{ stationList:
  [
{ d: '3', x: 120.136037, y: 30.282492 },
{ d: '5', x: 120.151129, y: 30.278562 },
{ d: '-8', x: 120.143008, y: 30.274695 },
{ d: '5', x: 120.153824, y: 30.273884 },
{ d: '-5', x: 120.163741, y: 30.276129 },
{ d: '8', x: 120.142577, y: 30.286671 },
{ d: '-7', x: 120.154435, y: 30.285049 },
{ d: '-1', x: 120.160184, y: 30.272886 },
{ d: '23', x: 120.162519, y: 30.28561 },
{ d: '-12', x: 120.156195, y: 30.277689 },
{ d: '-11', x: 120.147787, y: 30.271576 }],

distance:
  [
[0, 2395, 1389, 2596, 3424, 1467, 3437, 3747, 6006, 2380, 2370, 1432],
[2329, 0, 1925, 1308, 2439, 2115, 2149, 2459, 4718, 1092, 2006, 3173],
[2152, 2093, 0, 2342, 2711, 1938, 3183, 2731, 5752, 2126, 932, 2301],
[2436, 2157, 1738, 0, 1914, 2222, 1893, 2272, 4496, 836, 1819, 2986],
[3907, 3412, 2996, 1622, 0, 3761, 2287, 1977, 3564, 1799, 2358, 4244],
[2766, 2487, 3083, 2945, 3520, 0, 1954, 4096, 4935, 2729, 3643, 4229],
[1940, 1661, 2257, 2119, 1867, 1726, 0, 3270, 4111, 1903, 3205, 3403],
[3672, 3393, 2974, 1292, 1600, 3458, 2763, 0, 4622, 2072, 2022, 4126],
[2896, 2617, 3213, 2621, 2062, 2682, 1364, 2931, 0, 2238, 3357, 4359],
[2807, 2312, 1896, 1311, 1133, 2661, 1108, 2002, 3715, 0, 2020, 3144],
[2412, 1566, 1501, 1381, 1858, 2449, 3011, 1878, 4870, 1584, 0, 2471],
[1594, 2847, 1841, 3048, 3876, 2694, 3889, 4199, 6458, 2832, 2332, 0]],

depot: { x: 120.13054, y: 30.276753 } }

目前实现的方法分为两大类，一类是 uncapacitated，一类是 capacitated。其中 capacitated 又有三种不同的实现。不管是哪种方法都在得到路径后对路径进行优化，就是说在货车容量允许的情况下尽量拉走或者尽量放置自行车，避免由分块带来的多次访问相同站点的情况。

（1）在得到 tsp 路径后进行分块，分为零块，正块和负块。正块与负块之间的最小距离是正块与负块之间距离最小的两个点的距离。这种方法在遍历块的时候先遍历最前面的零块，然后遍历正块，到达匹配点时去遍历对应的负块，包括负块后面的零块，然后回到正块匹配点的下一个点继续遍历正块，然后再遍历正块后面的零块。对应下面的 Ktimescapacitated BSP。

（2）在得到 tsp 路径后进行分块，分为零块，正块和负块。正块与负块之间的最小距离是正块中两个相邻点分别到负块起始点和负块结束点（如果负块后面有零块就是负块后面最后一个零块的结束点）的距离之和加上正块与下一个正块（如果是最后一个正块则不考虑）之间距离的最小值。这种方法在遍历块的时候先遍历最前面的零块，然后遍历正块，到达匹配点时去遍历对应的负块，包括负块后面的零块，然后回到正块匹配点的下一个点继续遍历正

块，然后再遍历正块后面的零块。对应下面的 Capacitated BSP。

（3）在得到 tsp 路径后进行分块，分为正块和负块。正块与负块之间的最小距离是正块的最后一个点到负块的最后一个点的距离加上负块的第一个点到下一个正块起始点的距离之和。这种方法在遍历块的时候先遍历正块，然后倒着遍历负块，然后再回到下一个正块。对应下面的 noZeroCapacitated BSP。

Uncapacitated BSP sum cost: 19701
0(3) 5(8) 8(23) 4(-5) 7(-1) 3(5) 9(-12) 6(-7) 1(5) 10(-11) 2(-8)

Ktimescapacitated BSP sum cost: 26498
1(5) 9(-4) 3(5) 7(-1) 4(-5) 8(23) 6(-7) 9(-8) 5(8) 0(3) 2(-8) 10(-11)
P pieces:1
  0: Matching 0: 8(15)
N pieces:1
  0: Matching 0; 9(-8) 6(-7)
Zero pieces:5
  0: 7(-1) 3(1)
  1: 3(4) 9(-4)
  2: 1(5) 10(-5)
  3: 10(-6) 2(-8) 0(3) 5(8) 8(3)
  4: 8(5) 4(-5)

Capacitated BSP sum cost: 24133
8(23) 4(-5) 7(-1) 3(5) 9(-12) 2(-8) 0(3) 5(8) 6(-7) 1(5) 10(-11)
P pieces:1
  0: Matching 0: 5(3) 8(12)
N pieces:1
  0: Matching 0; 9(-2) 6(-7) 1(5) 10(-11)
Zero pieces:2
  0: 2(-8) 0(3) 5(5)
  1: 8(11) 4(-5) 7(-1) 3(5) 9(-10)

noZeroCapacitated BSP sum cost: 26836
8(23) 4(-5) 7(-1) 3(5) 9(-12) 6(-2) 5(8) 0(3) 2(-8) 6(-5) 1(5) 10(-11)
P pieces:2
  0: Matching 0: 2(-4) 0(3) 5(8) 8(8)
  1: Matching 1: 8(15)
N pieces:2
  0: Matching 0; 4(-5) 7(-1) 3(5) 9(-12) 6(-2)
  1: Matching 1; 6(-5) 1(5) 10(-11) 2(-4)
Zero pieces:0