課題1,2発表

4月27日

2021夏学期ゼミ #7

M1　増田慧樹

課題１：最短経路探索

1. 疑似コードを元に，Dijkstra法とA*アルゴリズムのコードを実装する.

疑似コードをもとに書いたコード

```
def Dijkstra(s, t):
    ####################ここから埋める######
    D=[]
    visited = []
    previous = []
    for i in range(1,node_number+1):
        D.append(float("inf"))
        visited.append(False)
        previous.append(-1)
```

```
    while len(S_list) > 0:
        j = float('inf')
        for i in S_list:
            if D[i-1] < j:
                j = D[i-1]
                w = S_list[S_list.index(i)]
```

鈴木くんのお手本コード

```
def Dijkstra(s, t):
    ####################ここから埋める########
    D = np.full(node_number,float("inf"))
    visited = np.full(node_number, False)
    previous = np.full(node_number,-1)
```

```
    while len(S_list) > 0:
        w = S_list[np.argmin(D[[ i-1 for i in S_list]])]
```

Numpy配列かリストか

課題1：最短経路探索

比べてみた

- 渋谷ネットワーク（node数1612，link数4832）
- ランダムにノードを2つ抽出 × 5回で計算時間を比べてみる
- Dijkstra法（D）とA*アルゴリズム（A）

経路（コスト）	増田(D)	お手本(D)	増田(A)	お手本(A)
809→389（2259）	0.10962391	0.22182083	2.04547	0.26495695
450→443（199）	0.007128	0.00305796	0.00624204	0.00254893
155→850（339）	0.0129199	0.02138376	0.00865388	0.0044632
1495→191（1646）	0.06618094	0.12815619	0.70323396	0.13846803
1359→1254（1640）	0.08319998	0.13464904	0.13810873	0.03922009

Dijkstra法だと
増田コードの方が早い

A*アルゴリズムだと
お手本コードの方が早い

課題1：最短経路探索

- 一般的に,
  - for文で要素ごとに操作するより，list内包表記の方が早い
  - list要素を一つずつ取り出す演算より，Numpyを使った処理の方が早い（特に配列数が膨大になってくると）

- 今回は簡単なコードで配列同士の大規模な演算がなかったので,
Numpyの強みがそこまで出なかったか.

- A*アルゴリズムは，増田コードだと関数Distanceをfor文で毎回呼び出すので遅くなったのではないか．お手本コードだと呼び出しは1回.

課題1：最短経路探索

比べてみた②

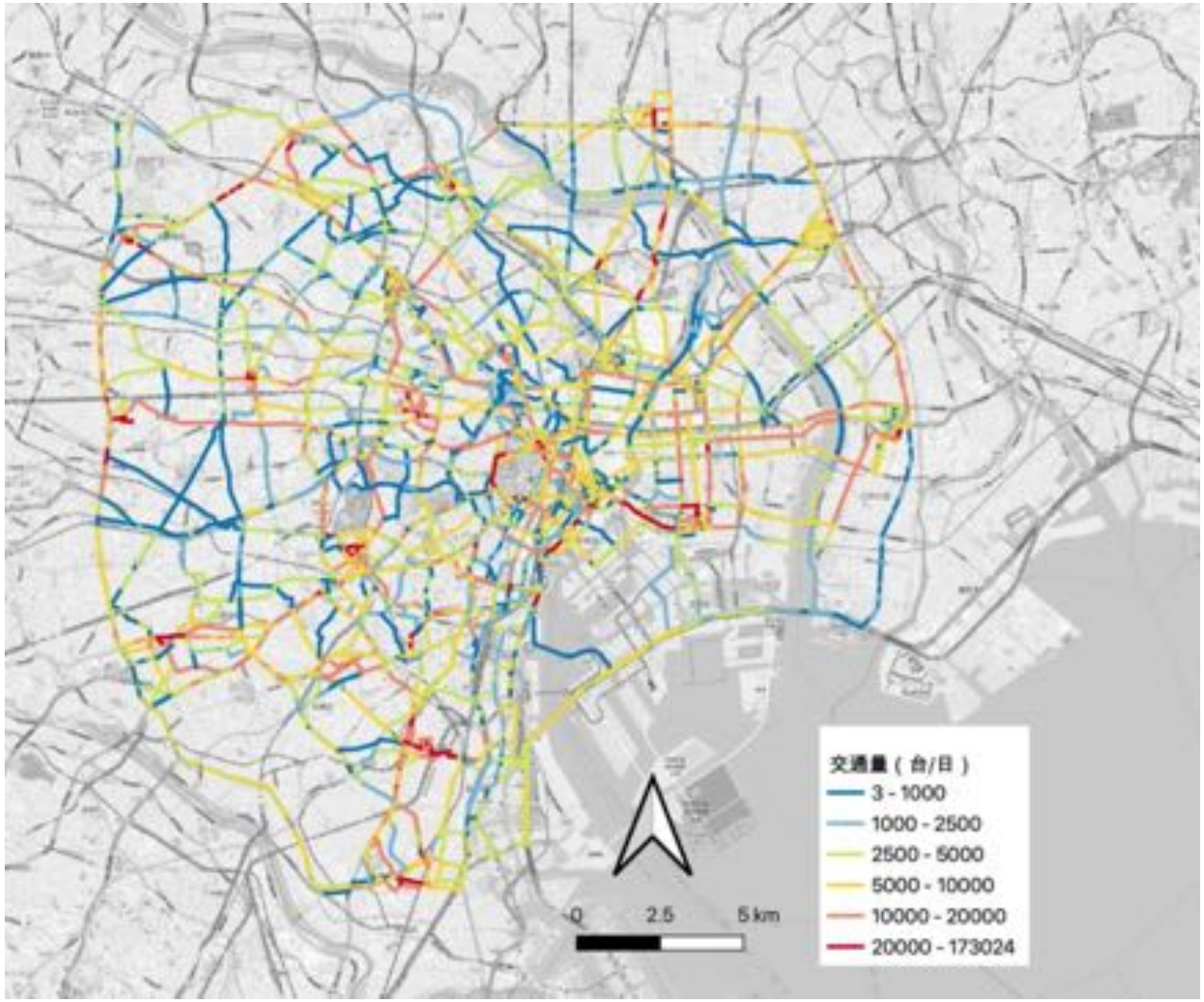
Dijkstra法，Bellman-Ford法，A*アルゴリズムの比較（全てお手本コード）

