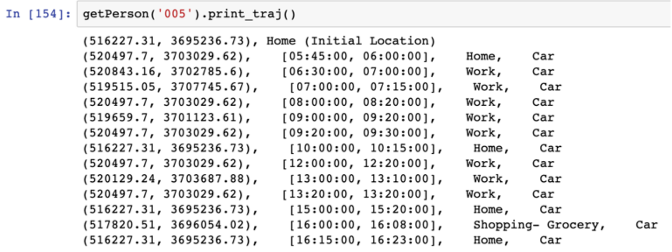
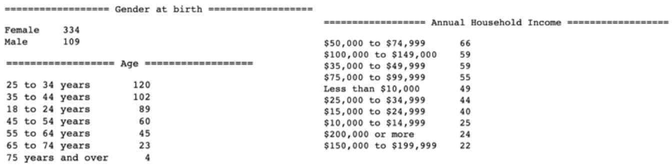


Figure 5: The Distribution of the Number of Legs



```
In [154]: getPerson('005').print_traj()
          (516227.31, 3695236.73), Home (Initial Location)
          (520497.7, 3703029.62),    [05:45:00, 06:00:00],    Home,    Car
          (520843.16, 3702785.6),    [06:30:00, 07:00:00],    Work,    Car
          (519515.05, 3707745.67),    [07:00:00, 07:15:00],    Work,    Car
          (520497.7, 3703029.62),    [08:00:00, 08:20:00],    Work,    Car
          (519659.7, 3701123.61),    [09:00:00, 09:20:00],    Work,    Car
          (520497.7, 3703029.62),    [09:20:00, 09:30:00],    Work,    Car
          (516227.31, 3695236.73),    [10:00:00, 10:15:00],    Home,    Car
          (520497.7, 3703029.62),    [12:00:00, 12:20:00],    Work,    Car
          (520129.24, 3703687.88),    [13:00:00, 13:10:00],    Work,    Car
          (520497.7, 3703029.62),    [13:20:00, 13:20:00],    Work,    Car
          (516227.31, 3695236.73),    [15:00:00, 15:20:00],    Home,    Car
          (517820.51, 3696054.02),    [16:00:00, 16:08:00],    Shopping- Grocery,    Car
          (516227.31, 3695236.73),    [16:15:00, 16:23:00],    Home,    Car
```

Figure 6: The Legs of a Potential Uber/Lyft Driver



```
================== Gender at birth ==================
                                                   ================== Annual Household Income ==================
Female    334
Male      109                                      $50,000 to $74,999      66
                                                   $100,000 to $149,000    59
================== Age ==================          $35,000 to $49,999      59
                                                   $75,000 to $99,999      55
25 to 34 years       120                           Less than $10,000       49
35 to 44 years       102                           $25,000 to $34,999      44
18 to 24 years        89                           $15,000 to $24,999      40
45 to 54 years        60                           $10,000 to $14,999      25
55 to 64 years        45                           $200,000 or more        24
65 to 74 years        23                           $150,000 to $199,999    22
75 years and over      4
```

Figure 7: Demographic Attributes in the Survey

Interestingly, we also find a day-plan with 13 legs which appears to be an outlier, but this survey participant could be an Uber/Lyft driver, whose day-plan is shown in Figure 6.

Demographic attributes are also collected, 3 of which are shown in Figure 7. These attributes are usually used in IPF to align with those of the entire population, whose marginal distributions are obtained from US Census Bureau [6].

However, we find that 451 participates are insufficient for this purpose, as we have the data sparsity problem. Assuming that only the 3 demographic attributes shown in Figure 7 are considered for IPF, then there are 2 x 7 x 10 = 140 possible combinations of attribute values. Given the 451 survey participants, on average each combination will have around 3 persons, which is a too small sampling pool to allow diversity when scaled up to the population scale, i.e., many people will share the same source and destination locations as well as departure time. In fact, there misses any participant in a lot of combinations of demographic attributes.

As a result, only geospatial information is considered for population generation.

Challenges. Even so, there are still 2 challenges remaining:

(1) For user-designated geographic regions, IPF only generates the number of people (e.g., residents) in them, and there is no information on where these people are, and on what activities they conduct and when they conduct these activities.

(2) Given the number of people generated above, denoted by *n*, a conventional population generation algorithm simply samples the pool of survey participates with the designated attribute values (regions here) for *n* day-plans [5]. While this approach compensates for the lack of spatial-temporal information on user activities, it requires a survey on many people. With only 451 people surveyed, each person's day-plan could be sampled many times. This leads to a synthetic population where many people commute to work from the same house to the same office at the same time, which gives rise to unrealistic traffic congestions.

Solution. To enable a more realistic simulation, we adopt a data-driven approach to model the different aspects of travel including (1) time, (2) location, (3) activities, and (4) mode. Since our survey data is small, we enrich it with public data in order to generate a more realistic population for Birmingham with realistic day-plans.

This is an example of Data Science where a domain-specific problem (transport simulation for planning) is solved by data-driven analytics and machine learning.

This is also an example of how to tackle the Small Data problem [7] using external data sources, and provides a valuable solution for small cities who wants to improve transport planning with MATSim but do not have a lot of funds for transport survey and curating transport data statistics.

Organization. The rest of this paper is organized as follows. Section II introduces our methods for population generation, and Section III reports some evaluation and visualization results with MATSim. Finally, Section IV concludes this paper.

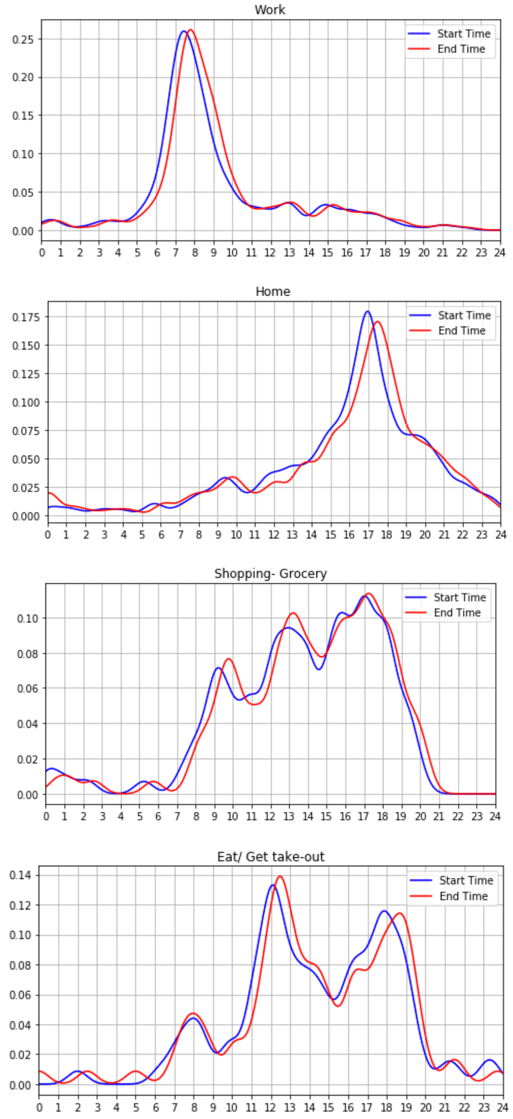


Figure 8: PDFs of *start_time* and *end_time* Obtained by Kernel Density Estimation (KDE)

II. Methods

This section presents our data-driven approach to model the different aspects of travel including (1) time, (2) location, (3) activities, and (4) mode; we then present how we use these models to generate a realistic population for Birmingham.

