问题2

In [1]:
```
import os
import pathlib
import plotly
import numpy as np
import pandas as pd
from time import time
from sklearn.cluster import KMeans
from numba import jit, njit, prange
import plotly.express as px
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
ROOTDIR = pathlib.Path(os.path.abspath('.'))
IMG_HTML = ROOTDIR / 'img-html'
IMG_PNG = ROOTDIR / 'img-png'
IMG_SVG = ROOTDIR / 'img-svg'
DATA_RAW = ROOTDIR / 'data-raw'
DATA_COOKED = ROOTDIR / 'data-processed'
```

读数据

In [3]:
```
weak_grid_data = pd.read_csv(DATA_RAW / '附件1 弱覆盖栅格数据(筛选).csv')
exist_site_data = pd.read_csv(DATA_RAW / '附件2 现网站址坐标(筛选).csv')
```

处理数据

最终得到问题2 已有、新建、总的站点数据（具体如何处理的不小心 x 掉（删除）了，但是已经保存成了 csv 文件，直接读取数据即可，凑合看吧）

以下是处理数据过程：

In [4]:
```
# TODO judge
```

```
angle45 = np.pi / 4
angle60 = np.pi / 3
angle90 = np.pi / 2
coverage = [10, 30]
cost = [1, 10]

@jit
def get_theta(x, y, m, n):
    if abs(x - m) < 1e-5: return angle90
    return np.arctan((y - n) / (x - m))

@jit
def get_r_th(theta_delta, cover):
    if theta_delta > angle60: return 0
    return cover * (1 - theta_delta / angle60)

@jit
def check_angle(th1, th2):
    return abs(th1 - th2) < angle45

@jit
def judge_in(site, grid):
    x, y, i, j, cover = site[0], site[1], grid[0], grid[1], coverage[int(site[5]) - 1]
#     print(site[5], cover)
    if abs(x - i) > cover or abs(y - j) > cover: return False
    elif (x - i)**2 + (y - j)**2 > cover**2: return False
    else: return True
```

In [5]:

```
q1_exist_sites_data = pd.read_csv(DATA_COOKED / "question1-exist_site_type_data.csv", index_col=0).iloc[:, 1:]
q1_build_sites_data = pd.read_csv(DATA_COOKED / "question1-new_build_sites.csv", index_col=0)
q1_exist_sites_data, q1_build_sites_data

z_build = pd.DataFrame([0 for _ in range(len(q1_build_sites_data))])
z_exist = pd.DataFrame([0 for _ in range(len(q1_exist_sites_data))])
for i in range(1, 4):
    q1_build_sites_data.insert(i + 1, f"angle{i}", z_build)
    q1_exist_sites_data.insert(i + 1, f"angle{i}", z_exist)
q1_exist_sites_data, q1_build_sites_data
```

Out[5]:
```
(         x     y  angle1  angle2  angle3  type
 0      818  2020       0       0       0     2
 1      713  2013       0       0       0     2
 2     2305   291       0       0       0     2
 3      700  1953       0       0       0     2
 4      949  2293       0       0       0     2
 ...    ...   ...     ...     ...     ...   ...
 1469  2324  1625       0       0       0     2
 1470  1135   852       0       0       0     2
 1471  2053  1818       0       0       0     2
 1472  1432  1797       0       0       0     2
 1473  1584   898       0       0       0     2

 [1474 rows x 6 columns],
         x     y  angle1  angle2  angle3  type
 0     932  2210       0       0       0     2
 1     972  1238       0       0       0     1
 2    1229  1894       0       0       0     2
 3    1311  2005       0       0       0     1
 4    1611  2233       0       0       0     2
 ..    ...   ...     ...     ...     ...   ...
 783  1171   812       0       0       0     1
 784   250  1065       0       0       0     1
 785  2353    24       0       0       0     1
 786  2422   583       0       0       0     2
 787  2249  1453       0       0       0     2

 [788 rows x 6 columns])
```

