




# BUS LINE OPTIMIZATION

성유지 조규선 황연재 송예은



# Bus Stop Selection

출근시간대인 7-9시, 퇴근시간대인 18-20시 해당하는 카드태깅 데이터만 추출, 승차/하차구분

```
In [33]: df_79_on.head()
```

```
Out[33]:
```

	사용자구분	최초승차일시	이비카드정류장ID	승차시간대	정류장이름	표준정류장ID	관할관청	위도	경도
0	1	2018-07-01 07:57:31	<a href="#">4197806.0</a>	07	서동탄역신일해피트리(중)	233001287	경기도 화성시	37.1994	127.053317
1	1	2018-07-01 08:41:45	<a href="#">4197806.0</a>	08	서동탄역신일해피트리(중)	233001287	경기도 화성시	37.1994	127.053317
2	1	2018-07-01 07:57:50	<a href="#">4197806.0</a>	07	서동탄역신일해피트리(중)	233001287	경기도 화성시	37.1994	127.053317
3	1	2018-07-01 09:41:01	<a href="#">4197806.0</a>	09	서동탄역신일해피트리(중)	233001287	경기도 화성시	37.1994	127.053317
4	1	2018-07-01 08:51:32	<a href="#">4197806.0</a>	08	서동탄역신일해피트리(중)	233001287	경기도 화성시	37.1994	127.053317

```
In [34]: df_1820_off.head()
```

```
Out[34]:
```

	사용자구분	최종하차일시	이비카드정류장ID	하차시간대	정류장이름	표준정류장ID	관할관청	위도	경도
0	1	2018-07-01 18:37:11	<a href="#">4199616.0</a>	18	대방노블랜드.풍림아이원	233001852	경기도 화성시	37.130317	126.927083
1	1	2018-07-01 19:16:52	<a href="#">4199616.0</a>	19	대방노블랜드.풍림아이원	233001852	경기도 화성시	37.130317	126.927083
2	1	2018-07-01 18:36:49	<a href="#">4199616.0</a>	18	대방노블랜드.풍림아이원	233001852	경기도 화성시	37.130317	126.927083
3	1	2018-07-01 18:29:28	<a href="#">4199616.0</a>	18	대방노블랜드.풍림아이원	233001852	경기도 화성시	37.130317	126.927083
4	1	2018-07-01 19:31:50	<a href="#">4199616.0</a>	19	대방노블랜드.풍림아이원	233001852	경기도 화성시	37.130317	126.927083

# Bus Stop Selection

출퇴근시간대 승차/하차가 빈번한 상위 50개 정류장 추출, 동일한 이름을 가진(상행/하행) 정류장 중 택 1

In [40]: point\_79\_on.sort\_values(['횡수'], ascending=False).head()

Out [40]:

	이비카드정류장ID	정류장이름	위도	경도	횡수
18322	<a href="#">4108037.0</a>	신영통현대타운.두산위브	37.235333	127.062100	1564.0
7320	<a href="#">4108035.0</a>	반월리큰고개	37.232550	127.064783	996.0
5202	<a href="#">4100048.0</a>	신창미션힐.송화초교	37.203300	127.038550	857.0
10074	<a href="#">4195716.0</a>	다은마을(중)	37.200417	127.067300	813.0
14717	<a href="#">4116673.0</a>	와우2리	37.215900	126.976183	792.0

In [52]: point\_1820\_off.sort\_values(['횡수'], ascending=False).head()

Out [52]:

	이비카드정류장ID	정류장이름	위도	경도	횡수
4234	4108036.0	신영통현대타운.두산위브	37.235467	127.062467	1276.0
7670	4170973.0	메타폴리스(중)	37.203700	127.087500	795.0
8537	4170243.0	동탄1동행정복지센터	37.208650	127.072567	772.0
1256	4108034.0	반월리큰고개	37.233000	127.064800	763.0
17625	4199835.0	농동마을입구	37.216850	127.058150	671.0

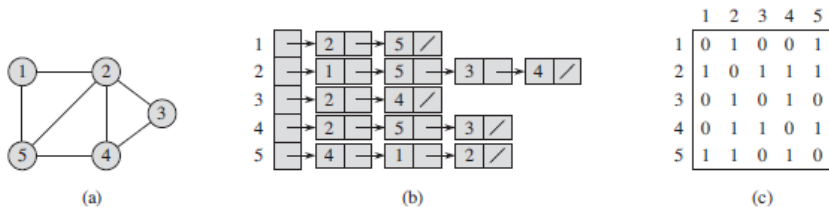
# Graph Algorithm Representation Selection