A. Time Modeling

Currently, we model the time aspects of a leg as a pair (*start_time*, *time_span*) which can derive other information such as: *end_time* = *start_time* + *time_span*.

We consider both *start_time* and *time_span* as random variables that follow a distribution conditional on the specific activity adopted. Once the activity of a leg is determined, we can sample (*start_time*, *time_span*) from the distribution for that activity to timestamp that leg.

To illustrate why conditioning time on activity makes good sense, we plot the probability density functions (PDFs) of *start_time* and *end_time* for 4 activities "Work", "Home", "Shopping-Grocery" and "Eat/Get take-out" in Figure 8 for illustration. These PDFs are estimated from the survey data using kernel density estimation (KDE). For example, to get the PDF of *start_time* for the activity "Work", we collect the start time of all legs in the survey whose destination activity is "Work", which are then input to the KDE model to fit a PDF.

We can see that Figure 8 aligns well with our intuition. For example, the time to "Work" peaks at 8 am and the time back "Home" peaks at 5 pm. Also, the "Eat/Get take-out" time peaks at noon and right after work, and "Shopping" happens mostly between 9 am and 8 pm.

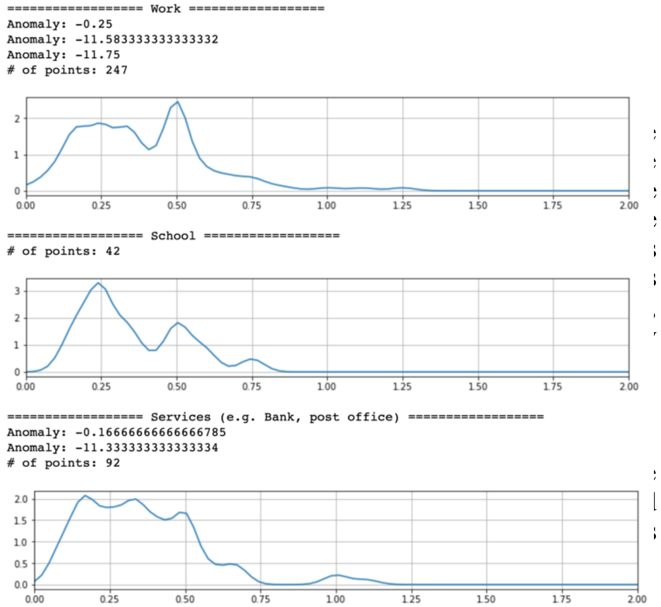

```
================== Work ==================
Anomaly: -0.25
Anomaly: -11.583333333333332
Anomaly: -11.75
# of points: 247

================== School ==================
# of points: 42

================== Services (e.g. Bank, post office) ==================
Anomaly: -0.16666666666666785
Anomaly: -11.333333333333334
# of points: 92
```

Figure 9: PDFs of *time_span* Obtained by KDE

Figure 9 shows the PDF of *time_span* for 3 activities, where the *x*-axis is in the unit of hours. We can see that most legs finish within 1 hour, which is within expectation as the survey is conducted for the Birmingham area only (Jefferson county + Shelby county).

There is, however, still some room to further improve the time modeling. For example, we can include the distance between the source and destination locations as another factor that impacts travel time (i.e., *time_span*). As a future work, we plan to model *time_span* as a regression function of the travel distance which can be learned from the survey data for each activity, to enable more realistic modeling.

B. Location Modeling: A ZCTA-based Approach

Public data such as census data usually report aggregate statistics for individual geographic regions, and thus we need to align the surveyed locations with these geographic regions in order to scale up the population and their locations in each region. Since we are only interested in two counties, Jefferson and Shelby, a county-level granularity is too coarse for an accurate modeling. We thus choose the granularity of ZIP Code Tabulation Areas (ZCTAs). Figure 10 below shows the counties and ZCTAs in Alabama, which are obtained from TIGERweb [8].

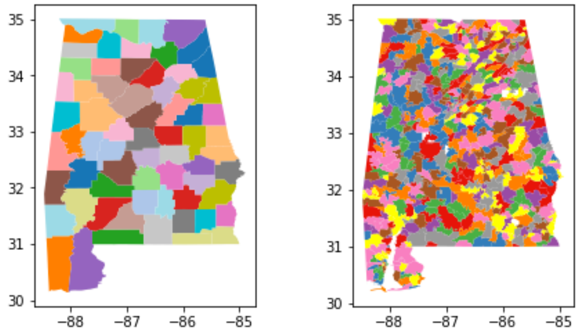


Figure 10: Counties (Left) and ZCTAs (Right) in Alabama

Since we are only interested in Jefferson and Shelby counties, we first partition the surveyed locations by ZCTAs as shown in Figure 11, where the brighter a ZCTA is, the more surveyed locations it contains.

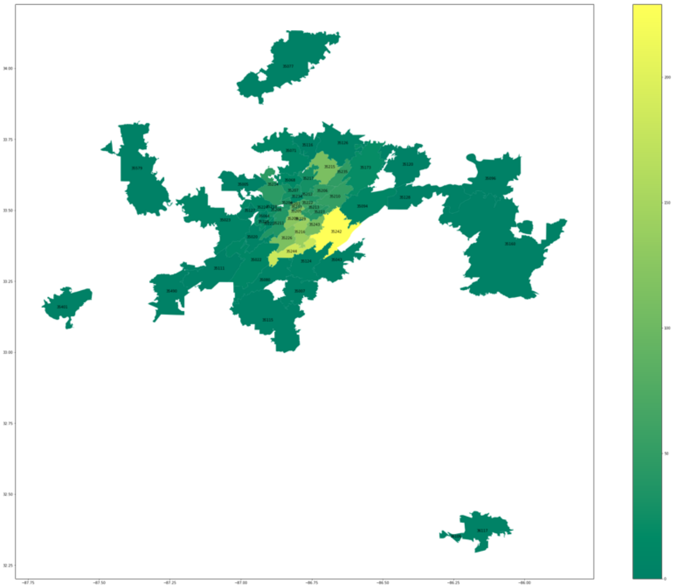
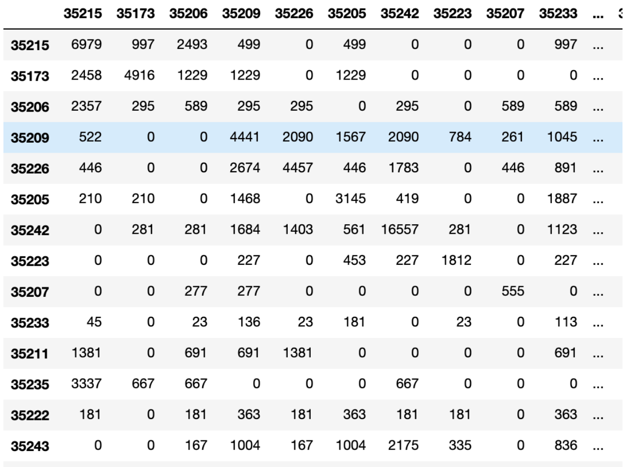


