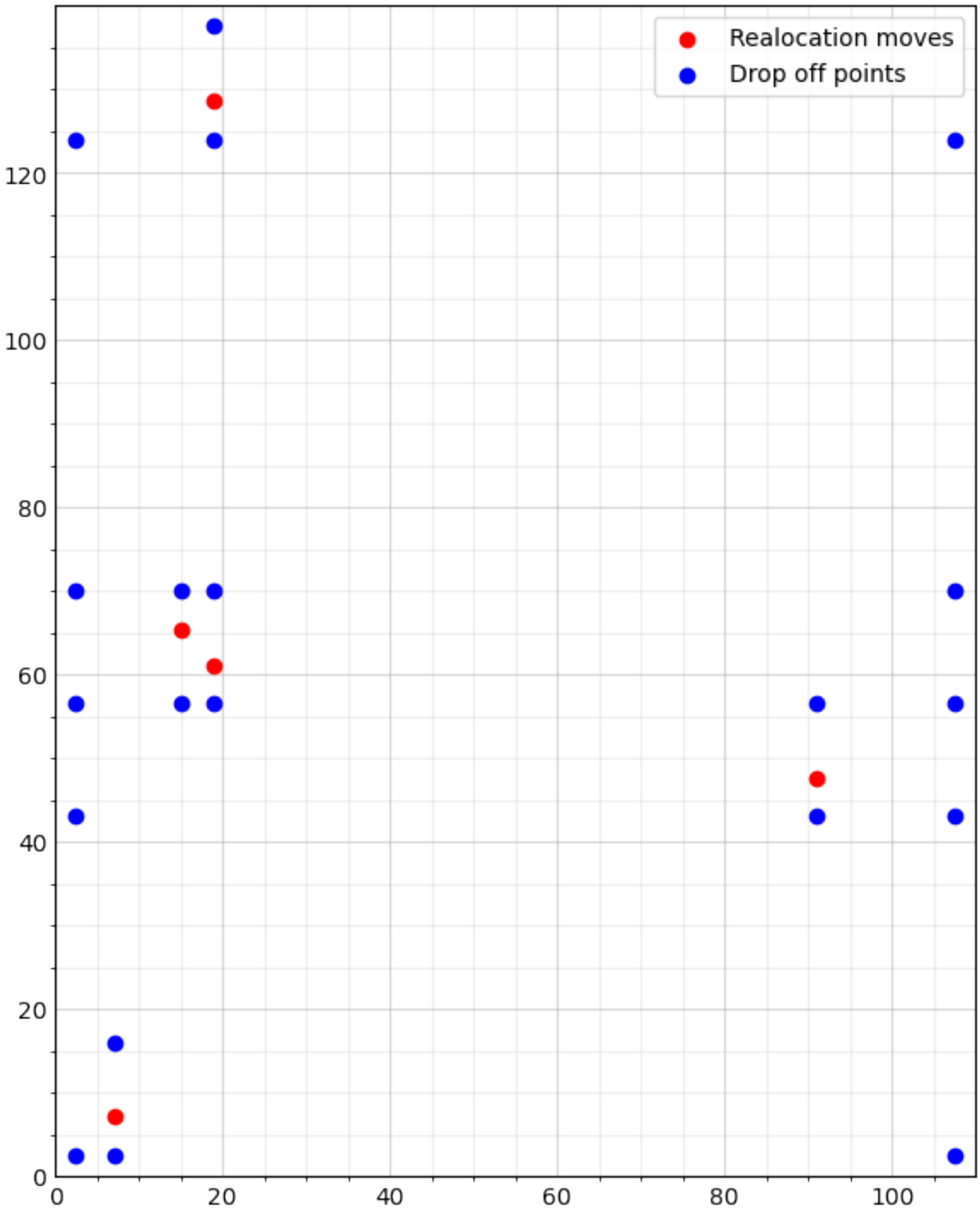
In [1]:
```
from Point import Point
from Trips import Trips
from Simulation import Simulation
from Solver import Solver
import time
import matplotlib.pyplot as plt
```

Single simulation

Preparate Data

In [2]:
```
n,m,ks,kr,kn,T_start,c=Simulation.get_simulation_number(0)
J,D=Simulation.initialize_map(n)
Simulation.plot_map(J,D,[],figsize=(5.5,7))
```



Solution of problem using Gurobi

In [3]:
```
start_time = time.time()
trips_problem=Solver.trptr_problem(n,m,J,D,Point(0,0),Point(0,0),time_lim
execution_time_problem=time.time()-start_time
```

```
Set parameter Username
Academic license - for non-commercial use only - expires 2023-10-29
Set parameter TimeLimit to value 300
```

```
Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (mac64[x86])
Thread count: 4 physical cores, 8 logical processors, using up to 8 threa
ds
Optimize a model with 2827 rows, 312 columns and 12122 nonzeros
Model fingerprint: 0x75f780ef
Variable types: 42 continuous, 270 integer (270 binary)
Coefficient statistics:
  Matrix range     [8e-01, 1e+02]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 2e+02]
Found heuristic solution: objective 125.1364241
Presolve removed 6 rows and 6 columns
Presolve time: 0.02s
Presolved: 2821 rows, 306 columns, 12082 nonzeros
Variable types: 42 continuous, 264 integer (264 binary)

Root relaxation: objective 3.445103e+01, 44 iterations, 0.01 seconds (0.0
0 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node T
ime

     0     0   34.45103    0   13  125.13642   34.45103  72.5%     -    0
s
H    0     0                      93.8620563   34.45103  63.3%     -    0
s
     0     0   34.45103    0   15   93.86206   34.45103  63.3%     -    0
s
     0     0   34.45103    0   18   93.86206   34.45103  63.3%     -    0
s
     0     0   34.45103    0   20   93.86206   34.45103  63.3%     -    0
s
     0     0   34.45103    0   22   93.86206   34.45103  63.3%     -    0
s
     0     0   34.45103    0   20   93.86206   34.45103  63.3%     -    0
s
     0     0   34.45103    0   15   93.86206   34.45103  63.3%     -    0
s
     0     0   34.45103    0   14   93.86206   34.45103  63.3%     -    0
s
     0     0   34.45103    0   13   93.86206   34.45103  63.3%     -    0
s
     0     0   34.45103    0   13   93.86206   34.45103  63.3%     -    0
s
     0     2   34.45103    0   13   93.86206   34.45103  63.3%     -    0
s
H  527   459                      93.3464688   34.45103  63.1%   7.7    0
s
H  706   484                      91.0131355   34.45103  62.1%   6.9    0
s
H 1001   715                      90.5131355   34.45103  61.9%   6.5    0
s
H 1132   690                      88.5131355   36.91613  58.3%   6.0    2
s
  1161   709   59.50000   38   33   88.51314   40.33900  54.4%   5.9    5
```