経路（コスト）	Dijkstra	Bellman-Ford	A*アルゴリズム
809→389（2259）	0.22182083	14.3619411	0.26495695
450→443（199）	0.00305796	12.60955	0.00254893
155→850（339）	0.02138376	12.168011	0.0044632
1495→191（1646）	0.12815619	12.447474	0.13846803
1359→1254 （1640）	0.13464904	12.6685297	0.03922009

- Bellman-Ford法はDijkstra法，A*アルゴリズムに比べて計算時間が大きい.
- 155→850，1359→1254ではDijkstra法に比べてA*アルゴリズムほほうが計算時間が顕著に小さい.

課題２：均衡配分

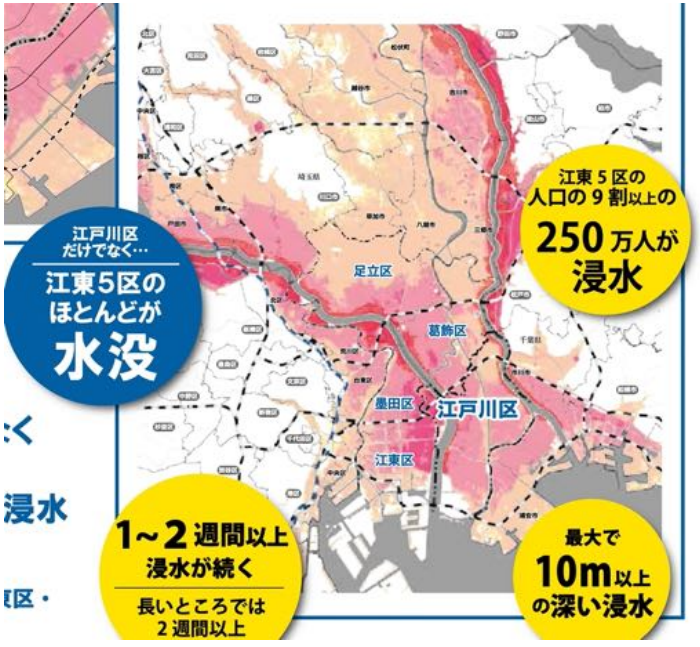
1. 東京都NWで均衡配分を実行し，その結果をQGISに表現する.



課題２：均衡配分

2. 均衡配分におけるパラメータや対象ネットワーク，分布交通量を興味のあるシナリオに沿って変化させて，均衡配分結果を考察してみる.

- 超巨大台風の接近により，江東5区（江東，墨田，江戸川，葛飾，足立）が広域避難を決定
- 江東5区を発地とするトリップ数
  ex)江東区→文京区
  (江東区の自家用車数)*(平時の江東→文京のOD交通量)/(江東区を発地とする平時の総OD交通量)
- 江東5区を着地とするトリップ数 = 0