Figure 11: ZCTAs with Surveyed Locations (with Zip Codes)

In order to scale up the population from the seed survey, we can create a seed matrix A where each element $A[i][j]$ indicates how many legs have source in ZCTA i and destination in ZCTA j.



	35215	35173	35206	35209	35226	35205	35242	35223	35207	35233	...
35215	6979	997	2493	499	0	499	0	0	0	997	...
35173	2458	4916	1229	1229	0	1229	0	0	0	0	...
35206	2357	295	589	295	295	0	295	0	589	589	...
35209	522	0	0	4441	2090	1567	2090	784	261	1045	...
35226	446	0	0	2674	4457	446	1783	0	446	891	...
35205	210	210	0	1468	0	3145	419	0	0	1887	...
35242	0	281	281	1684	1403	561	16557	281	0	1123	...
35223	0	0	0	227	0	453	227	1812	0	227	...
35207	0	0	277	277	0	0	0	0	555	0	...
35233	45	0	23	136	23	181	0	23	0	113	...
35211	1381	0	691	691	1381	0	0	0	0	691	...
35235	3337	667	667	0	0	0	667	0	0	0	...
35222	181	0	181	363	181	363	181	181	0	363	...
35243	0	0	167	1004	167	1004	2175	335	0	836	...

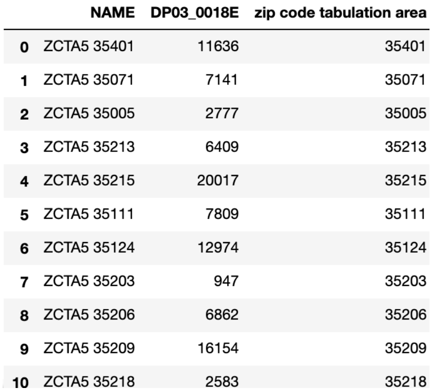
Figure 12: A Fragment of the Adjusted Seed Matrix by IPF

We can scale up the travel plans of the population for different activities individually. For example, for "Work" which is often a major factor for traffic congestion, we can obtain marginals such as how many people commute to work from each ZCTA (i.e., they live there), and how many people commute to work at each ZCTA (i.e., they work there) from public data. Assuming that the seed matrix A is also constructed out of only those legs whose activity is "Work", then we can adjust A to align with the marginals using IPF as illustrated in Figure 12.

To get the marginals of how many people commute to work from home in each ZCTA, we can use American Community Survey (ACS) [9], more specifically, variable "P03_0018E: COMMUTING TO WORK" which indicates how many people commute to work at each region. This can be obtained using the Census API of ACS; for example, in our scenario we used the following URL request:

https://api.census.gov/data/2017/acs/acs5/profile?get=NAME,DP03_0018E&for=zip%20code%20tabulation%20area:35215,35173,35206,35209,35226,35205,35242,35223,35207,35233,35211,35235,35222,35243,35203,35216,35217,35244,35208,35210,35214,35213,35218,35221,35204,35228,35212,35094,35234,35224,35071,36117,35023,35022,35020,35126,35043,35401,35096,35080,35490,35068,35124,35116,35128,35229,35064,35077,35115,35160,35007,35579,35127,36106,35120,35005,35111

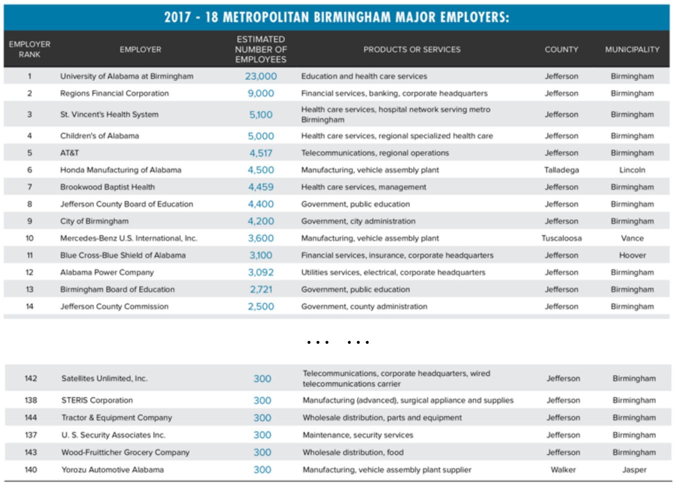
The returned data is illustrated in Figure 13 below. For example, ZCTA 35401 has 11,636 people commuting to work.



	NAME	DP03_0018E	zip code tabulation area
0	ZCTA5 35401	11636	35401
1	ZCTA5 35071	7141	35071
2	ZCTA5 35005	2777	35005
3	ZCTA5 35213	6409	35213
4	ZCTA5 35215	20017	35215
5	ZCTA5 35111	7809	35111
6	ZCTA5 35124	12974	35124
7	ZCTA5 35203	947	35203
8	ZCTA5 35206	6862	35206
9	ZCTA5 35209	16154	35209
10	ZCTA5 35218	2583	35218

Figure 13: Variable DP03_0018E Values for Selected ZCTAs

Unfortunately, even though ACS has another variable “B08604_001E: WORKER POPULATION FOR WORKPLACE GEOGRAPHY”, it only delves into the county level, and the ZCTA level data is missing (nulls are returned). We have to obtain the related data from other sources such as the major employer list from Birmingham Business Alliance (BBA) [10], as shown in Figure 14.