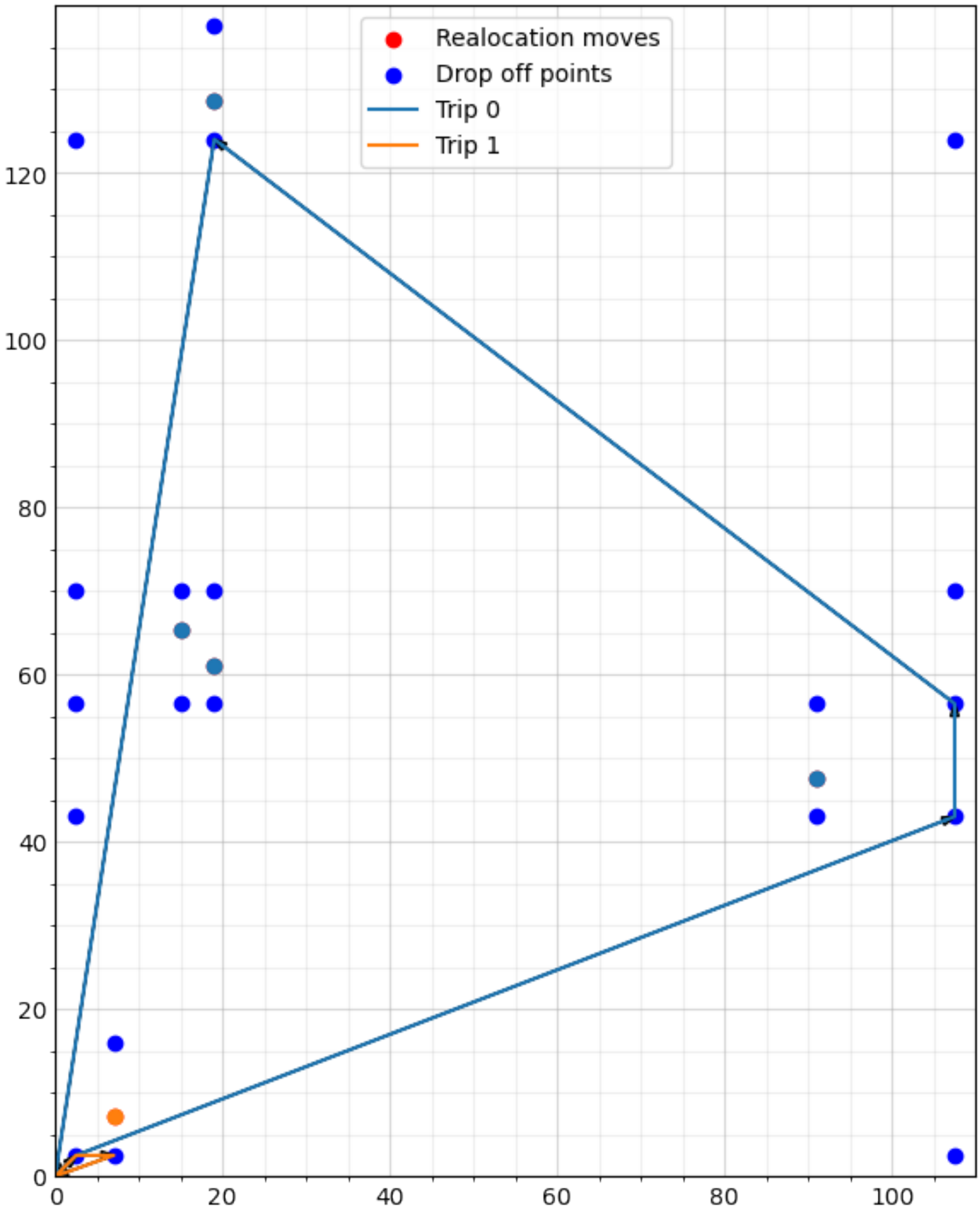
```
s
  2205  1048 infeasible   58          88.51314   42.21873  52.3%  14.6   10
s
 13841  6322     cutoff   50          88.51314   44.28436  50.0%  10.3   15
s
 25664 10713   71.40573   58   18    88.51314   46.36770  47.6%  10.2   20
s
 30999 12697   74.06593   46   13    88.51314   46.53436  47.4%   9.9   25
s
H31017 12073                        88.5131355   46.53436  47.4%   9.9   29
s
 31020 12075   55.33333   40   33    88.51314   46.53436  47.4%   9.9   30
s
H31028 11475                        86.0131355   46.53436  45.9%   9.9   32
s
 31042 11485   86.01314   69   36    86.01314   46.53436  45.9%   9.8   35
s
H31044 10911                        86.0131352   46.53436  45.9%   9.8   35
s
H31051 10369                        86.0131352   46.53436  45.9%   9.8   37
s
 31061 10375   58.59817   62   52    86.01314   46.53436  45.9%   9.8   40
s
 31080 10388   75.76881   54   35    86.01314   46.53436  45.9%   9.8   45
s
 31100 10401   61.26996   64   37    86.01314   46.53436  45.9%   9.8   50
s
 31116 10412   72.68480   72   38    86.01314   46.53436  45.9%   9.8   55
s
 31134 10424   53.26438   83   34    86.01314   46.53436  45.9%   9.8   60
s
H31171  9924                        86.0131352   46.53436  45.9%   9.8   64
s
 31184  9935   46.53436   34   31    86.01314   46.53436  45.9%  10.8   65
s
 34634 10917   73.88248   64   14    86.01314   46.53436  45.9%  11.4   70
s
 43966 12751 infeasible   60          86.01314   46.53436  45.9%  12.1   75
s
 52002 13408   56.78436   63   17    86.01314   46.53436  45.9%  12.5   80
s
 62606 16009   73.48906   83   10    86.01314   46.53436  45.9%  12.3   85
s
 75151 21027 infeasible   75          86.01314   46.53436  45.9%  12.4   90
s
 93308 29512     cutoff   91          86.01314   46.53436  45.9%  12.1   95
s
 107990 34421 infeasible   66          86.01314   46.53436  45.9%  12.2   10
0s
 123899 39775   46.53436   69   20    86.01314   46.53436  45.9%  12.1   10
5s
 142195 46014   63.68150   71   14    86.01314   46.53436  45.9%  11.8   11
0s
 163113 52908   47.51552   49   19    86.01314   46.53436  45.9%  11.3   11
5s
 181786 58257   49.18048   76   16    86.01314   46.69933  45.7%  11.1   12
0s
```

```
 198078 63474   49.51552   80   15   86.01314   47.03436  45.3%  11.0  12
5s
 213091 64983     cutoff   74        86.01314   47.85631  44.4%  11.1  13
0s
 230018 68332 infeasible   78        86.01314   48.61770  43.5%  11.1  13
5s
 249845 72145   51.41621   87   17   86.01314   49.28436  42.7%  10.9  14
0s
 269567 77310 infeasible   56        86.01314   49.75000  42.2%  10.8  14
5s
 293893 83540 infeasible   75        86.01314   50.68816  41.1%  10.6  15
0s
 313994 87389   84.59817   74   15   86.01314   51.74955  39.8%  10.5  15
5s
 335912 91949   71.32239   94    6   86.01314   53.21399  38.1%  10.4  16
0s
 352406 93840     cutoff   70        86.01314   54.19009  37.0%  10.4  16
5s
 373560 97120   75.59817   66   11   86.01314   55.25000  35.8%  10.3  17
0s
 392249 100499   80.90573   52   11   86.01314   56.00000  34.9%  10.3   1
75s
 413070 104114     cutoff   51        86.01314   56.75000  34.0%  10.2   1
80s
 430906 107526 infeasible   82        86.01314   56.78436  34.0%  10.2   1
85s
 449158 110677   57.04732   87    6   86.01314   57.03436  33.7%  10.2   1
90s
 469788 112920     cutoff   98        86.01314   57.76552  32.8%  10.2   1
95s
 485110 112677   59.33562   68   17   86.01314   58.28436  32.2%  10.2   2
00s
 502250 113671 infeasible   74        86.01314   58.93218  31.5%  10.3   2
05s
 520438 114850 infeasible   70        86.01314   59.03436  31.4%  10.3   2
10s
 533638 114100 infeasible   84        86.01314   59.33562  31.0%  10.3   2
15s
H545889 114142                       86.0131352   59.51552  30.8%  10.4   2
18s
 549962 115584   67.93961   64   12   86.01314   59.61770  30.7%  10.4   2
20s
 571804 119083   59.91664   99   13   86.01314   59.91664  30.3%  10.3   2
25s
 589122 118734     cutoff   91        86.01314   60.36770  29.8%  10.3   2
30s
 608498 119936 infeasible   93        86.01314   60.60631  29.5%  10.3   2
35s
 626925 118961   60.88254   68   11   86.01314   60.88254  29.2%  10.2   2
40s
 649696 119806     cutoff   98        86.01314   60.90430  29.2%  10.1   2
45s
 667846 119981   74.76484   70    8   86.01314   61.34817  28.7%  10.1   2
50s
 684151 119025   61.62895   79   14   86.01314   61.59885  28.4%  10.1   2
55s
 701924 120571     cutoff   86        86.01314   61.75000  28.2%  10.1   2
```

```
60s
 720153 119193 infeasible   82         86.01314   62.26506  27.6%  10.1  2
65s
 736620 118751 infeasible   92         86.01314   62.68964  27.1%  10.1  2
70s
 751860 118127 infeasible   72         86.01314   62.85631  26.9%  10.2  2
75s
 769604 116792   63.36369   71   11    86.01314   63.36369  26.3%  10.2  2
80s
 786833 115240 infeasible   67         86.01314   63.90573  25.7%  10.2  2
85s
 804986 113898     cutoff   76         86.01314   64.11770  25.5%  10.2  2
90s
 821857 112194     cutoff   87         86.01314   64.45448  25.1%  10.2  2
95s
 839435 109982   70.01203   51   15    86.01314   65.10631  24.3%  10.2  3
00s

Cutting planes:
  Cover: 1
  MIR: 2
  Flow cover: 18
  Relax-and-lift: 38

Explored 840250 nodes (8570561 simplex iterations) in 300.02 seconds (236
.45 work units)
Thread count was 8 (of 8 available processors)

Solution count 9: 86.0131 86.0131 86.0131 ... 125.136

Time limit reached
Best objective 8.601313519012e+01, best bound 6.511769793705e+01, gap 24.
2933%
```