In [6]:
```python
@jit
def angle(hu):
    return hu / np.pi * 180
```

In [7]:
```python
@jit
def cal_sites_angle():
    q1_weak_grid_data_np = np.array(weak_grid_data.copy()).astype(np.float32)
    q1_exist_sites_data_np = np.array(q1_exist_sites_data.copy()).astype(np.float32)
    q1_build_sites_data_np = np.array(q1_build_sites_data.copy()).astype(np.float32)
    l1, l2, l3 = len(q1_weak_grid_data_np), len(q1_exist_sites_data_np), len(q1_build_sites_data_np)
    print(l1)
    site_data = q1_build_sites_data_np
    l2 = len(site_data)
    print(l2)

    for i in prange(l2):
        if i % 50 == 0:
```

```
            print(i, '/', l2)
        site_data[i, 2] = 0
        site_data[i, 3] = angle60 * 2
        site_data[i, 4] = angle60 * 4

        include_weak_grid_list = []
        for j in prange(l1):
            if judge_in(site_data[i, :], q1_weak_grid_data_np[j, :]):
#                 print("/", end = '')
                include_weak_grid_list.append(q1_weak_grid_data_np[j, :])
        if len(include_weak_grid_list) == 0:
            continue
        include_weak_grid_list = pd.DataFrame(include_weak_grid_list)
        include_weak_grid_list.sort_values(by=2, inplace=True, ascending=False)
        include_weak_grid_list = np.array(include_weak_grid_list).reshape(-1, 3)

        site_data[i, 2] = get_theta(
            site_data[i, 0], site_data[i, 1],
            include_weak_grid_list[0, 0], include_weak_grid_list[0, 1],
        )

        if not len(include_weak_grid_list) > 1:
            continue
        idx = 1
        while idx < len(include_weak_grid_list):
#             print(",", end = '')
            th = get_theta(
                site_data[i, 0], site_data[i, 1],
                include_weak_grid_list[idx, 0], include_weak_grid_list[idx, 1],
            )
            idx += 1
            if check_angle(th, site_data[i, 2]):
                site_data[i, 3] = th
                break

        if not len(include_weak_grid_list) > 2:
            continue
        while idx < len(include_weak_grid_list):
#             print(".", end = '')
            th = get_theta(
                site_data[i, 0], site_data[i, 1],
                include_weak_grid_list[idx, 0], include_weak_grid_list[idx, 1],
            )
            idx += 1
            if check_angle(th, site_data[i, 2]) and check_angle(th, site_data[i, 3]):
#                 print(444, end = '')
```

```
                site_data[i, 4] = th
                break
#     print(site_data)
    return site_data
```

In [8]:
```
q1_build_sites_data_np = cal_sites_angle()
```

```
182807
788
0 / 788
50 / 788
100 / 788
150 / 788
200 / 788
250 / 788
300 / 788
350 / 788
400 / 788
450 / 788
500 / 788
550 / 788
600 / 788
650 / 788
700 / 788
750 / 788
```

In [9]:
```
pd.DataFrame(q1_build_sites_data_np, columns=['x','y','θ1','θ2','θ3','type'])
```

Out[9]:

	x	y	θ1	θ2	θ3	type
0	932.0	2210.0	1.570796	2.094395	4.188790	2.0
1	972.0	1238.0	1.570796	2.094395	4.188790	1.0
2	1229.0	1894.0	1.570796	2.094395	4.188790	2.0
3	1311.0	2005.0	1.570796	2.094395	4.188790	1.0
4	1611.0	2233.0	1.570796	1.570796	4.188790	2.0
...	...	...	...	...	...	...
783	1171.0	812.0	1.570796	1.570796	1.165905	1.0
784	250.0	1065.0	1.570796	1.152572	1.570796	1.0
785	2353.0	24.0	1.570796	2.094395	4.188790	1.0
786	2422.0	583.0	1.570796	1.012197	1.107149	2.0
787	2249.0	1453.0	-1.107149	-1.438245	-1.107149	2.0

788 rows × 6 columns

到这就处理数据代码没有了，只有处理好的 csv 文件，直接读取即可

In [10]:
```
d1 = pd.read_csv(DATA_COOKED / 'question2-exist_sites_angle.csv', index_col=0)
d2 = pd.read_csv(DATA_COOKED / 'question2-build_sites_angle.csv', index_col=0)
data = pd.concat([d1, d2], axis=0)
data.to_csv(DATA_COOKED / 'question2-total_sites_angle.csv')  # 总站点1（后面还会加入问题2中新的新建站点）
```