- Adjacency – List

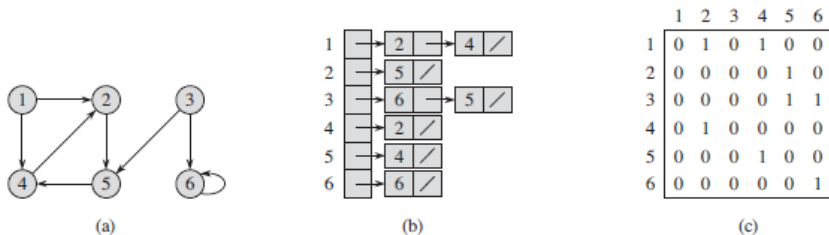
(인접리스트)

- Adjacency – Matrix

(인접행렬)



**Figure 22.1** Two representations of an undirected graph. (a) An undirected graph  $G$  with 5 vertices and 7 edges. (b) An adjacency-list representation of  $G$ . (c) The adjacency-matrix representation of  $G$ .



**Figure 22.2** Two representations of a directed graph. (a) A directed graph  $G$  with 6 vertices and 8 edges. (b) An adjacency-list representation of  $G$ . (c) The adjacency-matrix representation of  $G$ .

# Graph Weight Representation

In [53]: `morning_dist`

A matrix: 86 × 86 of type dbl

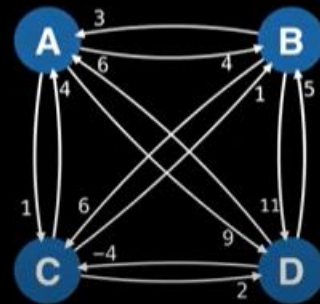
	4100048	4100049	4100050	4100051	4102921	4105628	4107846	4108007	4108028	4108031	...	4197662	419907
4100048	0.00000	36.94688	3022.7647	712.7978	6780.917	3840.465	5134.000	968.8379	2740.5182	3312.1971	...	3001.5132	7797.799
4100049	36.94688	0.00000	3059.3974	694.6048	6744.645	3872.264	5170.831	966.1577	2765.9056	3337.4706	...	3027.0013	7761.617
4100050	3022.76472	3059.39744	0.0000	3368.7000	9695.515	1597.019	2268.313	3137.4434	1958.7151	2160.4892	...	2016.6706	10699.030
4100051	712.79781	694.60476	3368.7000	0.0000	6328.792	3906.669	5576.914	447.9046	2633.7479	3186.8015	...	2889.5514	7335.556
4102921	6780.91727	6744.64508	9695.5149	6328.7922	0.000	9981.280	11896.845	6576.9747	8560.9065	8986.0766	...	8764.5686	1020.927
4105628	3840.46507	3872.26363	1597.0189	3906.6694	9981.280	0.000	2988.104	3533.3020	1427.6592	1142.1028	...	1258.7154	10944.534
4107846	5134.00000	5170.83060	2268.3133	5576.9143	11896.845	2988.104	0.000	5384.4387	4040.7808	4027.3864	...	4006.8770	12908.112
4108007	968.83790	966.15771	3137.4434	447.9046	6576.975	3533.302	5384.439	0.0000	2218.4839	2762.1976	...	2470.3568	7573.062
4108028	2740.51824	2765.90557	1958.7151	2633.7479	8560.906	1427.659	4040.781	2218.4839	0.0000	571.7709	...	261.2726	9519.949
4108031	3312.19707	3337.47058	2160.4892	3186.8015	8986.077	1142.103	4027.386	2762.1976	571.7709	0.0000	...	311.6077	9929.452
4108035	4001.28387	4022.86546	2938.4909	3772.0024	9196.299	1619.547	4607.623	3328.3291	1334.9217	834.0838	...	1106.3001	10103.090
4108036	4161.36415	4180.30468	3303.8728	3871.3334	9072.389	2002.655	4990.439	3423.6884	1599.8732	1163.8367	...	1400.1662	9960.882
4108037	4132.07578	4150.86071	3298.3630	3839.1137	9037.193	2010.120	4998.205	3391.4080	1580.4701	1153.3853	...	1384.3240	9926.063
4116673	5704.43777	5668.63544	8585.0645	5224.2743	1135.051	8847.068	10799.821	5457.9293	7426.1332	7851.3681	...	7629.5192	2115.132
4116674	5695.26341	5659.48029	8574.4096	5213.9955	1147.810	8834.678	10789.718	5446.8916	7413.6005	7838.4356	...	7616.8086	2126.203
4116677	6137.16524	6102.54511	8921.3451	5599.6500	1242.093	9055.127	11164.795	5784.3833	7627.5038	8013.9748	...	7813.9864	1966.348
4116678	6112.98061	6078.37537	8896.3535	5574.9824	1259.245	9029.985	11139.921	5759.4163	7602.3632	7988.9471	...	7788.8864	1989.897
4116680	6320.77954	6286.53766	9074.2031	5769.1520	1343.181	9170.053	11323.886	5939.9823	7742.6795	8116.7024	...	7923.7208	1945.297
4116682	6453.47812	6419.66439	9173.2417	5887.7466	1506.564	9231.291	11428.378	6044.2608	7805.1675	8166.8345	...	7980.7767	2005.996
4116691	2125.38059	2088.62412	5114.6095	1794.4717	4668.104	5690.886	7259.048	2160.7250	4369.8819	4898.1982	...	4616.2932	5687.788
4116692	1909.72166	1873.10562	4891.7962	1569.7409	4875.703	5468.083	7041.768	1939.7819	4152.0464	4683.3008	...	4399.5986	5894.399
4116693	1464.01097	1427.55085	4444.9034	1144.5730	5317.771	5051.125	6594.293	1540.9131	3759.3636	4301.1536	...	4011.0587	6335.447