In [4]:

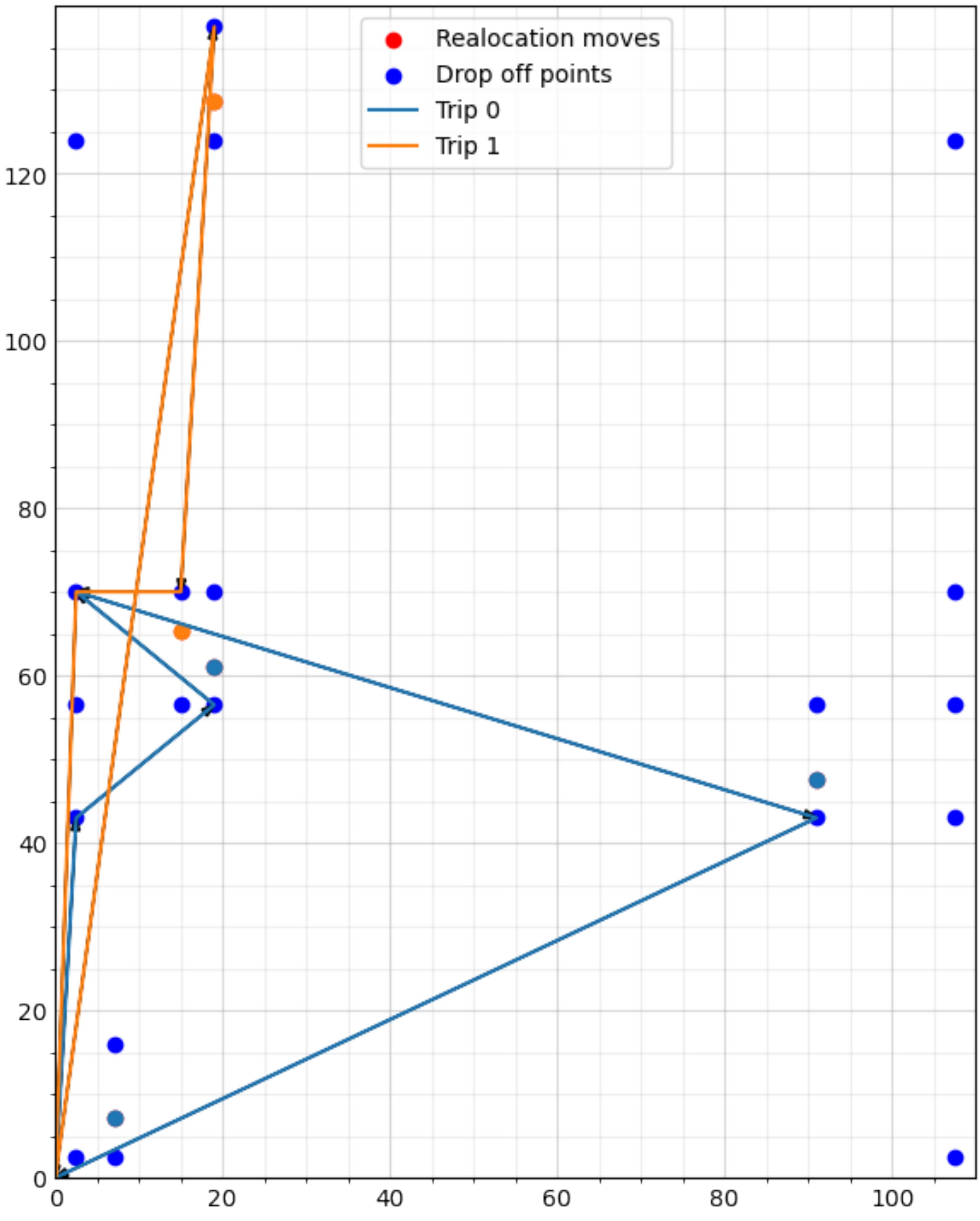
```
Simulation.plot_map(J,D,trips_problem,figsize=(5.5,7))
```