计算总业务量

In [11]:
```
@jit
def cal_total_business():
    total_bussiness = 0
    data = np.array(pd.read_csv(DATA_COOKED / 'question2-total_sites_angle.csv', index_col=0))
    LEN = len(data)

    grid = np.array(weak_grid_data.copy()).astype(np.float32)
    L = len(grid)

    for i in prange(LEN):
        for j in prange(L):
```

```
            if grid[j,3]:
                continue
            if judge_in(data[i], grid[j]):
                x1,y1=data[i,0],data[i,1]
                x2,y2=grid[i,1],grid[j,1]
                if get_r_th(
                    get_theta(x1 - x2, y1 - y2) - data[i, 3],
                    coverage[int(data[i, 5]) - 1]
                )**2 > (x1 - x2)**2 + (y1 - y2)**2:
                    grid[j,3] = 1
                    total_bussiness += grid[j, 2]
    return total_bussiness
```

In [12]:
```
weak_grid_data
```

Out[12]:

	x	y	traffic
0	66	1486	140.581390
1	67	1486	140.518829
2	177	1486	48.919178
3	187	1486	4.322495
4	284	1486	71.528404
...	...	...	...
182802	2350	2123	0.178571
182803	2353	2123	5.159708
182804	2354	2123	5.134017
182805	2355	2123	2.599999
182806	2372	2123	57.814999

182807 rows × 3 columns

In [13]:
```
weak_grid_data_cp  = weak_grid_data.copy()
weak_grid_data_cp
```

Out[13]:

	x	y	traffic
0	66	1486	140.581390
1	67	1486	140.518829
2	177	1486	48.919178
3	187	1486	4.322495
4	284	1486	71.528404
...	...	...	...
182802	2350	2123	0.178571
182803	2353	2123	5.159708
182804	2354	2123	5.134017
182805	2355	2123	2.599999
182806	2372	2123	57.814999

182807 rows × 3 columns

In [14]:
```
weak_grid_data_cp.insert(3, "ed", pd.DataFrame([0 for _ in range(len(weak_grid_data))]))
weak_grid_data_cp
```

Out[14]:

	x	y	traffic	ed
0	66	1486	140.581390	0
1	67	1486	140.518829	0
2	177	1486	48.919178	0
3	187	1486	4.322495	0
4	284	1486	71.528404	0
...	...	...	...	...
182802	2350	2123	0.178571	0
182803	2353	2123	5.159708	0
182804	2354	2123	5.134017	0
182805	2355	2123	2.599999	0
182806	2372	2123	57.814999	0

182807 rows × 4 columns

In [15]:

```
@jit
def cal_total_business():
    t0 = time()
    total_bussiness = 0
    data = np.array(pd.read_csv(DATA_COOKED / 'question2-total_sites_angle.csv', index_col=0))
    LEN = len(data)

    grid = np.array(weak_grid_data_cp.copy()).astype(np.float32)
    L = len(grid)

    for i in prange(LEN):
        if i % 300 == 0:
            print(i, '/', LEN)
        for j in prange(L):
            if grid[j,3]:
                continue
            if judge_in(data[i], grid[j]):
                if not np.random.randint(10):
                    continue
                x1,y1=data[i,0],data[i,1]
                x2,y2=grid[i,1],grid[j,1]
                grid[j,3] = 1
                total_bussiness += grid[j, 2]
```

```
    print('用时：', time() - t0)
    return total_bussiness, data, grid
```

In [16]:
```
to_bu = cal_total_business()  # 1000s
```

```
0 / 1576
300 / 1576
600 / 1576
900 / 1576
1200 / 1576
1500 / 1576
用时：  741.3745419979095
```

In [17]:
```
# todo 计算总业务量
total_bussiness, data, grid = to_bu
total_bussiness, data, grid
```