2017 - 18 METROPOLITAN BIRMINGHAM MAJOR EMPLOYERS:

EMPLOYER RANK	EMPLOYER	ESTIMATED NUMBER OF EMPLOYEES	PRODUCTS OR SERVICES	COUNTY	MUNICIPALITY
1	University of Alabama at Birmingham	23,000	Education and health care services	Jefferson	Birmingham
2	Regions Financial Corporation	9,000	Financial services, banking, corporate headquarters	Jefferson	Birmingham
3	St. Vincent's Health System	5,100	Health care services, hospital network serving metro Birmingham	Jefferson	Birmingham
4	Children's of Alabama	5,000	Health care services, regional specialized health care	Jefferson	Birmingham
5	AT&T	4,517	Telecommunications, regional operations	Jefferson	Birmingham
6	Honda Manufacturing of Alabama	4,500	Manufacturing, vehicle assembly plant	Talladega	Lincoln
7	Brookwood Baptist Health	4,459	Health care services, management	Jefferson	Birmingham
8	Jefferson County Board of Education	4,400	Government, public education	Jefferson	Birmingham
9	City of Birmingham	4,200	Government, city administration	Jefferson	Birmingham
10	Mercedes-Benz U.S. International, Inc.	3,600	Manufacturing, vehicle assembly plant	Tuscaloosa	Vance
11	Blue Cross-Blue Shield of Alabama	3,100	Financial services, insurance, corporate headquarters	Jefferson	Hoover
12	Alabama Power Company	3,092	Utilities services, electrical, corporate headquarters	Jefferson	Birmingham
13	Birmingham Board of Education	2,721	Government, public education	Jefferson	Birmingham
14	Jefferson County Commission	2,500	Government, county administration	Jefferson	Birmingham
			… …		
142	Satellites Unlimited, Inc.	300	Telecommunications, corporate headquarters, wired telecommunications carrier	Jefferson	Birmingham
138	STERIS Corporation	300	Manufacturing (advanced), surgical appliance and supplies	Jefferson	Birmingham
144	Tractor & Equipment Company	300	Wholesale distribution, parts and equipment	Jefferson	Birmingham
137	U. S. Security Associates Inc.	300	Maintenance, security services	Jefferson	Birmingham
143	Wood-Fruitticher Grocery Company	300	Wholesale distribution, food	Jefferson	Birmingham
140	Yorozu Automotive Alabama	300	Manufacturing, vehicle assembly plant supplier	Walker	Jasper

Source: Birmingham Business Alliance (BBA)

Figure 14: Birmingham Major Employer List from BBA

In fact, the list has covered more than 35% of the employees in Birmingham, and we can generate workplaces for the other 65% employees following the same distribution. For example, we can plot the work destinations in the survey to visualize the number of components and then fit a Gaussian mixture model (GMM) over these destination locations. We can then sample the other 65% work locations from the GMM (we should actually use the nearest OpenAddresses address of each sampled location to be described next). The marginals of how many people commute to work at each ZCTA can then be obtained for IPF.

C. Location Sampling: Solving the Location Scarcity Problem

Now that we have the value for each $A[i][j]$ indicating how many legs have source in ZCTA i and destination in ZCTA j, we can generate that many legs about “Work” whose time information can be sampled from the PDFs for the “Work” activity as described earlier. However, it is still unclear how to generate the source (e.g., home) and destination (e.g., office) of each leg.

One approach is to sample a random location in the corresponding ZCTA, but this is unlikely to be realistic as many places are sparsely populated or even far from the road network. Another approach is to sample only from the surveyed locations which are real. However, given that we only have 451 participants in our survey each with a few legs, there are only slightly more than 2,000 reported locations in total. This is quite a small pool to sample from if we want to generate the source and destination locations for the entire population in Birmingham, leading to unrealistic scenarios like many people commute to work from the same house.

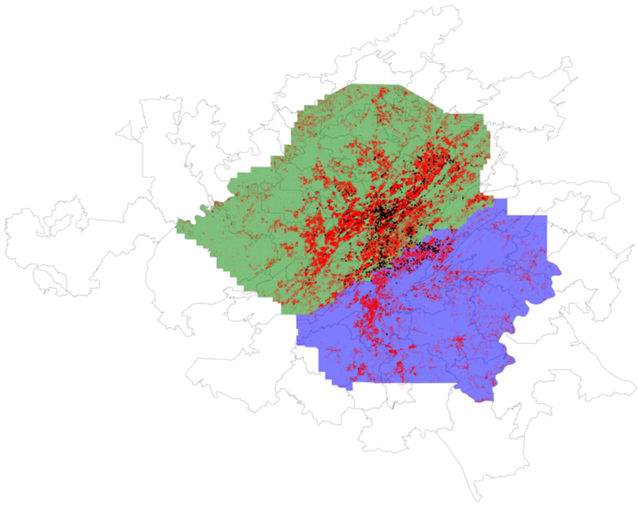


Figure 15: OpenAddresses Locations in Jefferson and Shelby Counties

Fortunately, OpenAddresses [11] collects address data which can be used to enrich our location pool for sampling. Figure 15 shows Jefferson county (green) and Shelby county (blue) with ZCTA boundaries marked (we only show those ZCTAs that intersect with any of the two counties); it also shows the addresses from OpenAddresses inside the two counties in red, and shows the surveyed locations in black. We can see that the red points significantly enrich the black ones.

In fact, the OpenAddresses locations well capture the road network and building block structures, as shown in Figure 16 which is a zoomed-in version. Moreover, our surveyed locations well align with the OpenAddresses locations: the black points appear more often at those regions where the red points are dense.