Solution of the problem using the heuristics

In [5]:
```
start_time = time.time()
trips=Solver.sa_approach(n, m, ks, kr, kn, T_start, c, J, D, Point(0,0),
execution_sa=time.time()-start_time
```

In [6]:
```
Simulation.plot_map(J,D,trips,figsize=(5.5,7))
```

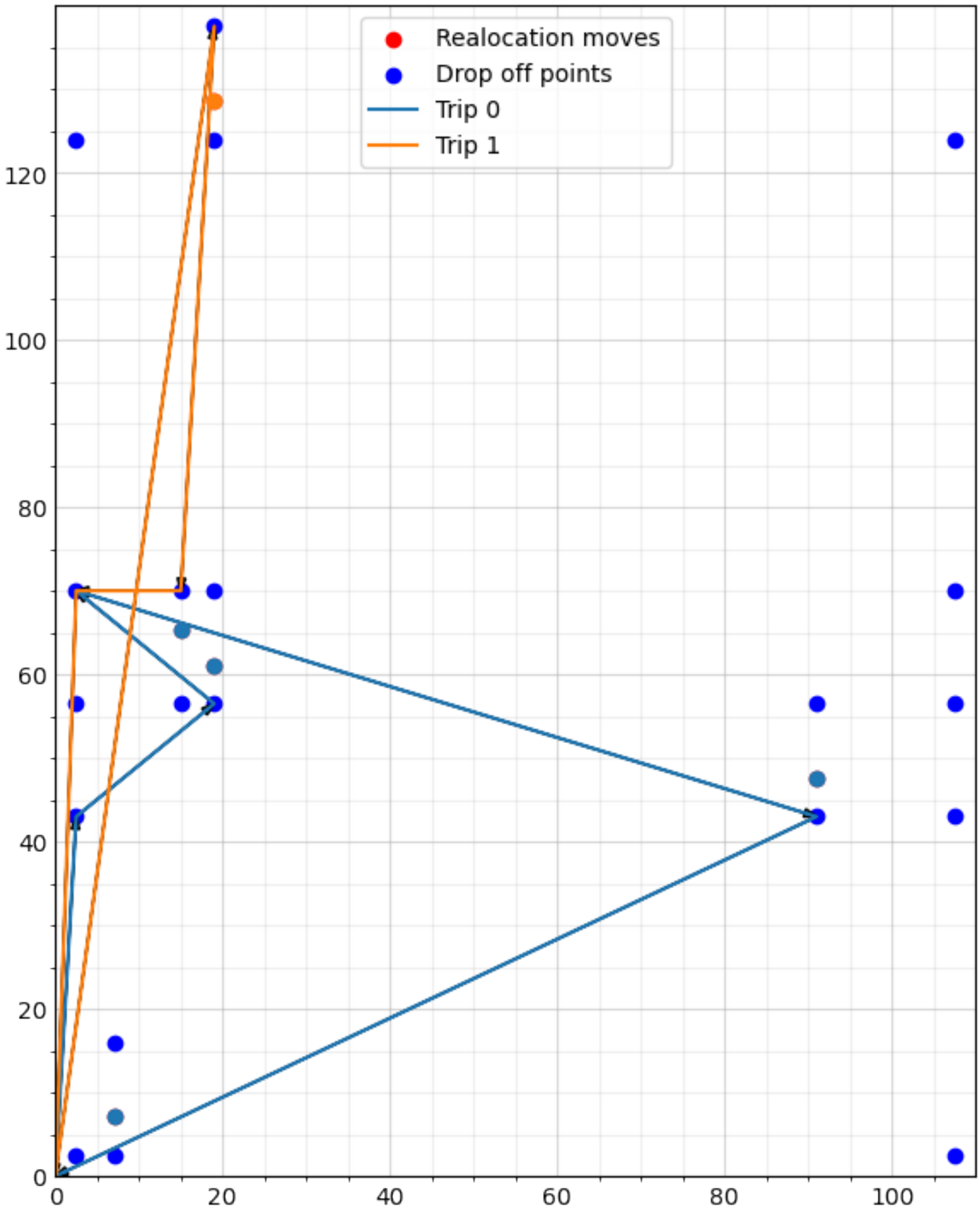


Select-and-assign matheuristic (SAM)

In [7]:
```
start_time = time.time()
trips_sam=Solver.sam_matheuristic(n, m, J, D, trips,time_limit=60)
execution_sam=time.time()-start_time
```

```
Set parameter TimeLimit to value 60
Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (mac64[x86])
Thread count: 4 physical cores, 8 logical processors, using up to 8 threa
ds
Optimize a model with 17 rows, 14 columns and 38 nonzeros
Model fingerprint: 0xd6d544f5
Variable types: 2 continuous, 12 integer (12 binary)
Coefficient statistics:
  Matrix range     [1e+00, 7e+01]
  Objective range  [1e+00, 6e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 184.7674177
Presolve removed 17 rows and 14 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.01 seconds (0.00 work units)
Thread count was 1 (of 8 available processors)

Solution count 2: 122.932 184.767

Optimal solution found (tolerance 1.00e-04)
Best objective 1.229315047962e+02, best bound 1.229315047962e+02, gap 0.0
000%

Solution
Binary variables: 1, if relocation move j in J is executed on taxi trip i
in I;0, otherwise
realocation move 0 is executed on taxi trip 0
realocation move 1 is executed on taxi trip 0
realocation move 2 is executed on taxi trip 0
realocation move 4 is executed on taxi trip 0
realocation move 3 is executed on taxi trip 1
Binary variables: 1, if taxi trip i in I is selected from the pool; 0,oth
erwise
taxi trip 0 is selected from the pool
taxi trip 1 is selected from the pool
```

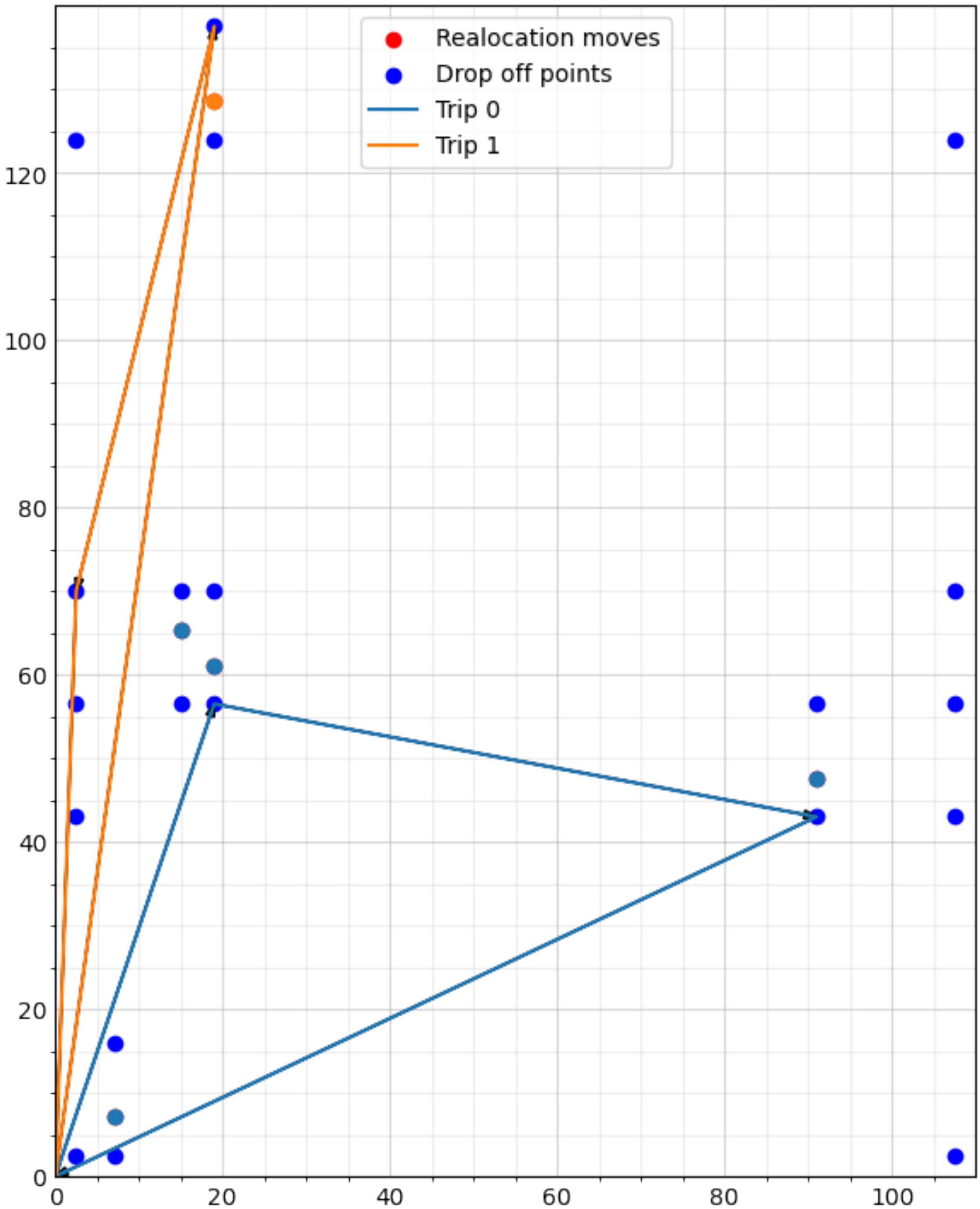
In [8]:
```
Simulation.plot_map(J,D,trips_sam,figsize=(5.5,7))
```



SAM-Local search

In [9]:
```
start_time = time.time()
trips_sam_ls=Solver.local_search(n,m,J, D, 100000,trips_sam)
execution_sam_ls=time.time()-start_time
```

In [10]:
```
Simulation.plot_map(J,D,trips_sam_ls,figsize=(5.5,7))
```



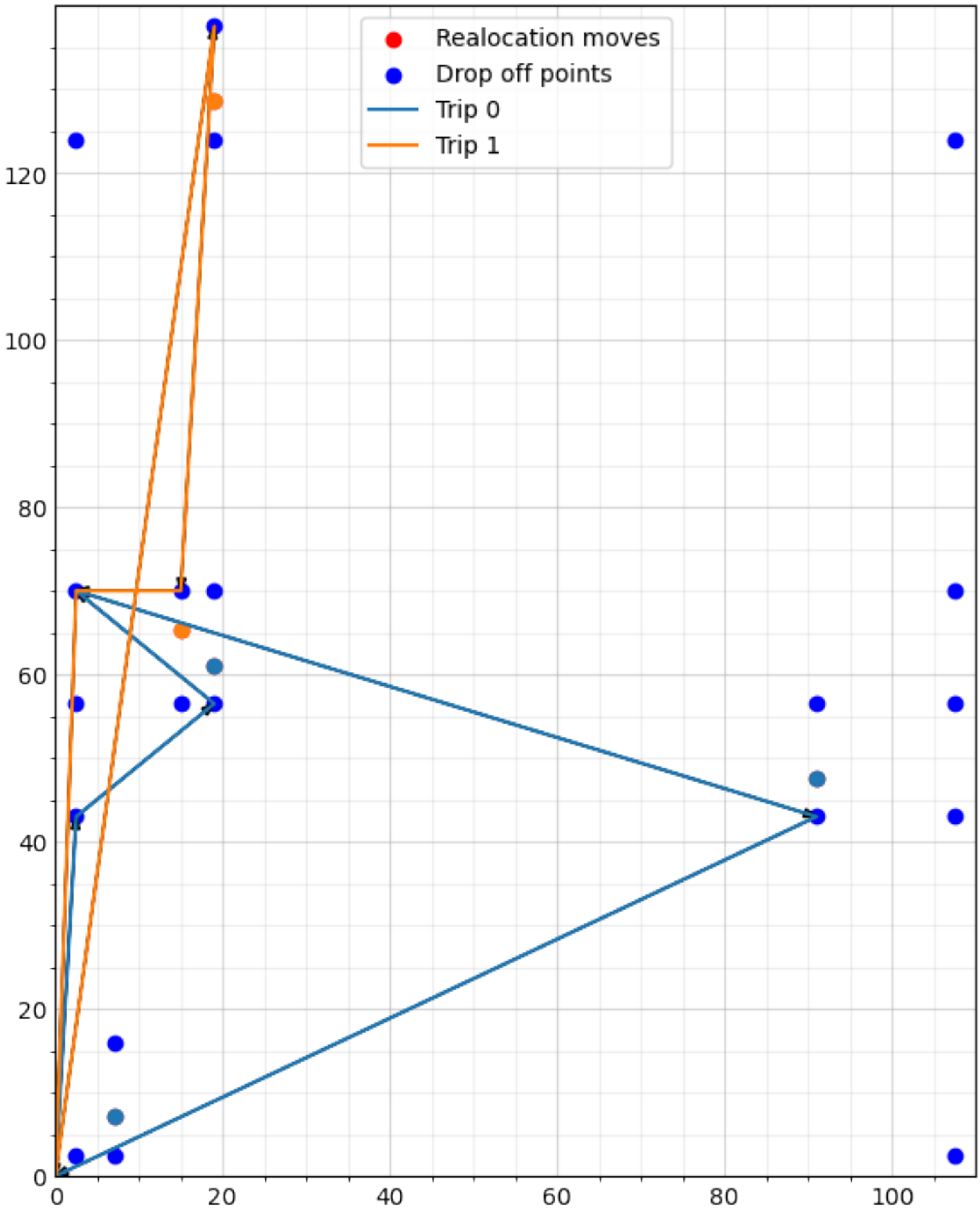
Select matheuristic (SM)

In [11]:
```
start_time = time.time()
trips_sm=Solver.sm_matheuristic(J,D,trips)
execution_sm=time.time()-start_time
```

```
Set parameter TimeLimit to value 60
Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (mac64[x86])
Thread count: 4 physical cores, 8 logical processors, using up to 8 threa
ds
Optimize a model with 5 rows, 2 columns and 5 nonzeros
Model fingerprint: 0xe32e69ba
Variable types: 0 continuous, 2 integer (2 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [6e+01, 7e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 125.4880096
Presolve removed 5 rows and 2 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.01 seconds (0.00 work units)
Thread count was 1 (of 8 available processors)

Solution count 1: 125.488

Optimal solution found (tolerance 1.00e-04)
Best objective 1.254880095923e+02, best bound 1.254880095923e+02, gap 0.0
000%
```

In [12]:

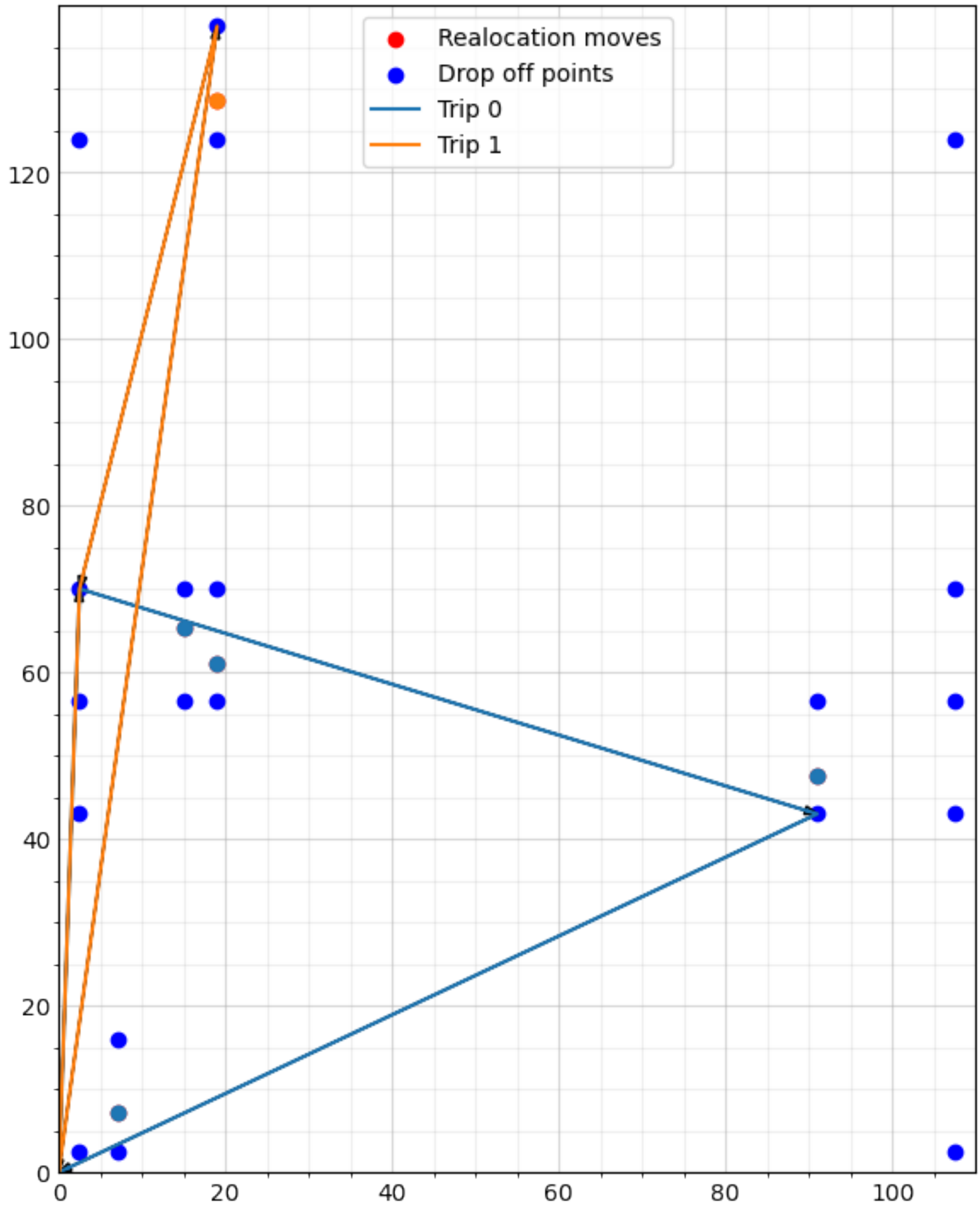
```
Simulation.plot_map(J,D,trips_sm,figsize=(5.5,7))
```



SM -Local search

In [13]:
```
start_time = time.time()
trips_sm_ls=Solver.local_search(n,m,J, D, 100000,trips_sm)
execution_sm_ls=time.time()-start_time
```

In [14]:
```
Simulation.plot_map(J,D,trips_sm_ls,figsize=(5.5,7))
```



Comparison

comparison of travel times obtained

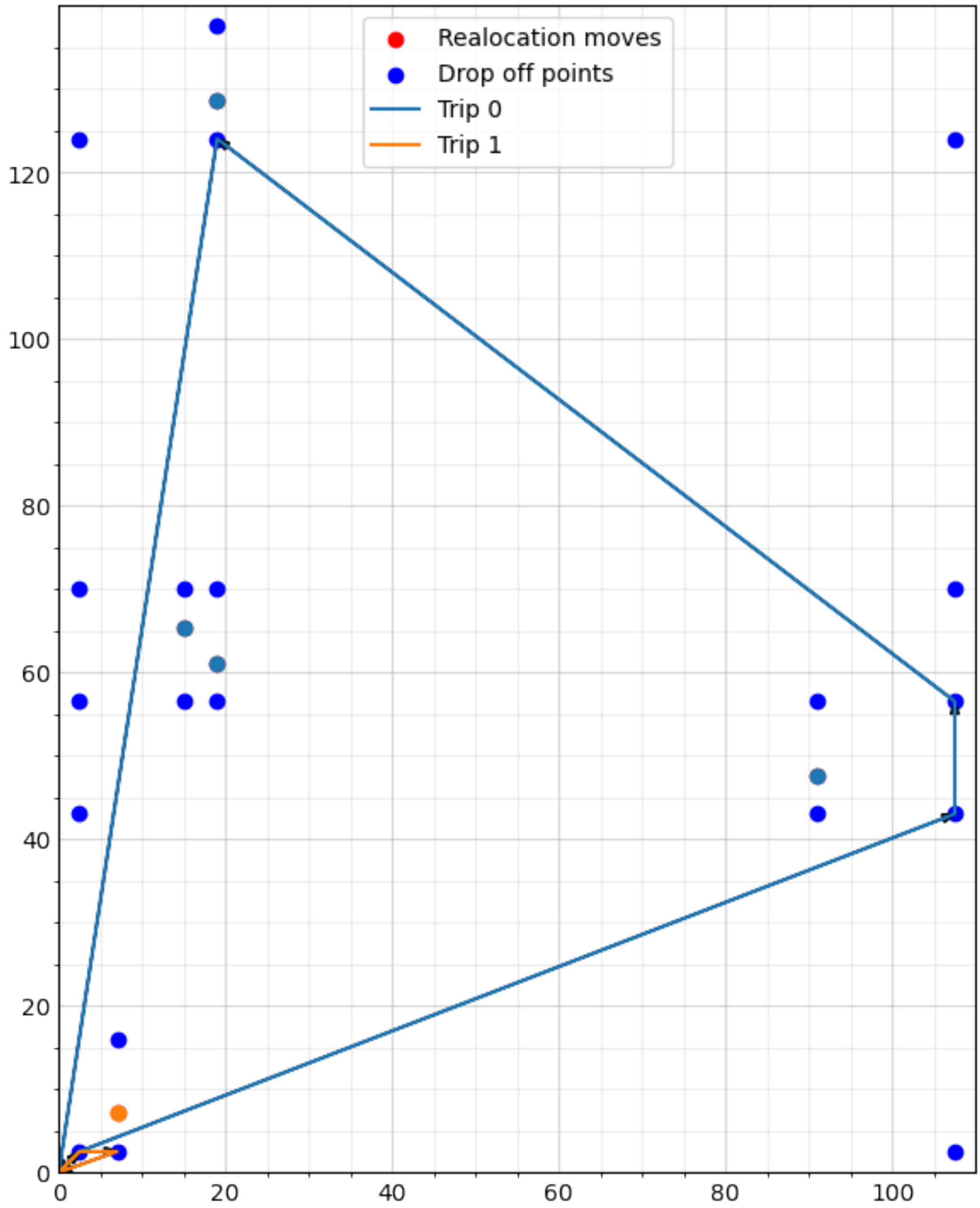
In [15]:
```
travel_time_problem=Trips.get_total_duration(trips_problem)
travel_time_sam=Trips.get_total_duration(trips_sam_ls)
travel_time_sm=Trips.get_total_duration(trips_sm_ls)
print(f"travel time problem ={travel_time_problem}")
print(f"travel time sam ={travel_time_sam}")
print(f"travel time sm ={travel_time_sm}")
```

```
travel time problem =81.0296762589928
travel time sam =113.33723021582735
travel time sm =116.85940870930014
```

Graphical comparison

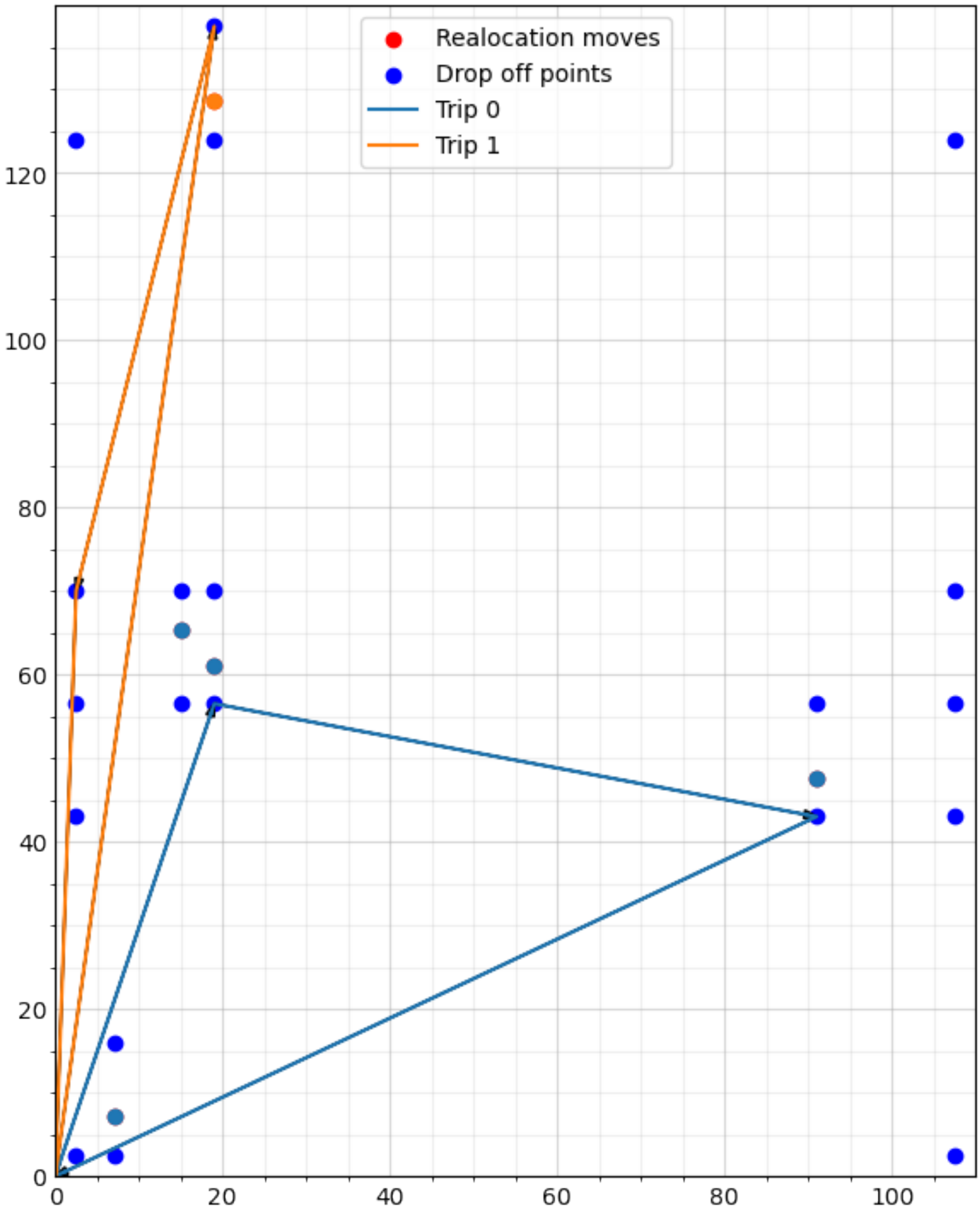
Travel map of problem solved with gurobi

In [16]:
```
Simulation.plot_map(J,D,trips_problem,figsize=(5.5,7))
```



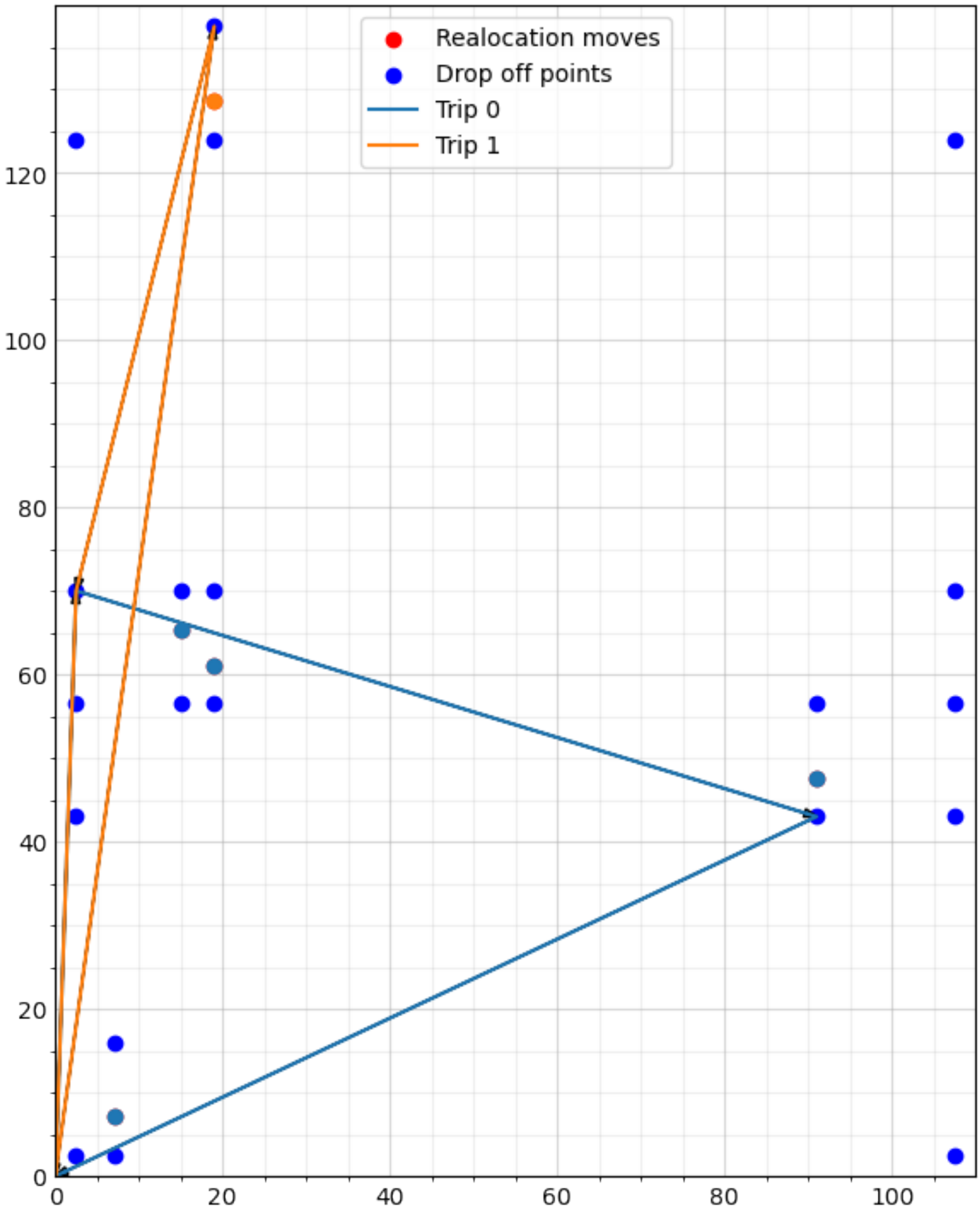
Travel map of problem solved with SAM

In [17]:
```
Simulation.plot_map(J,D,trips_sam_ls,figsize=(5.5,7))
```



Travel map of problem solved with SM

In [18]:
```
Simulation.plot_map(J,D,trips_sm_ls,figsize=(5.5,7))
```

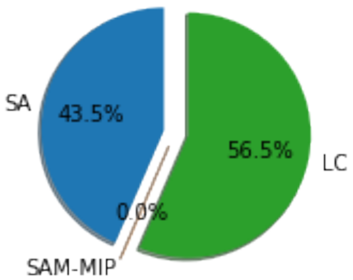


Running time comparison

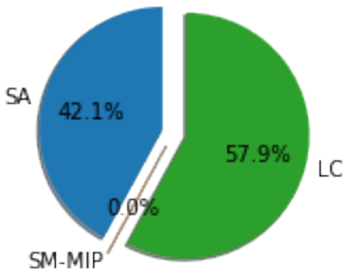
In [19]:
```
total_time_sam=execution_sa+execution_sam+execution_sam_ls
total_time_sm=execution_sa+execution_sm+execution_sm_ls
print(f'running time problem solved with Gurobi ={execution_time_problem}
print(f'running time problem using SMA heuristic ={total_time_sam}')
print(f'running time problem using SM heuristic ={total_time_sm} \n')

print("Time spent by heuristics SMA in phases:")
print(f"tima SA={execution_sa} in percentage {execution_sa/total_time_sam
print(f"tima SAM-MIP={execution_sam} in percentage {execution_sam/total_t
print(f"tima LC={execution_sam_ls} in percentage {execution_sam_ls/total_

print("")
print("Time spent by heuristics SM in phases:")
print(f"tima SA={execution_sa} in percentage {execution_sa/total_time_sm*
print(f"tima SM-MIP={execution_sm} in percentage {execution_sm/total_time
print(f"tima LC={execution_sm_ls} in percentage {execution_sm_ls/total_ti
```

```
running time problem solved with Gurobi =300.3434238433838
running time problem using SMA heuristic =69.64339327812195
running time problem using SM heuristic =72.0274703502655

Time spent by heuristics SMA in phases:
tima SA=30.29239511489868 in percentage 43.49643762176484%
tima SAM-MIP=0.017016887664794922 in percentage 0.024434317260845866%
tima LC=39.33398127555847 in percentage 56.47912806097432%

Time spent by heuristics SM in phases:
tima SA=30.29239511489868 in percentage 42.0567249794953%
tima SM-MIP=0.01640915870666504 in percentage 0.022781806200097892%
tima LC=41.718666076660156 in percentage 57.920493214303725%
```

In [20]:
```
# Pie chart, where the slices will be ordered and plotted counter-clockwi
labels = 'SA', 'SAM-MIP', 'LC'
sizes = [execution_sa/total_time_sam*100, execution_sam/total_time_sam*10
explode = (0.1, 0.1, 0.1)

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a ci

plt.show()
```



In [21]:
```
# Pie chart, where the slices will be ordered and plotted counter-clockwi
labels = 'SA', 'SM-MIP', 'LC'
sizes = [execution_sa/total_time_sm*100, execution_sm/total_time_sm*100,
explode = (0.1, 0.1, 0.1)

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a ci

plt.show()
```



In [ ]: