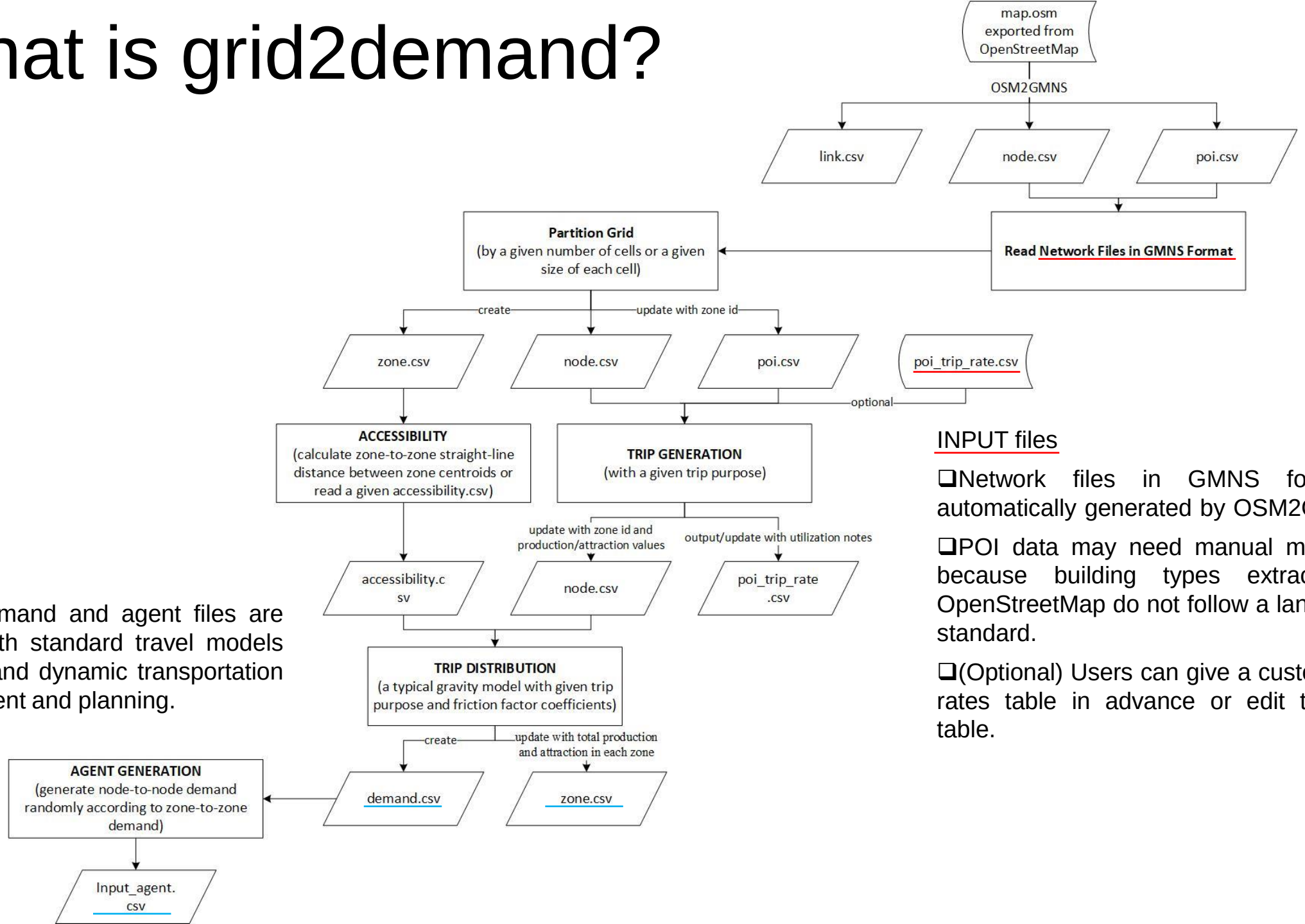
Introduction to GRID2DEMAND

What is grid2demand?



Output demand and agent files are aligned with standard travel models for static and dynamic transportation management and planning.