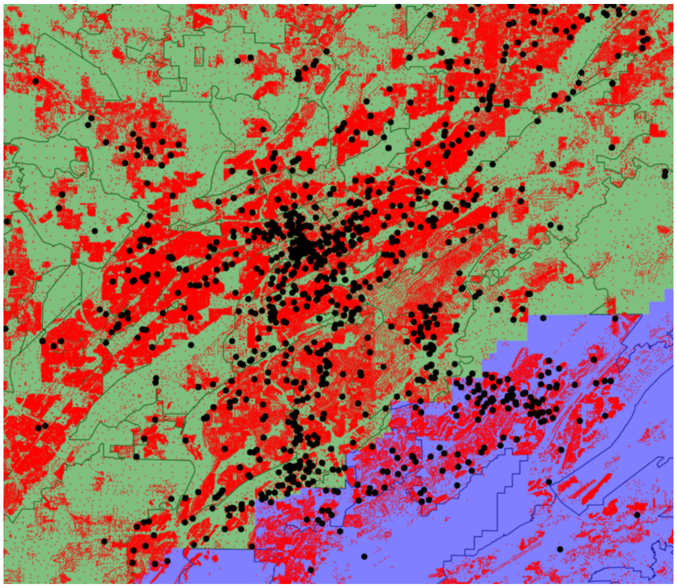


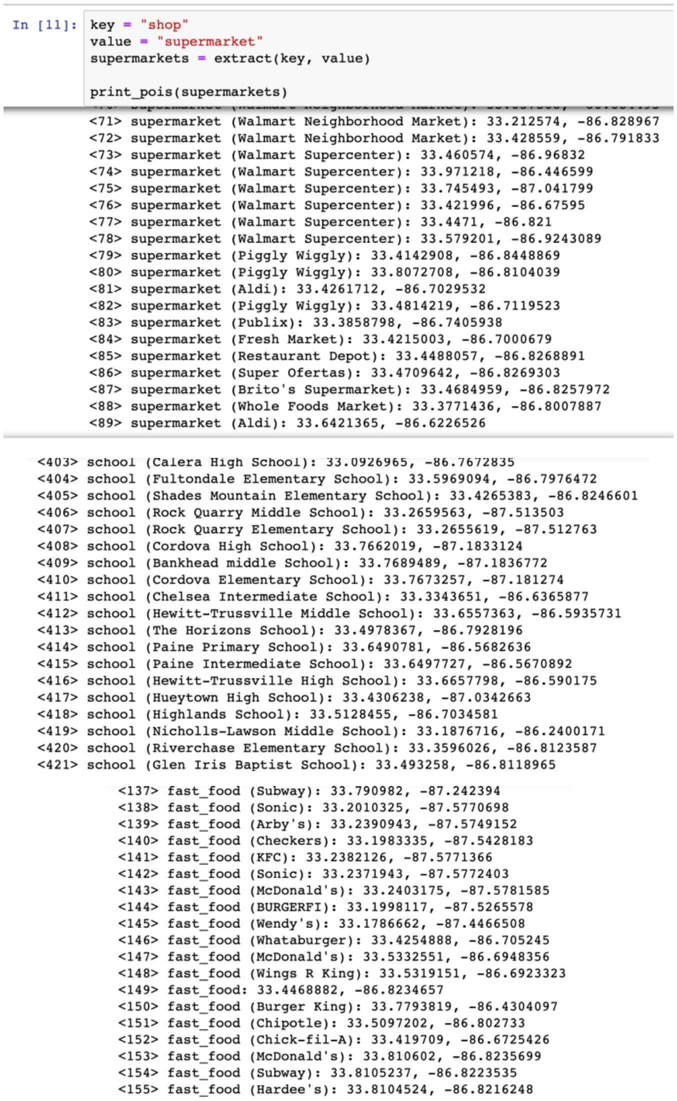
Figure 16: A Zoomed-in Version of Figure 15

Table I: Surveyed Activities and the Corresponding OpenStreetMap POI Types

Activity	OpenStreetMap POI Types
Shopping-Grocery	supermarket, convenience
Services (e.g. Bank, Post Office)	bank, atm, post_office
School	school
Shopping-Retail	department_store, mall
Eat / Get Take-out	restaurant, fast_food, café, deli, bar, pub
Nightlife / Bar	nightclub, bar, pub
Drop-off Passenger	all POIs (50%) and OpenAddresses locations (50%)
Pick-up Passenger	all POIs (50%) and OpenAddresses locations (50%)

So far, we have only discussed the activities "Home" and "Work". However, the sample pool should be different conditional on different activity. Fortunately, OpenStreetMap [12] maintains different types of points-of-interest (POIs) [13] that correspond to different activities, as listed in Table I.

Figure 17 shows some examples of POIs in types 'supermarket', 'department_store', 'fast_food' and 'school'. However, we remark that the list of POIs in each category may not be complete since OpenStreetMap is a project to share data by collaborative efforts but there is no guarantee of data completeness.



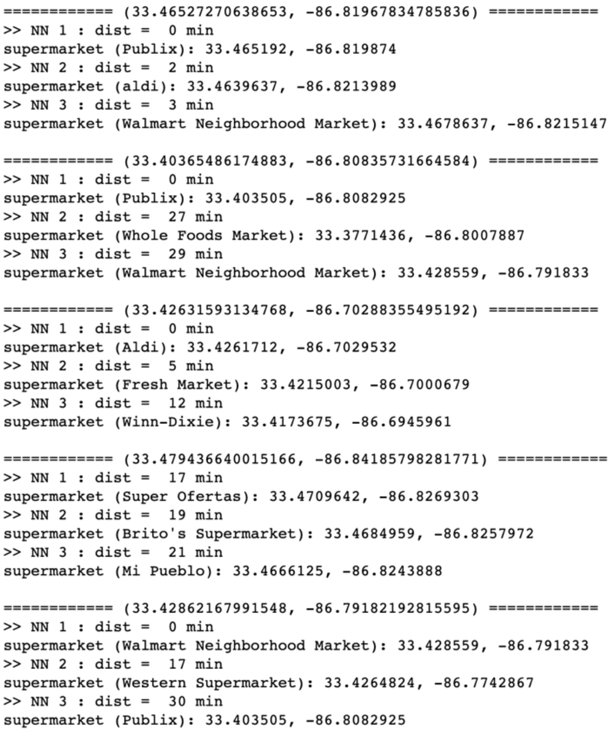
```
In [11]: key = "shop"
         value = "supermarket"
         supermarkets = extract(key, value)

         print_pois(supermarkets)
```

```
<71> supermarket (Walmart Neighborhood Market): 33.212574, -86.828967
<72> supermarket (Walmart Neighborhood Market): 33.428559, -86.791833
<73> supermarket (Walmart Supercenter): 33.460574, -86.96832
<74> supermarket (Walmart Supercenter): 33.971218, -86.446599
<75> supermarket (Walmart Supercenter): 33.745493, -87.041799
<76> supermarket (Walmart Supercenter): 33.421996, -86.67595
<77> supermarket (Walmart Supercenter): 33.4471, -86.821
<78> supermarket (Walmart Supercenter): 33.579201, -86.9243089
<79> supermarket (Piggly Wiggly): 33.4142908, -86.8448869
<80> supermarket (Piggly Wiggly): 33.8072708, -86.8104039
<81> supermarket (Aldi): 33.4261712, -86.7029532
<82> supermarket (Piggly Wiggly): 33.4814219, -86.7119523
<83> supermarket (Publix): 33.3858798, -86.7405938
<84> supermarket (Fresh Market): 33.4215003, -86.7000679
<85> supermarket (Restaurant Depot): 33.4488057, -86.8268891
<86> supermarket (Super Ofertas): 33.4709642, -86.8269303
<87> supermarket (Brito's Supermarket): 33.4684959, -86.8257972
<88> supermarket (Whole Foods Market): 33.3771436, -86.8007887
<89> supermarket (Aldi): 33.6421365, -86.6226526
```

```
<403> school (Calera High School): 33.0926965, -86.7672835
<404> school (Fultondale Elementary School): 33.5969094, -86.7976472
<405> school (Shades Mountain Elementary School): 33.4265383, -86.8246601
<406> school (Rock Quarry Middle School): 33.2659563, -87.513503
<407> school (Rock Quarry Elementary School): 33.2655619, -87.512763
<408> school (Cordova High School): 33.7662019, -87.1833124
<409> school (Bankhead middle School): 33.7689489, -87.1836772
<410> school (Cordova Elementary School): 33.7673257, -87.181274
<411> school (Chelsea Intermediate School): 33.3343651, -86.6365877
<412> school (Hewitt-Trussville Middle School): 33.6557363, -86.5935731
<413> school (The Horizons School): 33.4978367, -86.7928196
<414> school (Paine Primary School): 33.6490781, -86.5682636
<415> school (Paine Intermediate School): 33.6497727, -86.5670892
<416> school (Hewitt-Trussville High School): 33.6657798, -86.590175
<417> school (Hueytown High School): 33.4306238, -87.0342663
<418> school (Highlands School): 33.5128455, -86.7034581
<419> school (Nicholls-Lawson Middle School): 33.1876716, -86.2400171
<420> school (Riverchase Elementary School): 33.3596026, -86.8123587
<421> school (Glen Iris Baptist School): 33.493258, -86.8118965
```

```
<137> fast_food (Subway): 33.790982, -87.242394
<138> fast_food (Sonic): 33.2010325, -87.5770698
<139> fast_food (Arby's): 33.2390943, -87.5749152
<140> fast_food (Checkers): 33.1983335, -87.5428183
<141> fast_food (KFC): 33.2382126, -87.5771366
<142> fast_food (Sonic): 33.2371943, -87.5772403
<143> fast_food (McDonald's): 33.2403175, -87.5781585
<144> fast_food (BURGERFI): 33.1998117, -87.5265578
<145> fast_food (Wendy's): 33.1786662, -87.4466508
<146> fast_food (Whataburger): 33.4254888, -86.705245
<147> fast_food (McDonald's): 33.5332551, -86.6948356
<148> fast_food (Wings R King): 33.5319151, -86.6923323
<149> fast_food: 33.4468882, -86.8234657
<150> fast_food (Burger King): 33.7793819, -86.4304097
<151> fast_food (Chipotle): 33.5097202, -86.802733
<152> fast_food (Chick-fil-A): 33.419709, -86.6725426
<153> fast_food (McDonald's): 33.810602, -86.8235699
<154> fast_food (Subway): 33.8105237, -86.8223535
<155> fast_food (Hardee's): 33.8104524, -86.8216248
```