# Travelling Salesman Problem

- 모든 도시들을 단 한 번만 방문 후, 원래 시작점으로 돌아오는 최소 비용의 이동 순서를 구하는 알고리즘

In other words, the problem is: given a **complete graph** with weighted edges (as an adjacency matrix) what is the **Hamiltonian cycle** (path that visits every node once) of minimum cost?

	A	B	C	D
A	0	4	1	9
B	3	0	6	11
C	4	1	0	2
D	6	5	-4	0



# 화성시 다람쥐버스

- 일정 구간을 똑같이 반복하여 도는 버스



350x350

ppt 제목

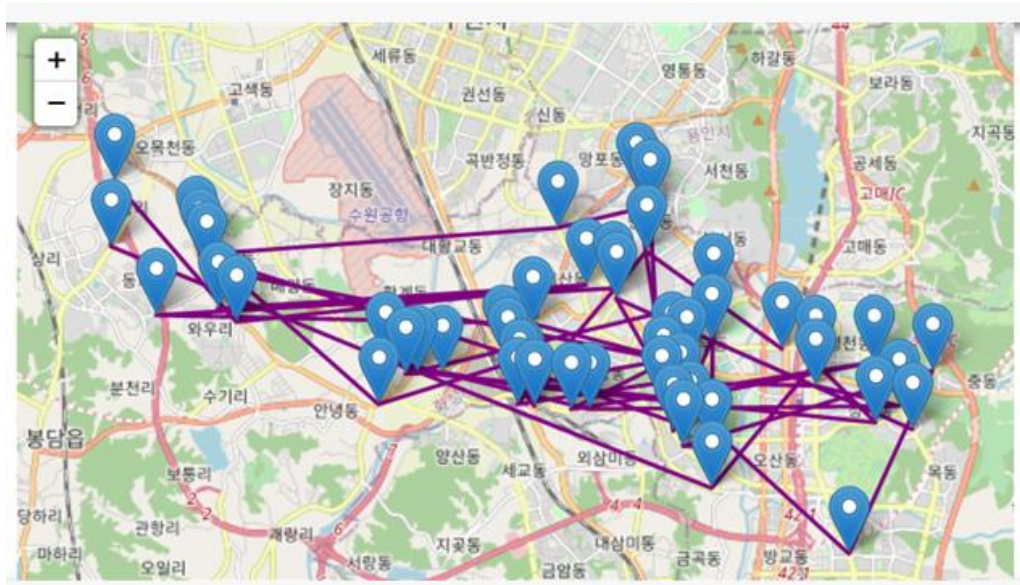
출퇴근 시간의 히어로, 다람쥐버스



7 / n

# 화성시 다람쥐버스 노선 제안 1/7

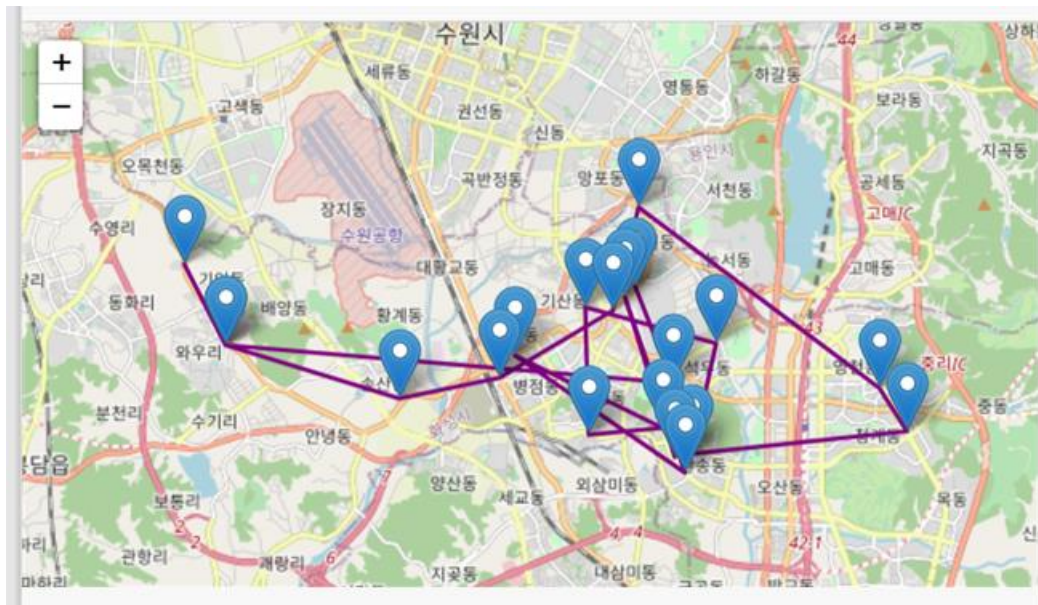
- 출근시간에 가장 승차인원이 많은 정류장 50개 사용
- TSP 알고리즘으로 노선 구현
- Folium으로 시각화