INPUT files

- ❑Network files in GMNS format can automatically generated by OSM2GMNS tool.
- ❑POI data may need manual modification, because building types extracted from OpenStreetMap do not follow a land use type standard.
- ❑(Optional) Users can give a customized trip rates table in advance or edit the default table.

Why proposing grid2demand?

Challenges for transportation planning:

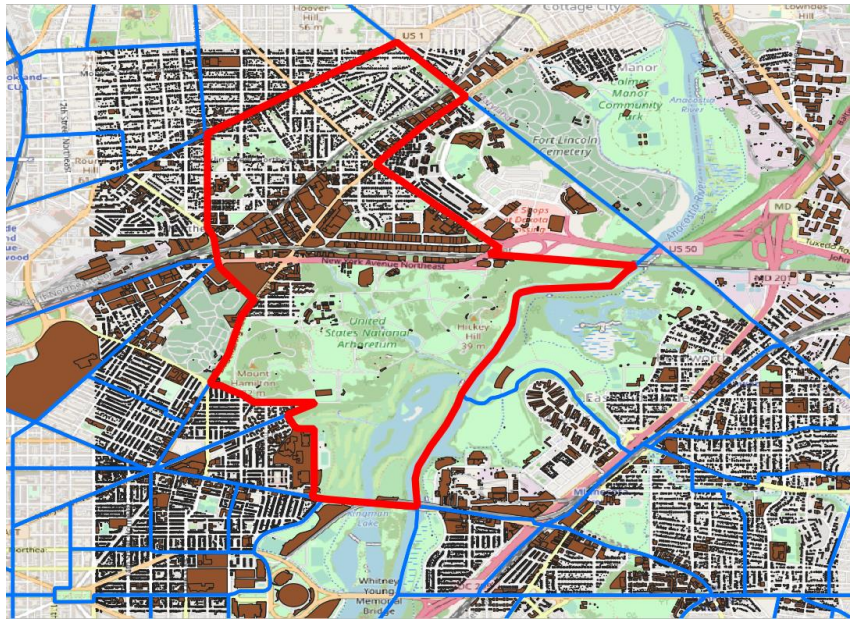
- ❑ Where does the demand come from?
  - Four-step method: trip generation (production/attraction), trip distribution (origin/destination), modal split, traffic assignment
  - Internal traffic and external traffic

- ❑ How to get the essential data to forecast travel demand?
  - Census data (demographic and socioeconomic statistics)
  - Travel survey
  - Transit data
  - Point of Interest (POI)
  - Trip rate
  - Trip purpose

- ❑ How to partition traffic analysis zones?
  - Irregular shape like census tract or administration boundary
  - Regular polygon like grid, hexagon

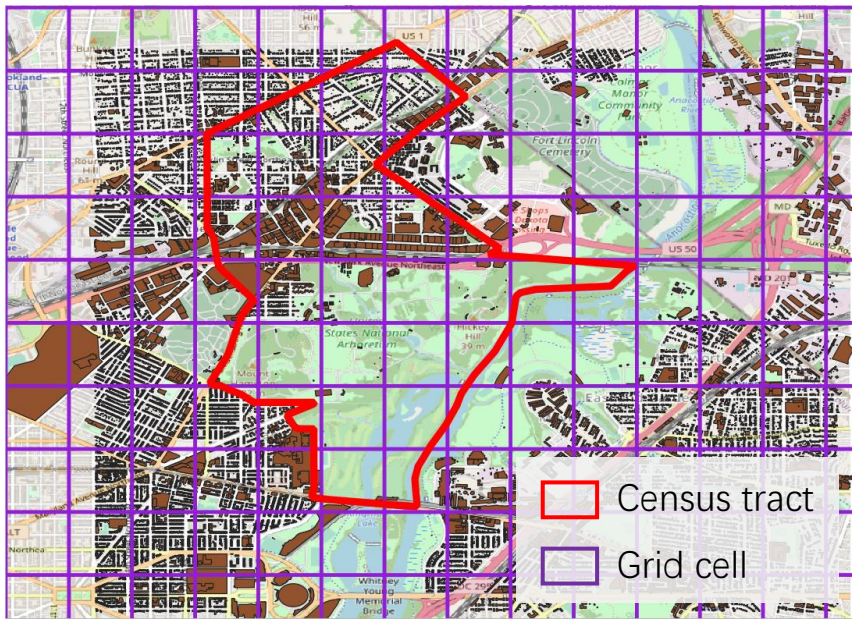
Zones partitioned by grids could reduce spatial biases and are suitable for any size of the area.

TAZ based on census tracts



Land use is ignored in TAZs. The spatial biases may deteriorate the accuracy of travel demand.

Grid zones



Trip generation directly comes from activity nodes, which is aggregated in grid zones. Smaller grid, more accurate demand.

Where can grid2demand be used?

	Name	Function description
1	grid2demand	Forecast travel demand or freight demand in a region or subarea around the world
2	map2grid	Partition grid zones for any map in GMNS format
3	node2grid	Partition network nodes by grid zones
4	poi2grid	Partition poi nodes by grid zones
5	grid2stat	Produce grid-by-grid statistics for # of nodes in different categories
6	grid2production	Obtain total production of each grid
7	grid2attraction	Obtain total attraction of each grid
8	rate2generation	Calculate trip production and attraction values for POI nodes
9	boundary2generation	Define production and attraction values for boundary nodes
10	residential2generation	Define production and attraction values for residential nodes
11	purpose2production	Calculate production values under a specific trip purpose
12	purpose2attraction	Calculate attraction values under a specific trip purpose
13	PA2ODmatrix	Obtain OD matrix according to production and attraction of each zone
14	matrix2agent	Obtain node-to-node agent according to zone-to-zone OD matrix
15	ReadLocalFile	Read network files in GMNS format
16	RateChecking	Obtain trip rate table for all building types
17	TripPurposeChecking	Obtain trip purpose according to production and attraction rates
18	AccessibilityChecking	Obtain zone-to-zone accessibility
19	GridVisual	Visualize grid zones
20	DemandVisual	Visualize zone-to-zone demand according to volume size

Major characteristics and significance

- Take advantage of open-source data and open to the public
- Integrate land use and transportation
- Avoid inconsistence of multi-source basic data
- Help students learn the four-step method and have the idea of transportation planning
- Help scholars obtain the demand data for research
- Help engineers develop local transportation model with consistent data format
- Contribute to the community of multi-discipline like urban planning, geography, and transportation engineering

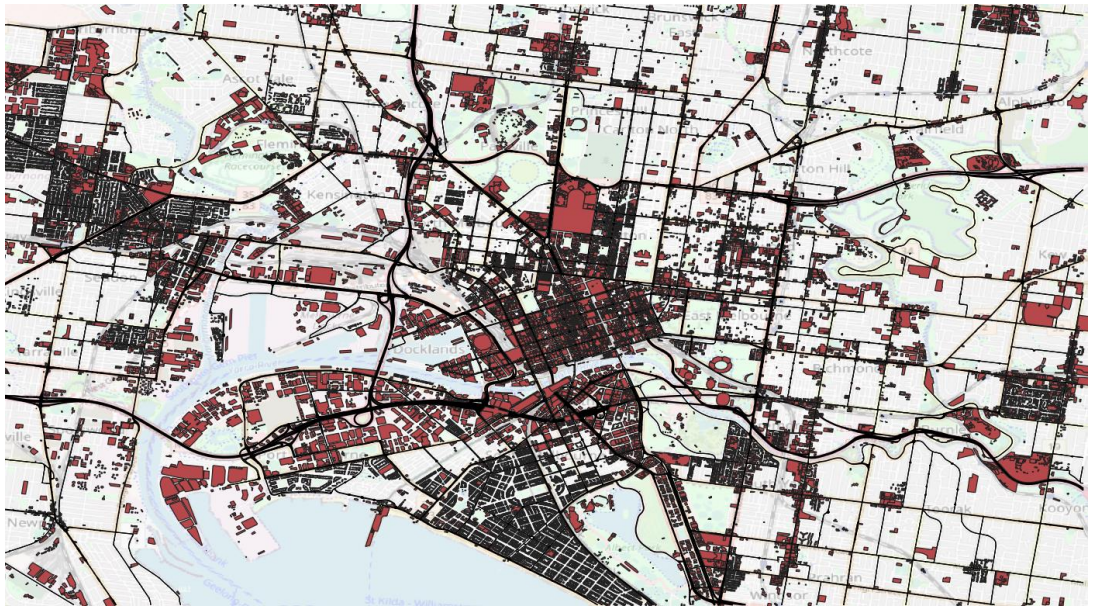
How to use grid2demand?

Step 1: Network preparation

- Export a network file from OpenStreetMap.
- Use OSM2GMNS tool to convert the network from OpenStreetMap to a routable transportation network with *.csv* files in standard GMNS format. (https://github.com/asu-trans-ai-lab/OSM2GMNS)

```
import osm2gmns as og
net = og.getNetFromOSMFile('map.osm', network_type=('auto'), POIs=True, default_lanes=True,default_speed=True)
og.connectPOIWithNet(net)
og.generateNodeActivityInfo(net)
og.outputNetToCSV(net)
```

Road network



Road network with Point Of Interest (POI)

Step 2: Import GRID2DEMAND and read the network data

```
import grid2demand as gd
gd.ReadNetworkFiles()
```

Step 3: Partition Grid into cells

- Grids can be automatically generated by a given number of grid cells or cell's width and height in meters.

```
gd.PartitionGrid(number_of_x_blocks=None, number_of_y_blocks=None, cell_width=500, cell_height=500)
```

l	total_poi_count	residential_poi_co	office_poi_count	shopping_poi_cour	school_poi_count	parking_poi_count	boundary_node_co	tot
-11	262	181	0	21	0	3	0	
-11	51	8	1	4	2	12	0	
-11	65	0	2	16	1	3	0	
-11	32	2	0	9	0	4	0	
-11	39	0	0	4	0	9	0	
-11	18	1	0	3	0	0	0	
-11	187	0	0	12	0	0	0	
-11	154	1	0	18	0	0	0	
-11	47	2	0	3	0	0	0	
-11	94	0	0	1	1	4	0	
-11	5	0	0	0	0	0	0	
-11	5	0	0	3	0	0	0	
-11	9	0	2	1	0	0	0	
-11	19	1	0	12	0	0	0	
-11	69	51	0	12	0	0	0	
-11	45	44	1	0	0	0	0	
-11	41	23	0	1	1	0	0	
-11	89	19	1	0	0	8	0	
-11	71	4	2	1	0	12	0	

Step 4: Get production/attraction rates of each land use type with a specific trip purpose

- Users can customize trip purpose and *poi_trip_rate.csv*. Default trip rates are collected according to ITE trip generation rates tables.
- The values of production and attraction of each POI node are calculated according to land use type, trip purpose and the area of the building (or other unit of measurement).

```
gd.GetPoiTripRate(trip_purpose=1)
```

land_use_type	building	ITE_land_descrip	unit_of_measure	production_rate1	attraction_rate1
Religious	cathedral	Church	1,000 Sq. Ft. GFA	0.49	0.1
	chapel	Church	1,000 Sq. Ft. GFA	0.49	0.1
	church	Church	1,000 Sq. Ft. GFA	0.49	0.1
	monastery	Mosque	1,000 Sq. Ft. GFA	4.22	0.1
	mosque	Mosque	1,000 Sq. Ft. GFA	4.22	0.1
	religious	Church	1,000 Sq. Ft. GFA	0.49	0.1
	shrine	Mosque	1,000 Sq. Ft. GFA	4.22	0.1
	synagogue	Church	1,000 Sq. Ft. GFA	0.49	0.1
	temple	Mosque	1,000 Sq. Ft. GFA	4.22	0.1

A sample of *poi_trip_rate.csv*

Step 5: Define production/attraction values of each node according to the node's activity type

- Three kinds of nodes in the network can generate travel demand: 1) POI nodes, 2) residential nodes, and 3) boundary nodes.
- Users can customize production and attraction values of residential nodes and boundary nodes.

```
gd.GetNodeDemand(residential_production=20, residential_attraction=5, boundary_production=40, boundary_attraction=10)
```

A sample of updated *node.csv*

name	node_id	osm_node_id	osm_highw	zone_id	ctrl_type	node_type	activity_typ	is_boundar	x_coord	y_coord	main_node	poi_id	notes	production	attraction	activity_location_tab
	17717			33	0		residential	1	144.8657	-37.84452			boundary n	1000	1000	boundary
	17718			2	0		residential	0	144.891	-37.7659			boundary n	20	20	residential
	17719			36	0		primary	1	144.8657	-37.77062			boundary n	1000	1000	boundary
	17720			55	0		residential	1	144.8853	-37.856			boundary n	1000	1000	boundary
	17737			52	0		trunk	1	144.983	-37.856			boundary n	1000	1000	boundary
	17738			30	0		residential	0	144.9915	-37.856			boundary n	20	20	residential
	17739			51	0		residential	1	144.9882	-37.856			boundary n	1000	1000	boundary
	17756			13	0		poi	0	144.9715	-37.80472		0		119.2185	119.2185	poi
	17757			13	0		poi	0	144.9717	-37.80332		1		220.2184	220.2184	poi
	17758			4	0		poi	0	144.9489	-37.77222		2		62.03979	62.03979	poi
	17759			7	0		poi	0	145.0176	-37.77758		3		2.376994	56.81016	poi
	17760			4	0		poi	0	144.9426	-37.77048		4		538.5512	26.39957	poi
	17761			4	0		poi	0	144.9485	-37.77208		5		4.996177	119.4086	poi
	17762			5	0		poi	0	144.964	-37.77571		6		21.20318	506.756	poi

Step 6: Calculate zone-to-zone accessibility

- Different latitudes represent different lengths on a flat surface. The average latitude of the area of interest will be used for converting x/y coordinates into lengths on a flat surface.

Latitude	City	Degree-equivalent distance (km)	(miles)
60°	Saint Petersburg	55.80	34.67
51°	Greenwich	69.47	43.17
45°	Bordeaux	78.85	49.00
30°	New Orleans	96.49	59.96
0°	Quito	111.3	69.16

- Currently, accessibility is measured by straight-line distance between zone centroids.

```
gd.ProduceAccessMatrix()
```

o_zone_id	o_zone_name	d_zone_id	d_zone_name	accessibility	geometry
1	A1	1	A1	0	LINESTRING (144.8738771 -37.7812111,144.8738771 -37.7812111)
1	A1	2	A2	1.240470762	LINESTRING (144.8738771 -37.7812111,144.8894837 -37.7792282)
1	A1	3	A3	3.738371745	LINESTRING (144.8738771 -37.7812111,144.921227 -37.7788018)
1	A1	4	A4	5.770048412	LINESTRING (144.8738771 -37.7812111,144.9469159 -37.7767061)
1	A1	5	A5	7.515572835	LINESTRING (144.8738771 -37.7812111,144.9690411 -37.7758512)
1	A1	6	A6	9.708368997	LINESTRING (144.8738771 -37.7812111,144.9968093 -37.7743317)
1	A1	7	A7	11.68238863	LINESTRING (144.8738771 -37.7812111,145.0218923 -37.7746688)
1	A1	8	A8	12.79241745	LINESTRING (144.8738771 -37.7812111,145.0360656 -37.7772268)
1	A1	9	B1	0.948804466	LINESTRING (144.8738771 -37.7812111,144.8752054 -37.7931705)
1	A1	10	B2	1.986657085	LINESTRING (144.8738771 -37.7812111,144.8932448 -37.7973263)
1	A1	11	B3	3.773581487	LINESTRING (144.8738771 -37.7812111,144.919949 -37.7941626)

A sample of *accessibility.csv*

Step 7: Apply gravity model to perform trip distribution

- For each OD pair, a typical gravity model is applied to calculate zone-to-zone demand volume.

$$T_{ij} = P_i \cdot \frac{A_j \cdot F_{ij} \cdot K_{ij}}{\sum_j (A_j \cdot F_{ij} \cdot K_{ij})}$$

$$F_{ij} = a(d_{ij})^b e^{c(d_{ij})}$$

where,

T_{ij}: Trips from zone i to zone j ; P_i: Productions in zone i ; A_j: Attractions in zone j

F_{ij}: Friction factor for travel from zone i to zone j ;

K_{ij}: Correction factor for travel from zone i to zone j . Default value equals to 1;

d_{ij}: Accessibility from zone i to zone j;

a, b, c: Friction factor coefficients.

Trip Purpose	a	b	c
1 (HBW)	28507	-0.02	-0.123
2 (HBO)	139173	-1.285	-0.094
3 (NHB)	219113	-1.332	-0.1

Default values of friction factor coefficients under a specific trip purposes

Step 7: Apply gravity model to perform trip distribution

- Users can specify a trip purpose to use default coefficients or customize friction factor coefficients.

```
gd.RunGravityModel(trip_purpose=1, a=None, b=None, c=None)
```

A sample of output *demand.csv*

o_zone_id	o_zone_name	d_zone_id	d_zone_name	accessibility	volume	geometry
1	A1	1	A1	0	0	LINESTRING (132.7088033 34.4062304,132.7088033 34.4062304)
1	A1	2	A2	0.52597618	9	LINESTRING (132.7088033 34.4062304,132.714254 34.4061674)
1	A1	3	A3	0.793795729	1	LINESTRING (132.7088033 34.4062304,132.7170292 34.4063404)
1	A1	4	B1	0.347838257	16	LINESTRING (132.7088033 34.4062304,132.7102556 34.402931)
1	A1	5	B2	0.654836033	28	LINESTRING (132.7088033 34.4062304,132.7148118 34.4030752)
1	A1	6	B3	0.950270834	3	LINESTRING (132.7088033 34.4062304,132.7175319 34.4016695)
1	A1	7	C1	0.716570732	12	LINESTRING (132.7088033 34.4062304,132.7107105 34.3990531)
1	A1	8	C2	0.935279901	18	LINESTRING (132.7088033 34.4062304,132.7148674 34.3986685)
1	A1	9	C3	1.280400947	1	LINESTRING (132.7088033 34.4062304,132.7179452 34.3966121)
1	A1	10	D1	1.399361966	1	LINESTRING (132.7088033 34.4062304,132.7073985 34.3917959)
1	A1	11	D2	1.385940153	3	LINESTRING (132.7088033 34.4062304,132.7135851 34.3926861)
1	A1	12	D3	1.58078435	4	LINESTRING (132.7088033 34.4062304,132.7184847 34.3930142)
1	A1	13	Gate1	1.399097621	1	LINESTRING (132.7088033 34.4062304,132.70383 34.39261)
1	A1	14	Gate2	0.945272919	5	LINESTRING (132.7088033 34.4062304,132.70383 34.39779)
1	A1	15	Gate3	0.573797957	13	LINESTRING (132.7088033 34.4062304,132.70383 34.40297)
1	A1	16	Gate4	0.514377066	1	LINESTRING (132.7088033 34.4062304,132.70383 34.40815)
1	A1	17	Gate5	0.685333827	0	LINESTRING (132.7088033 34.4062304,132.70901 34.41333)
1	A1	18	Gate6	0.859907828	7	LINESTRING (132.7088033 34.4062304,132.71419 34.41333)
1	A1	19	Gate7	1.228347246	0	LINESTRING (132.7088033 34.4062304,132.71937 34.41333)

Step 8: Generate Agent

- Based on zone-to-zone demand volume, agent-based node-to-node demand will be generated randomly.

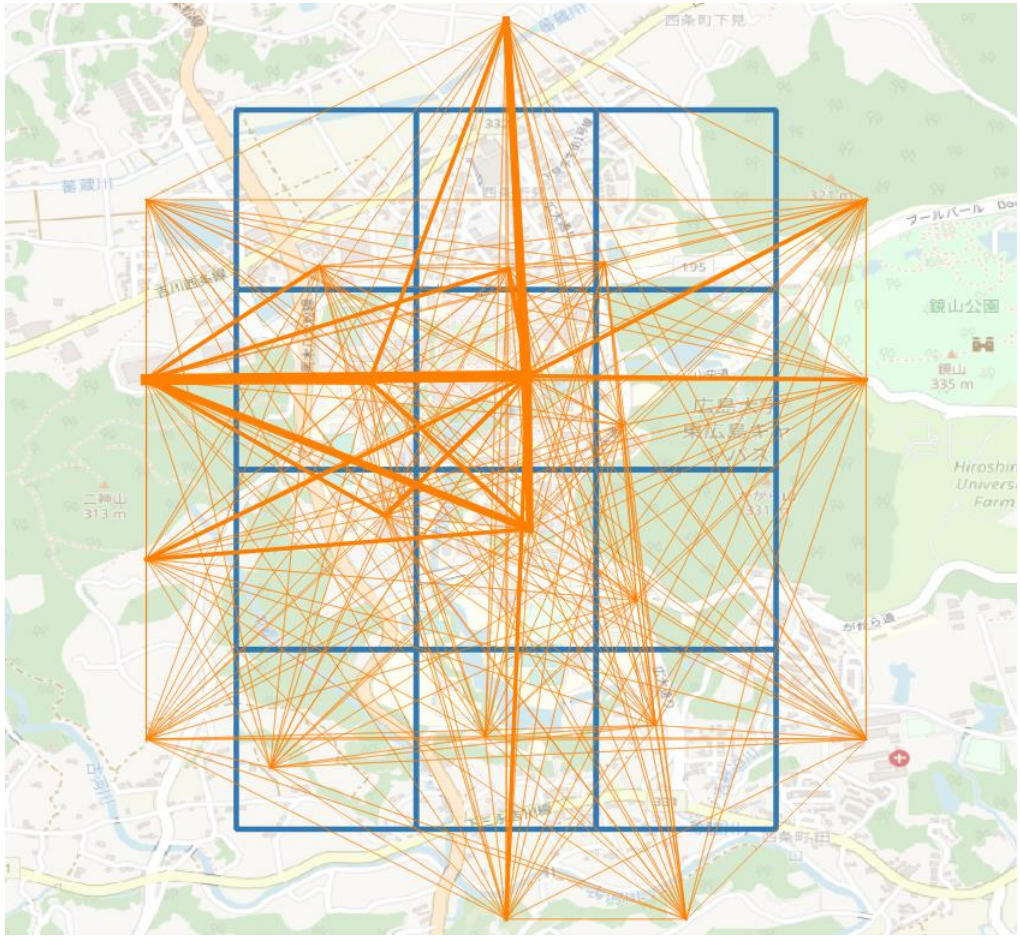
```
gd.GenerateAgentBasedDemand()
```

A sample of output *input_agent.csv*

agent_id	agent_type	o_node_id	d_node_id	o_osm_node_id	d_osm_node_id	o_zone_id	d_zone_id	geometry	departure_time
1	v	31034	19449			1	2	LINESTRING(144.874	740
2	v	32612	22418			1	2	LINESTRING(144.881	745
3	v	33407	22417			1	2	LINESTRING(144.875	729
4	v	33098	33013			1	2	LINESTRING(144.880	730
5	v	31571	4885		257521042	1	2	LINESTRING(144.867	745
6	v	33083	33585			1	2	LINESTRING(144.881	714
7	v	33441	10810		690259903	1	2	LINESTRING(144.871	729
8	v	33089	33317			1	2	LINESTRING(144.880	742
9	v	6023	32639	289580464		1	2	LINESTRING(144.879	711
10	v	32138	33598			1	2	LINESTRING(144.874	723
11	v	31693	33010			1	2	LINESTRING(144.866	742
12	v	33273	32587			1	2	LINESTRING(144.875	733
13	v	33602	13249		2069193685	1	2	LINESTRING(144.881	730

Visualization

- All features are stored with geometry (i.e. x/y coordinates). It's readily available to visualize the output *demand.csv* and other features in QGIS.



https://colab.research.google.com/github/asu-trans-ai-lab/osm_test_data_set/blob/main/grid2demand.ipynb