Figure 17: Examples of POIs from OpenStreetMap



```
============= (33.46527270638653, -86.81967834785836) =============
>> NN 1 : dist =  0 min
supermarket (Publix): 33.465192, -86.819874
>> NN 2 : dist =  2 min
supermarket (aldi): 33.4639637, -86.8213989
>> NN 3 : dist =  3 min
supermarket (Walmart Neighborhood Market): 33.4678637, -86.8215147

============= (33.40365486174883, -86.80835731664584) =============
>> NN 1 : dist =  0 min
supermarket (Publix): 33.403505, -86.8082925
>> NN 2 : dist =  27 min
supermarket (Whole Foods Market): 33.3771436, -86.8007887
>> NN 3 : dist =  29 min
supermarket (Walmart Neighborhood Market): 33.428559, -86.791833

============= (33.42631593134768, -86.70288355495192) =============
>> NN 1 : dist =  0 min
supermarket (Aldi): 33.4261712, -86.7029532
>> NN 2 : dist =  5 min
supermarket (Fresh Market): 33.4215003, -86.7000679
>> NN 3 : dist =  12 min
supermarket (Winn-Dixie): 33.4173675, -86.6945961

============= (33.479436640015166, -86.84185798281771) =============
>> NN 1 : dist =  17 min
supermarket (Super Ofertas): 33.4709642, -86.8269303
>> NN 2 : dist =  19 min
supermarket (Brito's Supermarket): 33.4684959, -86.8257972
>> NN 3 : dist =  21 min
supermarket (Mi Pueblo): 33.4666125, -86.8243888

============= (33.42862167991548, -86.79182192815595) =============
>> NN 1 : dist =  0 min
supermarket (Walmart Neighborhood Market): 33.428559, -86.791833
>> NN 2 : dist =  17 min
supermarket (Western Supermarket): 33.4264824, -86.7742867
>> NN 3 : dist =  30 min
supermarket (Publix): 33.403505, -86.8082925
```

Figure 18: Three Nearest POIs and Their Walking Distances

In order to estimate the coverage of POIs for each activity, we use *k* nearest neighbor (*k*-NN) queries to find the nearest POIs of the destination of each leg of that activity. Figure 18 shows the 3-NN query results for activities "Shopping-Grocery" (left) and "Eat / Get Take-out" (right) and their walking distances. We can see that many of them are an obvious hit with walking distance < 3 minutes, but a few of them are missed with walking distance > 10 min.

We can use the hit rate to estimate the fraction of POIs covered, denoted by *p*. During location sampling for an activity, we sample from the OpenStreetMap POIs with probability *p*, and sample a random location from OpenAddresses locations with probability (1 – *p*). For activities about picking up and dropping off passenger, we expect that one end of a leg is home and the other end is another activity, and thus we can sample locations from OpenAddresses with 50% probability (as home locations) and from POIs of all types with 50% probability. Of course, the probability may also be estimated using the actual ratio observed in our survey rather than 50%.

OpenStreetMap allows us to download map data in a region with a bounding box. Besides POIs, we also use it to get the road network as required by MATSim.

D. Modeling Activity Sequence

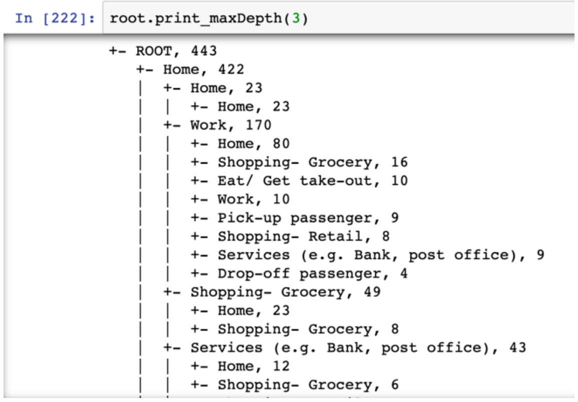
So far, we have conditioned time and location generation on activities, and travel mode can be similarly generated. There is, however, one key problem remaining: how to generate a day-plan as a sequence of activities?

Our solution is to generate the day-plans according to our survey data. The intuition is that the day-plan of people usually does not change much with their locations; for example, most people go to work at around 8 am and leave at around 5 pm, regardless of where they are. This allows us to use the entire 451 survey participants to generate day-plans to counteract the data scarcity issue.

A straightforward approach is to sample from the 451 day-plans in our survey. However, given a sequence of activities A_1, A_2, …, we need to ensure that the A_1 ends before A_2 starts, which means that we need to fix our previous model of time generation to one that is conditioned on an activity sequence rather than an individual activity.

Suppose that we already sampled activities A_1, A_2, …, A_{i-1}, and the next activity sampled is A_i (or we may decide that the sequence ends). To maximize the pool of samples that we can use to fit a PDF for the travel time of A_i (e.g., using KDE), we need to find all day-plans among our 451 participants that have A_1, A_2, …, A_{i-1} as a subsequence. While we can find the pool in a brute-force manner for each activity sequence when we need it, here we propose a more efficient method that indexes the 451 day-plans as a preprocessing step to enable much more efficient branch-and-bound search given any activity sequence as a query.

One way is to index all 451 activities using a trie (or prefix tree) structure, but in that case, (1) Home-Work-Home and (2) Home-Eat-Work-Home will diverge into two different branches, and so when we have a sequence Home-Work-Home sampled and would like to decide the time for the last activity Home and its time (e.g., 5 pm), we will only have access to Sequence (1) following the trie, while Sequence (2) is lost even though it also well captures the time off work, for example, the two sequences may be contributed by one person had breakfast at home while the other had breakfast at Chick-fil-A before heading to work. Similarly, (3) Home-Work-Eat-Work-Home will not contribute to the estimation of off-work time even though it makes a perfect sense.



```
In [222]: root.print_maxDepth(3)

          +- ROOT, 443
             +- Home, 422
             |  +- Home, 23
             |  |  +- Home, 23
             |  +- Work, 170
             |  |  +- Home, 80
             |  |  +- Shopping- Grocery, 16
             |  |  +- Eat/ Get take-out, 10
             |  |  +- Work, 10
             |  |  +- Pick-up passenger, 9
             |  |  +- Shopping- Retail, 8
             |  |  +- Services (e.g. Bank, post office), 9
             |  |  +- Drop-off passenger, 4
             |  +- Shopping- Grocery, 49
             |  |  +- Home, 23
             |  |  +- Shopping- Grocery, 8
             |  +- Services (e.g. Bank, post office), 43
             |  |  +- Home, 12
             |  |  +- Shopping- Grocery, 6
```

Figure 19: A Fragment of Activity Trie

To tackle this problem, we first mine all frequent sequential activity patterns from the 451 day-plans in our survey, where we consider an activity sequence as a frequent pattern if it is a subsequence of at least 4 persons' entire activity sequence in our survey.

Then, instead of building a trie over all day-plans, we build a trie over these frequent patterns, and for each tree-path $A_1 \rightarrow \ldots \rightarrow A_{i-1} \rightarrow A_i$ that stops at node labeled with activity A_i, we maintain the set of day-plans that contain $A_1 \rightarrow \ldots \rightarrow A_{i-1} \rightarrow A_i$ as a subsequence, denoted by $D(A_i)$. Note that to construct $D(A_i)$ we only need to filter $D(A_{i-1})$ rather than going through all the 451 day-plans.

Figure 19 shows such a trie where each activity node A_i is also marked with how many day-plans contain pattern $A_1 - \ldots - A_{i-1} - A_i$. These frequency numbers are used to estimate the probability of sampling the next activity. For example, if we already sampled Home-Work stopping at the node with frequency 170, then we sample the next event as 'Home' with probability 80/170, while we sample 'Eat/Get take-out' with probability only 10/170.

Once the next event is sampled, we can estimate its time with all the possible day-plans. For example, consider again the previous 3 day-plans (1) Home-Work-Home, (2) Home-Eat-Work-Home and (3) Home-Work-Eat-Work-Home. Given that subsequence Home-Work-Home is already sampled and corresponding tree-path reaches a node A_i marked "Home", $D(A_i)$ contains all the 3 day-plans (1), (2) and (3) which are used collectively to estimate the time off work.

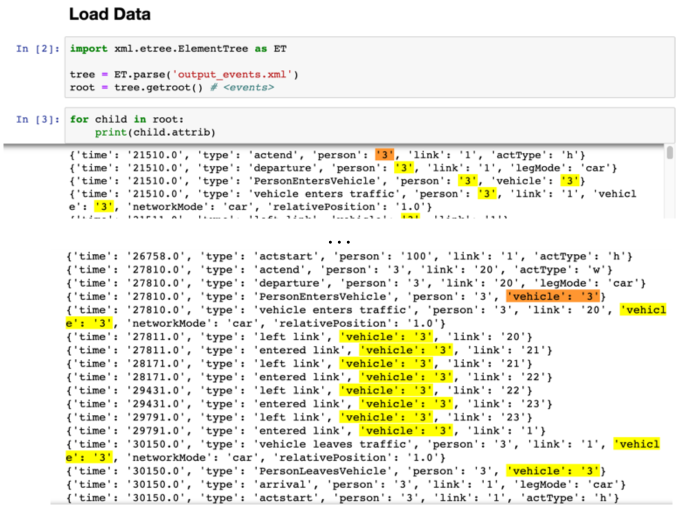
Of course, those day-plans whose time off work is before the end time of A_{i-1} should be filtered during the KDE fitting of the time PDF. If the number of day-plans for KDE fitting is less than 4, we consider the sequence as ending at A_{i-1} since the subsequent activities sequence may not generalize to the population.

III. Evaluation

The current project is still under development even though many modules have been ready. Therefore, this workshop paper only reports a preliminary test considering Home-Work traffic flow only, which should be sufficient to model the peak hour traffic. For simplicity of leg sampling, we currently model each person to have only one Home-Work leg, where time is sampled from the "Work" PDF estimated by KDE, followed by a random address location in the source ZCTA (for Home) and one in the destination ZCTA (for Work). The simulation can be visualized at https://youtu.be/ZIm0WsmKB4E. A more comprehensive modeling and evaluation effort will be our future work, where traffic flows of different types and multiple legs will be properly generated according to their proportion in the survey data following our activity trie.

MATSim outputs a sequence of events in chronological order, and Figure 20 provides an illustration. We can see in the top part that (1) Person #3 ends its stay at home (Event Type: actend) and goes to work, (2) he/she then enters Vehicle #3 on Link #1. At the bottom part, the same person finishes work and starts to drive home, where we can see events like "entered link 20", "left link 20", "entered link 21", "left link 21", "entered link 22", "left link 22", "entered link 23", "left link 23", "entered link 1" (where home locates), "vehicle leaves traffic" and "PersonLeavesVehicle".

Using a one-pass streaming algorithm over the events, we can calculate the traffic conditions on every road segment for the entire day. The information can then be compared with the ground truth to evaluate the accuracy of simulation.



Load Data

```
In [2]: import xml.etree.ElementTree as ET

        tree = ET.parse('output_events.xml')
        root = tree.getroot() # <events>
```

```
In [3]: for child in root:
            print(child.attrib)
```

```
{'time': '21510.0', 'type': 'actend', 'person': '3', 'link': '1', 'actType': 'h'}
{'time': '21510.0', 'type': 'departure', 'person': '3', 'link': '1', 'legMode': 'car'}
{'time': '21510.0', 'type': 'PersonEntersVehicle', 'person': '3', 'vehicle': '3'}
{'time': '21510.0', 'type': 'vehicle enters traffic', 'person': '3', 'link': '1', 'vehicl
e': '3', 'networkMode': 'car', 'relativePosition': '1.0'}
```

...

```
{'time': '26758.0', 'type': 'actstart', 'person': '100', 'link': '1', 'actType': 'h'}
{'time': '27810.0', 'type': 'actend', 'person': '3', 'link': '20', 'actType': 'w'}
{'time': '27810.0', 'type': 'departure', 'person': '3', 'link': '20', 'legMode': 'car'}
{'time': '27810.0', 'type': 'PersonEntersVehicle', 'person': '3', 'vehicle': '3'}
{'time': '27810.0', 'type': 'vehicle enters traffic', 'person': '3', 'link': '20', 'vehicl
e': '3', 'networkMode': 'car', 'relativePosition': '1.0'}
{'time': '27811.0', 'type': 'left link', 'vehicle': '3', 'link': '20'}
{'time': '27811.0', 'type': 'entered link', 'vehicle': '3', 'link': '21'}
{'time': '28171.0', 'type': 'left link', 'vehicle': '3', 'link': '21'}
{'time': '28171.0', 'type': 'entered link', 'vehicle': '3', 'link': '22'}
{'time': '29431.0', 'type': 'left link', 'vehicle': '3', 'link': '22'}
{'time': '29431.0', 'type': 'entered link', 'vehicle': '3', 'link': '23'}
{'time': '29791.0', 'type': 'left link', 'vehicle': '3', 'link': '23'}
{'time': '29791.0', 'type': 'entered link', 'vehicle': '3', 'link': '1'}
{'time': '30150.0', 'type': 'vehicle leaves traffic', 'person': '3', 'link': '1', 'vehicl
e': '3', 'networkMode': 'car', 'relativePosition': '1.0'}
{'time': '30150.0', 'type': 'PersonLeavesVehicle', 'person': '3', 'vehicle': '3'}
{'time': '30150.0', 'type': 'arrival', 'person': '3', 'link': '1', 'legMode': 'car'}
{'time': '30150.0', 'type': 'actstart', 'person': '3', 'link': '1', 'actType': 'h'}
```

Figure 20: Part of the Event File Output by MATSim

Table II: Average Link Speed Between 10 am and 11 am

MATSim ID	Interstate	AVG Monitored Speed	AVG Simulation Speed
50779678_0	I-65 N	59	58.17957611
50779677_0	I-65 N	59	74.02948169
50779689_0	I-65 N	59	72.39813154
50779690_1	I-65 N	64	73.73942556
50779690_2	I-65 N	64	52.64153811
50779701_0	I-65 N	56	63.17555545
50779764_0	I-65 N	57	55.39014001
50779763_0	I-65 N	57	72.006793
50779763_1	I-65 N	59	73.10841707
50779839_1	I-65 N	62	72.87918497
50779839_2	I-65 N	60	71.91137377
176219152_0	I-65 N	60	49.9337033
176219153_0	I-65 N	60	71.9594894
176219153_1	I-65 N	60	69.63908062
176217152_0	I-65 N	60	74.31905239
176217151_1	I-65 N	59	73.62659128
83144665_0	I-65 N	59	66.78876233
83144673_1	I-65 N	61	72.13814069
…	…	…	…
191520276_1	I-20 W/ I-59 S	52	72.41188686
83142927_0	I-20 W/ I-59 S	56	65.31363755
119869823_0	I-20 W/ I-59 S	62	72.82643204
119869818_0	I-20 W/ I-59 S	62	73.91760037
119869818_1	I-20 W/ I-59 S	65	74.44139168
119869818_2	I-20 W/ I-59 S	66	73.68865908
50780387_0	I-20 W/ I-59 S	66	61.1290964
119869818_2	I-20 W/ I-59 S	66	73.68865908
50780385_0	I-20 W/ I-59 S	66	58.74780766
50780374_1	I-20 W/ I-59 S	67	67.23911964
50780369_0	I-20 W/ I-59 S	67	71.16430077
50780365_0	I-20 W/ I-59 S	67	71.23445757
50780365_1	I-20 W/ I-59 S	64	73.74722821
119869822_0	I-20 W/ I-59 S	63	68.11513108
119869822_0	I-20 W/ I-59 S	63	68.11513108
50780962_0	I-20 W/ I-59 S	63	59.42168602
50780963_1	I-20 W/ I-59 S	64	70.78876887
50781075_0	I-20 W/ I-59 S	64	59.92531749

We use a sample of the real road monitoring data from the National Performance Management Research Data Set (NPMRDS) obtained from Federal Highway Administration (FHWA) with the help of the Regional Planning Commission of Greater Birmingham (RPCGB).

Table II shows the ground-truth average speed (miles per hour) of important road segments between 10 am and 11 am, and the average speed from our simulation for the same time period. We can see that the speed values of simulation match approximately to that of the monitored readings, though ours tend to be higher as we do not consider other trips such as those for school or for shopping. We also notice that the trips are not sufficiently clustered around the morning peak hours, and may need to improve our methods to fit the KDE model of activity "Work", such as to isolate out regular working trips from special ones such as those of Uber drivers, and to boost the proportion of the former to make the peak hour speed match better to the monitored speed readings. As this workshop paper just aims to be a proof of concept and feasibility of our approach, we will leave the more accurate modeling of the Birmingham traffic model as our future work.

III. Conclusion

While the current project is still under development, we have demonstrated a reasonable match of our simulation with the real traffic data on various road segments. This shows that our data-driven approach using open data to address the small data problem of our user survey is effective.

Acknowledgment

This project is partially supported by the STRIDE center and NSF OAC-1755464 and NSF DGE-1723250.

References

[1] MATSim: https://matsim.org/

[2] Horni, Andreas, Kai Nagel, and Kay W. Axhausen, eds. The multi-agent transport simulation MATSim. London: Ubiquity Press, 2016.

[3] MATSim Scenario Gallery: https://www.matsim.org/gallery/

[4] OpenStreetMap: https://www.openstreetmap.org

[5] Abdoul-Ahad Choupani and Amir Reza Mamdoohi. "Population synthesis using iterative proportional fitting (IPF): A review and future research." Transportation Research Procedia 17 (2016): 223-233.

[6] US Census Bureau: https://www.census.gov/

[7] Oliver Kennedy, D. Richard Hipp, Stratos Idreos, Amélie Marian, Arnab Nandi, Carmela Troncoso, and Eugene Wu. "Small data," In 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 1475-1476. IEEE, 2017.

[8] TIGERweb: https://tigerweb.geo.census.gov

[9] American Community Survey: https://www.census.gov/programs-surveys/acs

[10] Major Employers List by BBA: https://www.birminghambusinessalliance.com/major-employers

[11] OpenAddresses: https://openaddresses.io/

[12] OpenStreetMap: https://www.openstreetmap.org

[13] OpenStreetMap POI Types: https://wiki.openstreetmap.org/wiki/Map_Features