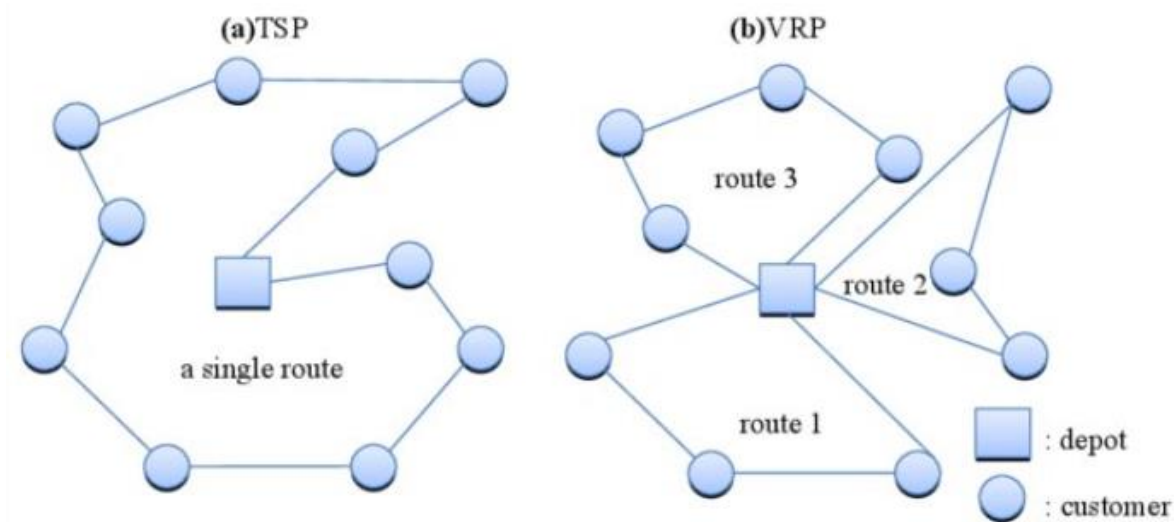


# 화성시 다람쥐버스 노선 제안 2/7

- 출근시간에 가장 승차인원/하차인원이 많은 정류장 각각 10개 사용
- TSP 알고리즘으로 노선 구현
- Folium으로 시각화

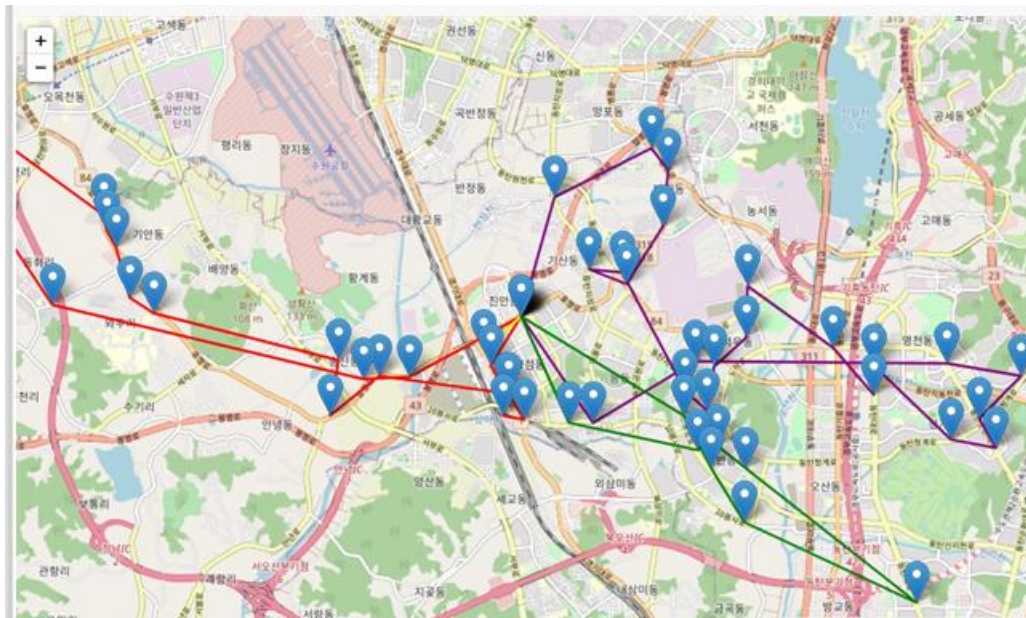


# Vehicle Route Problem



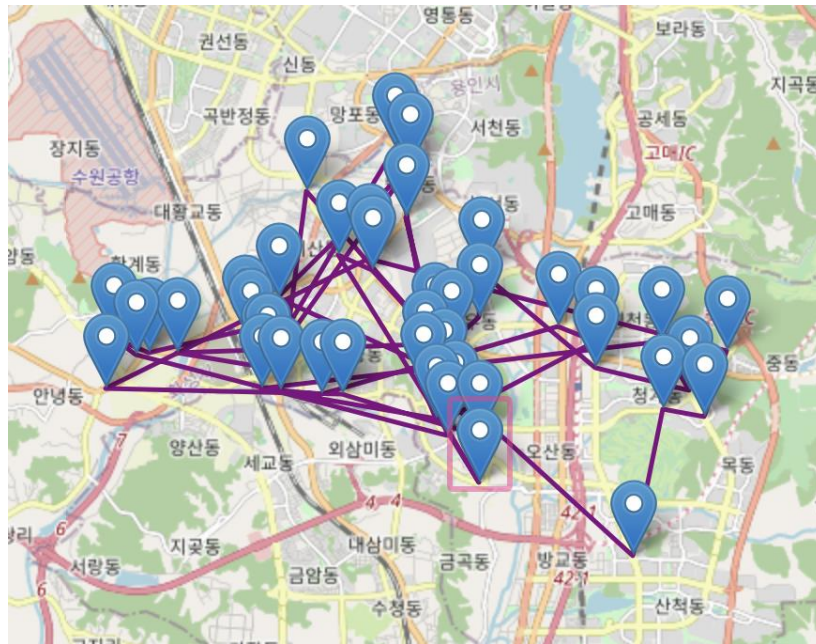
## 화성시 다람쥐버스 노선 제안 3/7

- 출근시간에 가장 승차인원이 많은 정류장 50개 사용
- VRP 알고리즘으로 노선 구현, 노선 3개
- Folium으로 시각화



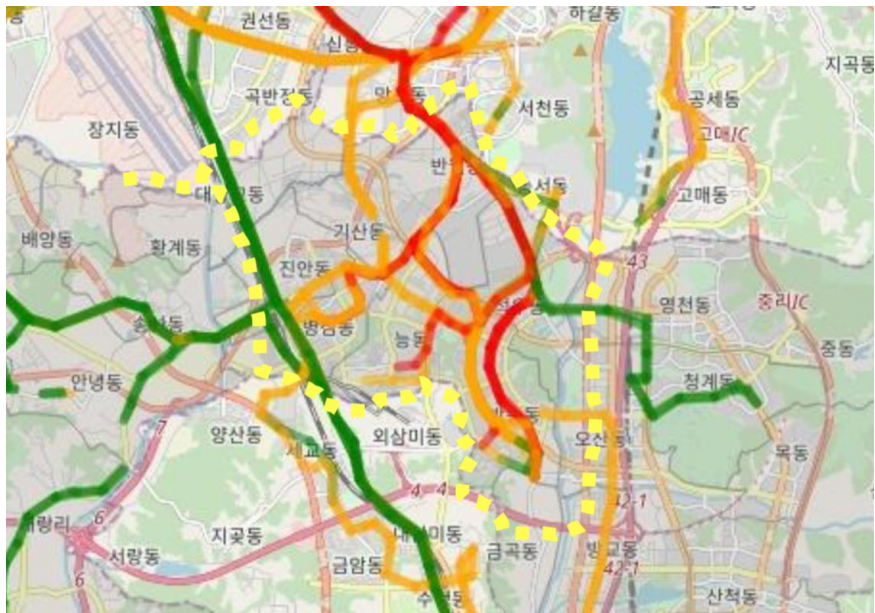
# 화성시 다람쥐버스 노선 제안 4/7

- 출근시간에 가장 승차인원이 많은 정류장 50개 사용
- 봉담읍 지역 제외
- TSP 알고리즘으로 노선 구현
- Folium으로 시각화

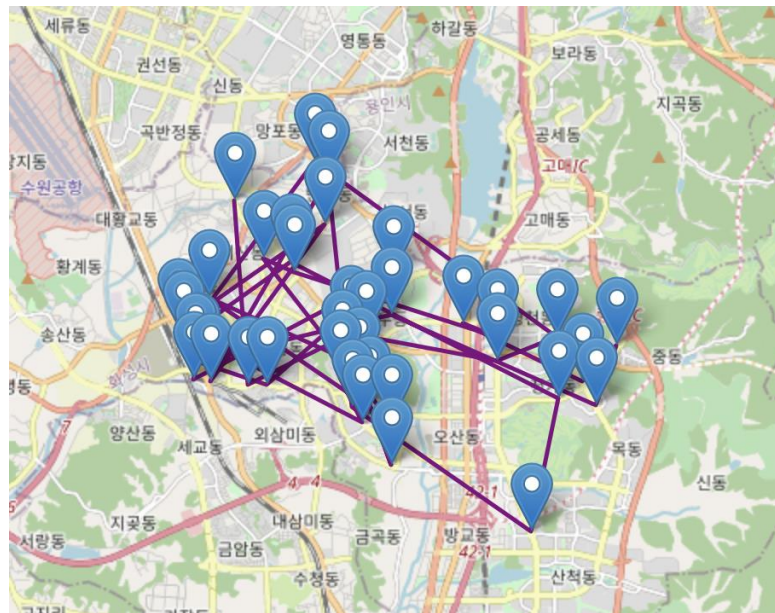




# 화성시 다람쥐버스 노선 제안 5/7



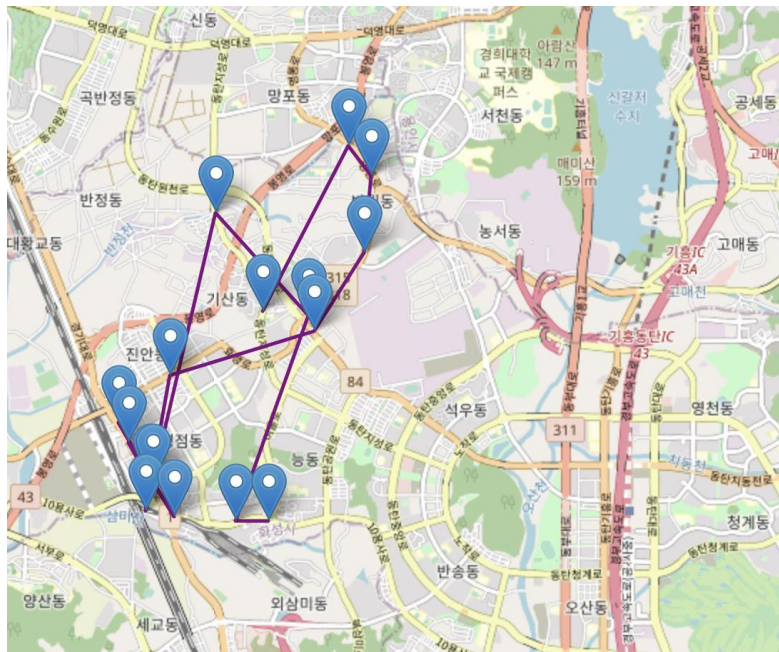
- 혼잡도가 높은 구간의 정류장만 추출



- TSP 알고리즘을 통해 노선 구현
- Folium으로 시각화

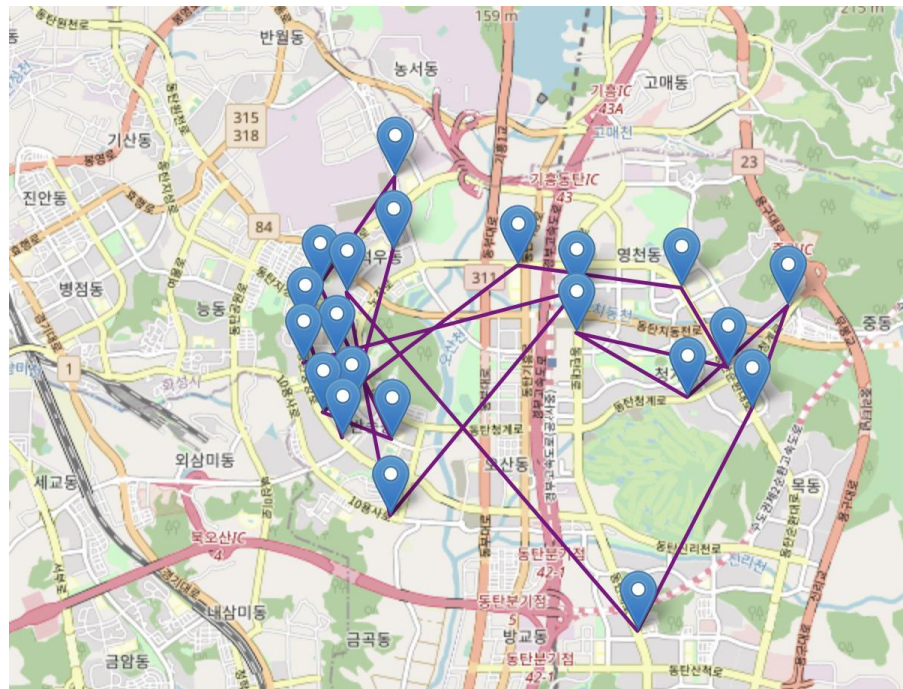
# 화성시 다람쥐버스 노선 제안 6/7

- 출근시간에 가장 승차인원이 많은 정류장 50개 사용
- 병점동 지역만 추출
- TSP 알고리즘으로 노선 구현
- Folium으로 시각화



# 화성시 다람쥐버스 노선 제안 7/7

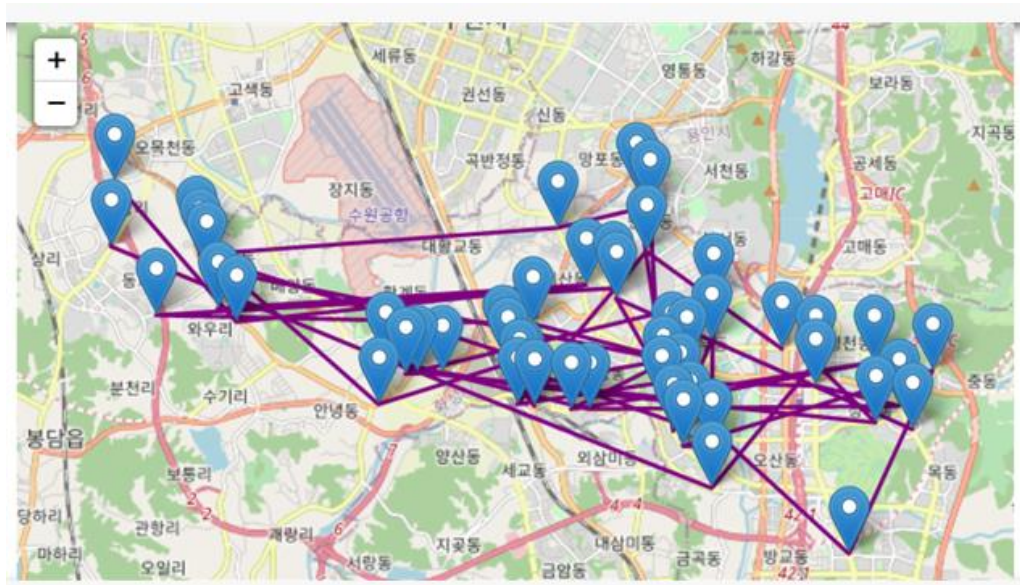
- 출근시간에 가장 승차인원이 많은 정류장 50개 사용
- 동탄동 지역만 추출
- TSP 알고리즘으로 노선 구현
- Folium으로 시각화





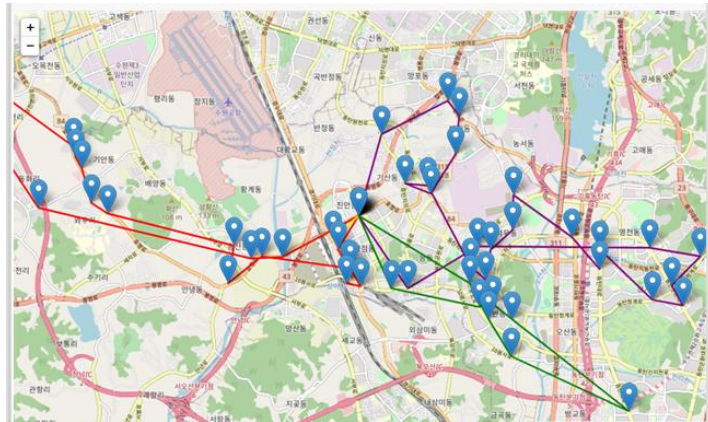
# 프로젝트 한계

1. 노드 간의 직접적인 거리를 weight로 사용해 실제 도로로 이어지지 않은 부분도 있음 (ex. 산 뚫기,,)
2. 거리의 최소합을 찾는거다 보니 현실적인 버스 노선과 살짝 차이가 있음

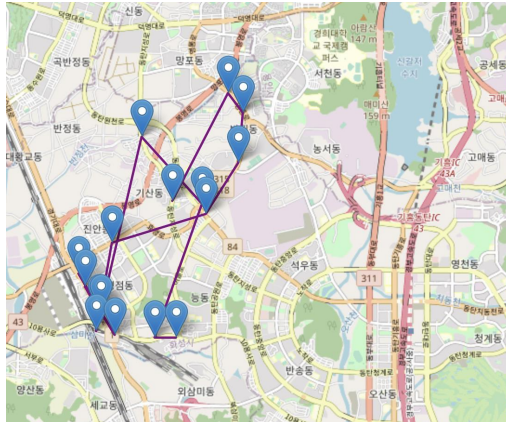




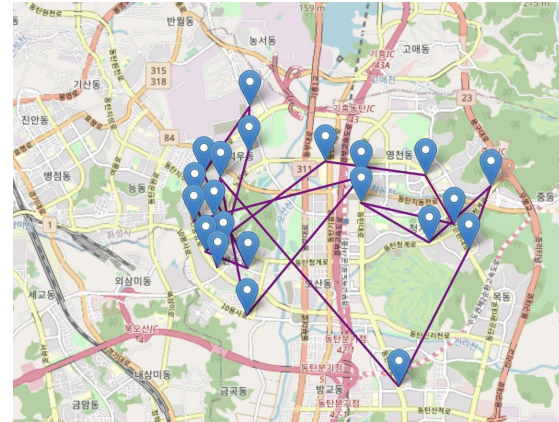
# 실질적인 다람쥐 버스 노선 제안



- 출근시간에 가장 승차인원이 많은 정류장 50개 사용
- VRP 알고리즘을 통해 노선 구현



- 병점동 지역만 추출
- TSP 알고리즘



- 동탄동 지역만 추출
- TSP 알고리즘