Out[17]:
```
(3922204.01093581,
 array([[9.3200000e+02, 2.2100000e+03, 1.5707964e+00, 2.0943952e+00,
         4.1887903e+00, 2.0000000e+00],
        [9.7200000e+02, 1.2380000e+03, 1.5707964e+00, 2.0943952e+00,
         4.1887903e+00, 1.0000000e+00],
        [1.2290000e+03, 1.8940000e+03, 1.5707964e+00, 2.0943952e+00,
         4.1887903e+00, 2.0000000e+00],
        ...,
        [2.3530000e+03, 2.4000000e+01, 1.5482696e+00, 4.7776027e+00,
         5.3622860e+00, 1.0000000e+00],
        [2.4220000e+03, 5.8300000e+02, 1.7090038e+00, 4.0597887e+00,
         6.1582727e+00, 2.0000000e+00],
        [2.2490000e+03, 1.4530000e+03, 4.2226570e+00, 4.9158216e+00,
         1.0833534e+00, 2.0000000e+00]]),
 array([[  66.      , 1486.      ,  140.58139 ,    0.      ],
        [  67.      , 1486.      ,  140.51883 ,    0.      ],
        [ 177.      , 1486.      ,   48.919178,    0.      ],
        ...,
        [2354.      , 2123.      ,    5.134017,    0.      ],
        [2355.      , 2123.      ,    2.599999,    0.      ],
        [2372.      , 2123.      ,   57.815   ,    0.      ]],
       dtype=float32))
```

In [18]:
```
remain_ggrid = pd.DataFrame(grid).copy()
remain_ggrid
```

Out[18]:

	0	1	2	3
0	66.0	1486.0	140.581390	0.0
1	67.0	1486.0	140.518829	0.0
2	177.0	1486.0	48.919178	0.0
3	187.0	1486.0	4.322495	0.0
4	284.0	1486.0	71.528404	1.0
...	...	...	...	...
182802	2350.0	2123.0	0.178571	0.0
182803	2353.0	2123.0	5.159708	0.0
182804	2354.0	2123.0	5.134017	0.0
182805	2355.0	2123.0	2.599999	0.0
182806	2372.0	2123.0	57.814999	0.0

182807 rows × 4 columns

In [19]:
```
# todo 根据总业务量排序
cond = remain_ggrid.iloc[:, 3] == 0
remain_grid = remain_ggrid[cond]
remain_grid.sort_values(by=2, inplace=True, ascending=False)
remain_grid
```

Out[19]:

	0	1	2	3
103182	844.0	1962.0	47795.011719	0.0
96789	1356.0	2271.0	43295.000000	0.0
110362	869.0	2292.0	32200.953125	0.0
57442	881.0	1256.0	24688.421875	0.0
107373	1096.0	1658.0	23715.160156	0.0
...	...	...	...	...
179506	2212.0	2423.0	0.000192	0.0
181329	2195.0	2433.0	0.000192	0.0
179502	2208.0	2423.0	0.000192	0.0
11838	2223.0	2460.0	0.000192	0.0
154838	2059.0	2358.0	0.000192	0.0

108338 rows × 4 columns

kmeans 聚类

In [20]:
```
# todo kmeans 聚类
kmeans = KMeans(n_clusters=100)
kmeans.fit(remain_grid.iloc[:, :2])
kmeans.labels_
```