江戸川区ハザードマップより

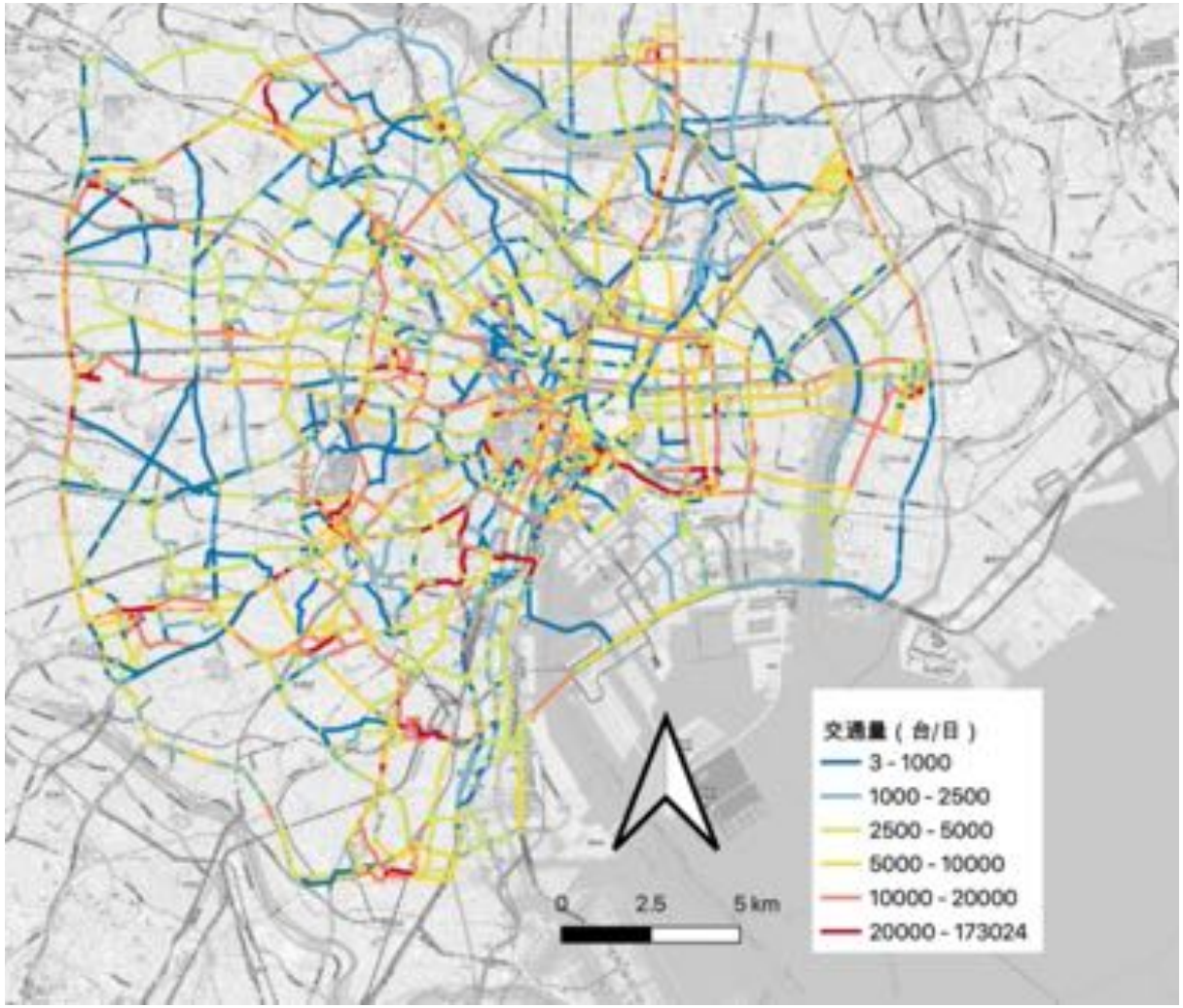
課題２：均衡配分

例）江東区を発地とするOD交通量（台/日）

発地	着地	平時トリップ数	広域避難時トリップ数
江東区	千代田区	9079	11648
江東区	中央区	25513	32731
江東区	港区	10312	13229
江東区	新宿区	2665	3419
江東区	文京区	2633	3378
江東区	台東区	4567	5859
江東区	墨田区	22667	0
江東区	江東区	134148	0
江東区	品川区	3910	5016
江東区	目黒区	772	990
江東区	大田区	5272	6764
江東区	世田谷区	1852	2376
江東区	渋谷区	2164	2776
江東区	中野区	1315	1687
江東区	杉並区	1283	1646
江東区	豊島区	1195	1533
江東区	北区	841	1079
江東区	荒川区	1404	1801
江東区	板橋区	1545	1982
江東区	練馬区	984	1262
江東区	足立区	5043	0
江東区	葛飾区	4878	0
江東区	江戸川区	21144	0

課題２：均衡配分

結果：広域避難シナリオ



課題２：均衡配分

詳細（隅田川周辺）：広域避難シナリオ

課題２：均衡配分

詳細（隅田川周辺）：デフォルトの配分結果

課題２：均衡配分

- ここで表現された結果は「台風上陸の数日前に広域避難の指示が出て，その後十分時間がたったときの状態」と考えられる.

→あまり現実的でない…

- 「荒川や隅田川にかかる橋のところがボトルネックになり，大渋滞が発生する」のような顕著な現象は考えられない

→均衡状態を計算しているので，災害時の突発的な交通現象を表すことはできない

- 何を表現したいかによって適切な手法が違うということがわかった