Out[20]:
```
array([24, 61, 24, ..., 45, 45, 89])
```

In [21]:
```
remain_grid.insert(4, "labels", kmeans.labels_)
remain_grid
```

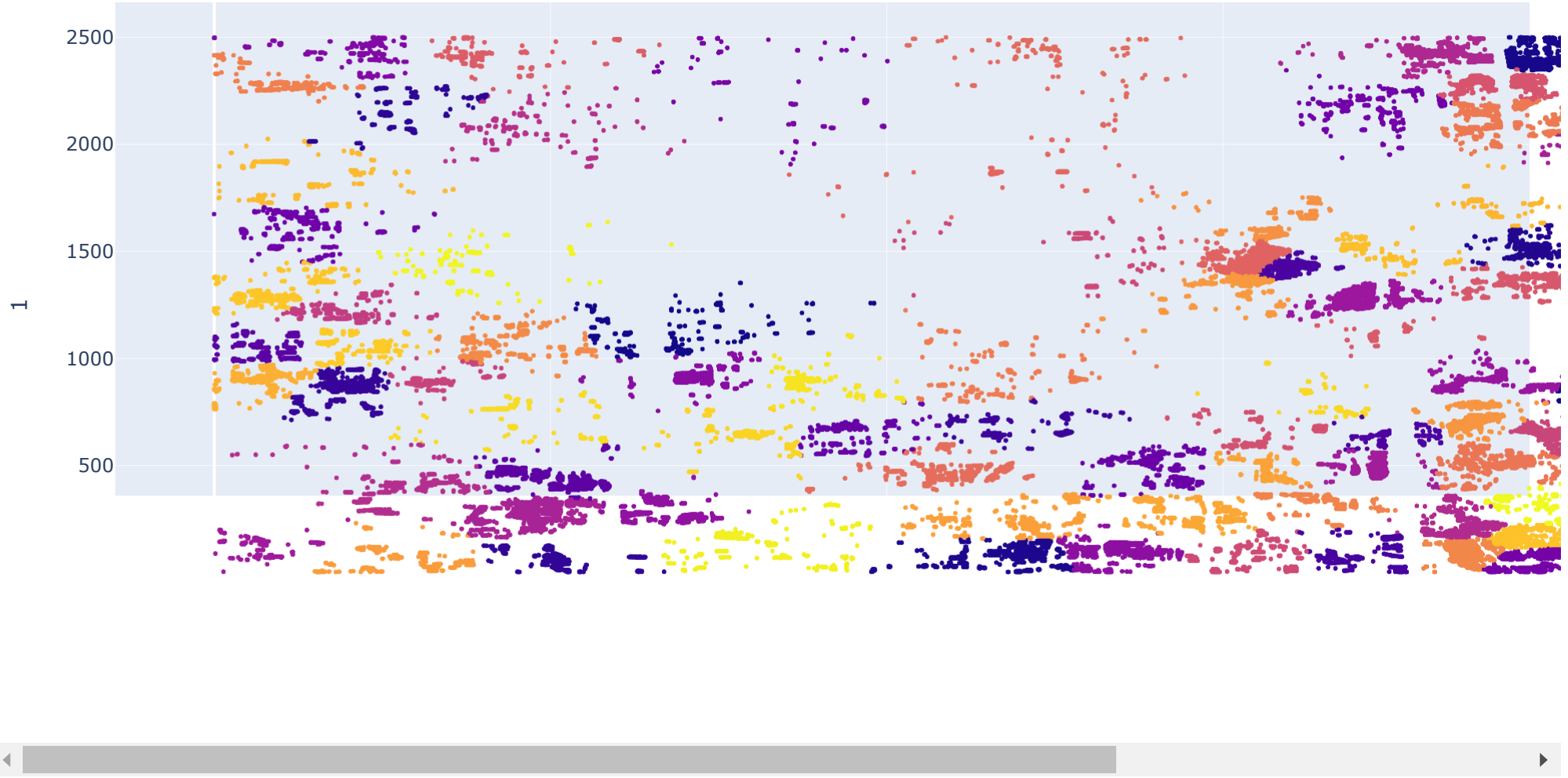
Out[21]:

	0	1	2	3	labels
103182	844.0	1962.0	47795.011719	0.0	24
96789	1356.0	2271.0	43295.000000	0.0	61
110362	869.0	2292.0	32200.953125	0.0	24
57442	881.0	1256.0	24688.421875	0.0	1
107373	1096.0	1658.0	23715.160156	0.0	60
...	...	...	...	...	...
179506	2212.0	2423.0	0.000192	0.0	45
181329	2195.0	2433.0	0.000192	0.0	45
179502	2208.0	2423.0	0.000192	0.0	45
11838	2223.0	2460.0	0.000192	0.0	45
154838	2059.0	2358.0	0.000192	0.0	89

108338 rows × 5 columns

In [22]:

```
# todo 绘制聚类结果
fig = px.scatter(data_frame=remain_grid, x=0, y=1, color='labels')
fig.update_traces(marker={"size": 3})
fig.write_html(IMG_HTML / "question2-kmeans100.html")
fig.write_image(IMG_PNG / "question2-kmeans100.png")
fig.write_image(IMG_SVG / "question2-kmeans100.svg")
fig.show()
del fig
```



根据聚类结果和总业务量选择新建基站

In [23]:
```
# 业务量
now_bn = 5905185.385538206  # 现有业务量
targe_bn = 6350607.1031652  # 目标业务量
```

In [24]:
```
def existed(x, y):
    return x not in data[:, 0] and y not in data[:, 1]
```

In [25]:
```
remain_grid
```

Out[25]:

	0	1	2	3	labels
103182	844.0	1962.0	47795.011719	0.0	24
96789	1356.0	2271.0	43295.000000	0.0	61
110362	869.0	2292.0	32200.953125	0.0	24
57442	881.0	1256.0	24688.421875	0.0	1
107373	1096.0	1658.0	23715.160156	0.0	60
...	...	...	...	...	...
179506	2212.0	2423.0	0.000192	0.0	45
181329	2195.0	2433.0	0.000192	0.0	45
179502	2208.0	2423.0	0.000192	0.0	45
11838	2223.0	2460.0	0.000192	0.0	45
154838	2059.0	2358.0	0.000192	0.0	89

108338 rows × 5 columns

In [26]:

```
angle37_5 = np.pi * 37.5 / 180
anglex = np.pi * 60 / 180

@jit
def is_far(x,y,m,n,c):
    return abs(x-m) > c or abs(y-n) > c
@jit
def is_in(x,y,m,n,c):
    return (x-m)**2 + (y-n)**2 < c**2
@jit
def calcal(remain_grid=remain_grid):
    total_data = []
    now_bn = 5905185.385538206
    targe_bn = 6350607.1031652

    remain_grid = np.array(remain_grid.copy())
    for i in prange(len(remain_grid)):
        if not remain_grid[i, 3]:
            x, y, t = int(remain_grid[i, 0]), int(remain_grid[i, 1]), np.random.randint(2) + 1
            t1, t2, t3 = [(angle60*2*(j-1)+i+np.random.uniform(-anglex, anglex)) % (np.pi*2) \
                          for j in range(2, 5)]
            total_data.append([x, y, t1, t2, t3, t])
            now_bn += remain_grid[i, 2]
```

```
            for j in prange(i + 1, len(remain_grid)):
                m,n,c = int(remain_grid[j, 0]), int(remain_grid[j, 1]), coverage[t - 1]
                if not is_far(x,y,m,n,c) and is_in(x,y,m,n,c):
                    now_bn += remain_grid[j, 2]
                    remain_grid[j, 3] = 1
            if now_bn > targe_bn:
                break
        if i % 50 == 0:
            print(i, '/', len(remain_grid))
    return total_data, now_bn
```

In [27]:
```
total_data, tolbn = calcal()
```

```
0 / 108338
```

In [28]:
```
print(tolbn)
pd.DataFrame(total_data)
```

```
6358173.907919989
```

Out[28]:

	0	1	2	3	4	5
0	844	1962	2.941118	3.542697	5.843110	1
1	1356	2271	2.985915	5.858931	1.994340	2
2	869	2292	3.323346	0.426135	2.267055	1
3	881	1256	4.109999	0.009277	3.834270	2
4	1096	1658	0.579936	2.522800	3.615235	1
5	683	2198	6.118514	3.457958	4.685333	2
6	865	2012	2.266906	2.992941	5.307013	2
7	1335	2206	4.453241	6.275982	1.811592	1
8	1357	1086	5.313147	0.607825	2.838493	2
9	675	1957	5.664622	1.837217	3.970463	1
10	1019	1593	0.152704	2.747127	4.986251	2
11	935	1665	0.874917	4.206048	4.716827	1
12	1489	1228	2.727830	5.605429	0.155014	2
13	699	2014	2.525903	6.121284	1.445451	1
14	1172	1797	4.327217	5.576492	1.411652	1
15	1314	1125	0.174048	1.978144	2.504269	2

In [29]:
```
pd.DataFrame(
    total_data,
    columns=['x','y','θ1','θ2','θ3','type'],
).to_csv(DATA_COOKED / "question2-add_sites_angle.csv")
```

In [30]:
```
total_sites_data = pd.read_csv(DATA_COOKED / "question2-total_sites_angle.csv", index_col=0)
total_sites_data
add_sites_data = pd.read_csv(DATA_COOKED / "question2-add_sites_angle.csv", index_col=0)
add_sites_data
total_sites_data.append(add_sites_data).to_csv(DATA_COOKED / "question2-total_sites_angle.csv")
total_sites_data  # 最终所有站点数据
```

Out[30]:

	x	y	θ1	θ2	θ3	type
0	932.0	2210.0	1.570796	2.094395	4.188790	2.0
1	972.0	1238.0	1.570796	2.094395	4.188790	1.0
2	1229.0	1894.0	1.570796	2.094395	4.188790	2.0
3	1311.0	2005.0	1.570796	2.094395	4.188790	1.0
4	1611.0	2233.0	1.570796	1.570796	4.188790	2.0
...	...	...	...	...	...	...
783	1171.0	812.0	6.095762	2.418297	4.759339	1.0
784	250.0	1065.0	0.680948	3.113659	3.892996	1.0
785	2353.0	24.0	1.548270	4.777603	5.362286	1.0
786	2422.0	583.0	1.709004	4.059789	6.158273	2.0
787	2249.0	1453.0	4.222657	4.915822	1.083353	2.0

1576 rows × 6 columns

In [31]:
```
SUM = 0
for i in range(len(total_sites_data)):
    SUM += cost[int(total_sites_data.iloc[i, 5]) - 1]
SUM  # 问题2新建站点之前总站点数
```

Out[31]:
```
10324
```

In [32]:
```
add_sites_data
```

Out[32]:

	x	y	θ1	θ2	θ3	type
0	844	1962	2.941118	3.542697	5.843110	1
1	1356	2271	2.985915	5.858931	1.994340	2
2	869	2292	3.323346	0.426135	2.267055	1
3	881	1256	4.109999	0.009277	3.834270	2
4	1096	1658	0.579936	2.522800	3.615235	1
5	683	2198	6.118514	3.457958	4.685333	2
6	865	2012	2.266906	2.992941	5.307013	2
7	1335	2206	4.453241	6.275982	1.811592	1
8	1357	1086	5.313147	0.607825	2.838493	2
9	675	1957	5.664622	1.837217	3.970463	1
10	1019	1593	0.152704	2.747127	4.986251	2
11	935	1665	0.874917	4.206048	4.716827	1
12	1489	1228	2.727830	5.605429	0.155014	2
13	699	2014	2.525903	6.121284	1.445451	1
14	1172	1797	4.327217	5.576492	1.411652	1
15	1314	1125	0.174048	1.978144	2.504269	2

In [33]:
```
for i in range(len(add_sites_data)):
    SUM += cost[int(add_sites_data.iloc[i, 5]) - 1]
SUM  # 问题2新建站点之后总站点数
```

Out[33]:
```
10412
```

绘制所有站点散点图

In [34]:
```
def plot_4kinds_sites():
    trace1 = go.Scatter(
        mode='markers',
        x=weak_grid_data.loc[:, "x"],
        y=weak_grid_data.loc[:, "y"],
        marker={"color": "blue", "size": 0.5},
        name="弱覆盖点",
```

```
    )
    trace2 = go.Scatter(
        mode='markers',
        x=total_sites_data.iloc[:1474, 0],
        y=total_sites_data.iloc[:1474, 1],
        marker={"color": "pink", "size": 4},
        name="现网基站",
    )
    trace3 = go.Scatter(
        mode='markers',
        x=total_sites_data.iloc[1474:, 0],
        y=total_sites_data.iloc[1474:, 1],
        marker={"color": "orange", "size": 5},
        name="新建基站-1",
    )
    trace4 = go.Scatter(
        mode='markers',
        x=add_sites_data.iloc[:, 0],
        y=add_sites_data.iloc[:, 1],
        marker={"color": "red", "size": 6},
        name="新建基站-2",
    )

    data = [trace1, trace2, trace3]
    fig = go.Figure(data=data)
    fig.write_html(IMG_HTML / "question2-3kinds_sites.html")
    fig.write_image(IMG_PNG / "question2-3kinds_sites.png")
    fig.write_image(IMG_SVG / "question2-3kinds_sites.svg")
#     fig.show()
    del fig

    data = [trace1, trace2, trace3, trace4]
    fig = go.Figure(data=data)
    fig.write_html(IMG_HTML / "question2-4kinds_sites.html")
    fig.write_image(IMG_PNG / "question2-4kinds_sites.png")
    fig.write_image(IMG_SVG / "question2-4kinds_sites.svg")
#     fig.show()
    del fig
    return None
```

In [35]:
```
plot_4kinds_sites()
```

这里文件太大，不方便展示，自行查看 `question2-3kinds_sites.html` `question2-4kinds_sites.html` 文件

In [ ]: