Workgroup #6

A solver for VRPTW and Fixed Fleet size VRPTW

Members

Ovidiu Birgu	231567	Francesco Maggiolino	193509
Sabrina Camurati	195318	Martino Mensio	232297
Giacomo Gustavino	225266	Tommaso Pifferi	191636

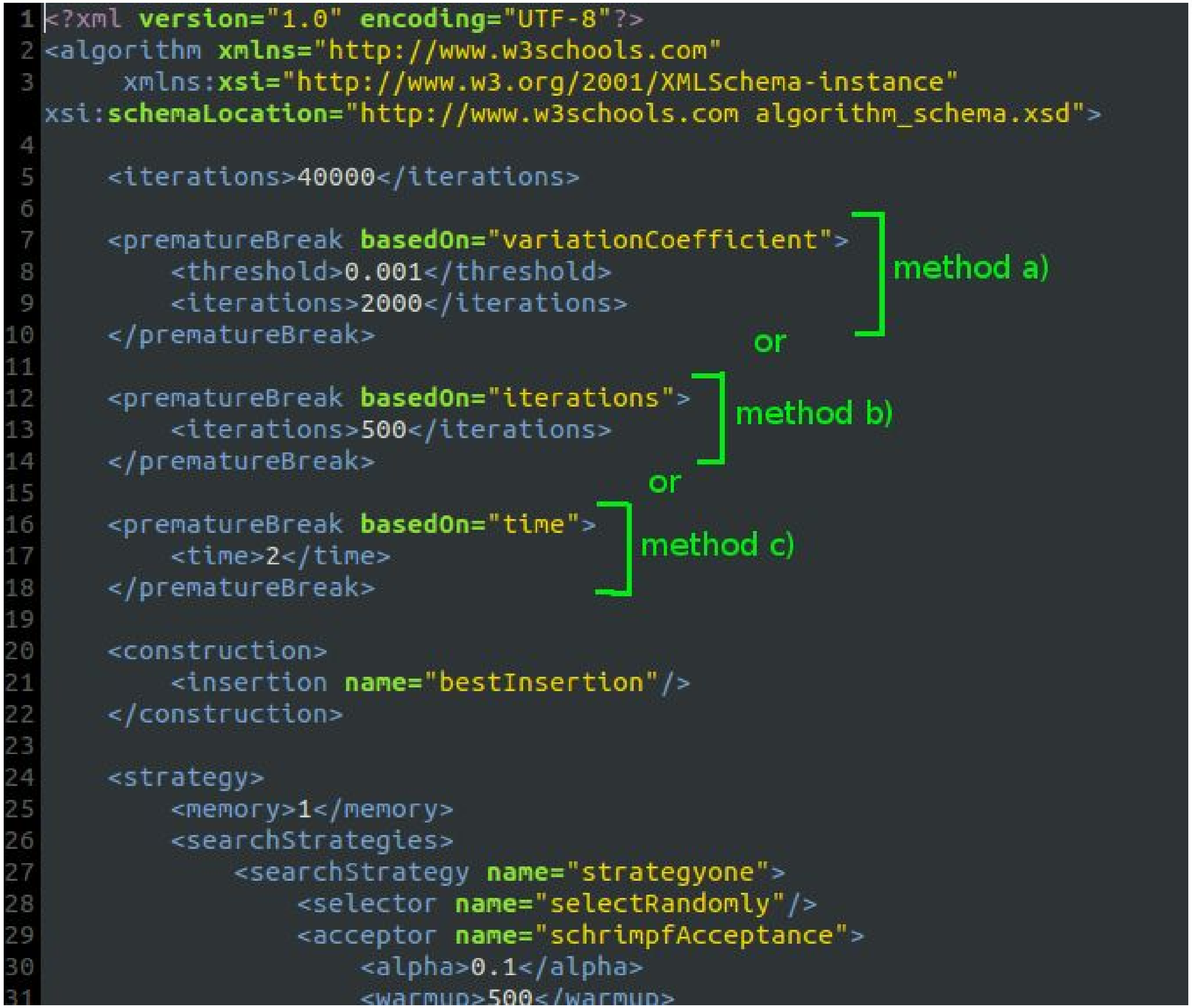
www.orgroup.polito.it

Assignment 1

How we started

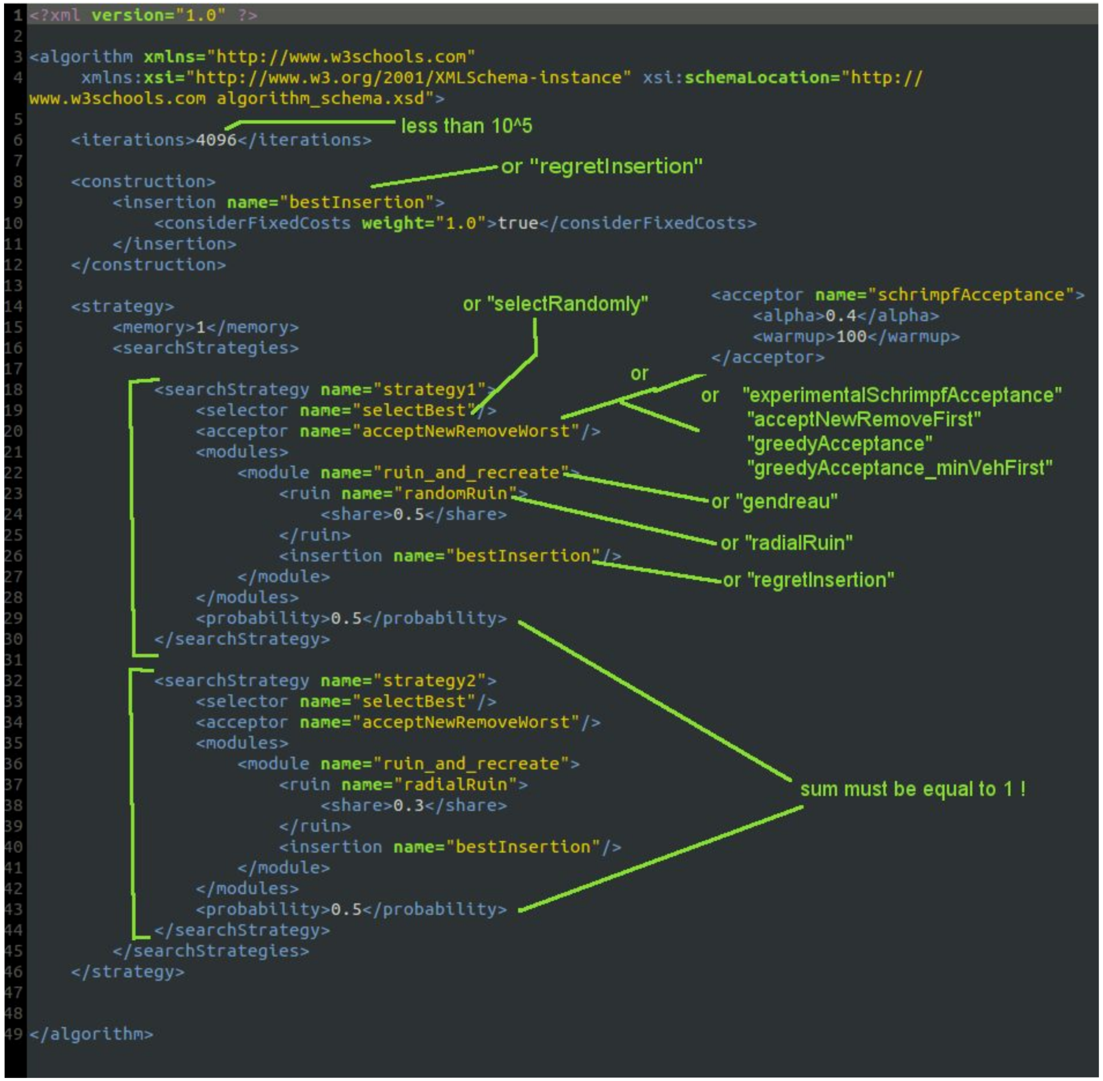
- ➔The metaheuristic is already given
- ➔How to tweak the parameters inside the xml file?
- ➔Little to no documentation, so
  - ◆main/resources has a bunch of XML config files
  - ◆an .xsd file that defines the general schema
  - ◆java examples based on the above custom configurations

Premature Break



```
<?xml version="1.0" encoding="UTF-8"?>
<algorithm xmlns="http://www.w3schools.com"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com algorithm_schema.xsd">

    <iterations>40000</iterations>

    <prematureBreak basedOn="variationCoefficient">
        <threshold>0.001</threshold>
        <iterations>2000</iterations>
    </prematureBreak>

    <prematureBreak basedOn="iterations">
        <iterations>500</iterations>
    </prematureBreak>

    <prematureBreak basedOn="time">
        <time>2</time>
    </prematureBreak>

    <construction>
        <insertion name="bestInsertion"/>
    </construction>

    <strategy>
        <memory>1</memory>
        <searchStrategies>
            <searchStrategy name="strategyone">
                <selector name="selectRandomly"/>
                <acceptor name="schrimpfAcceptance">
                    <alpha>0.1</alpha>
                    <warmup>500</warmup>
```

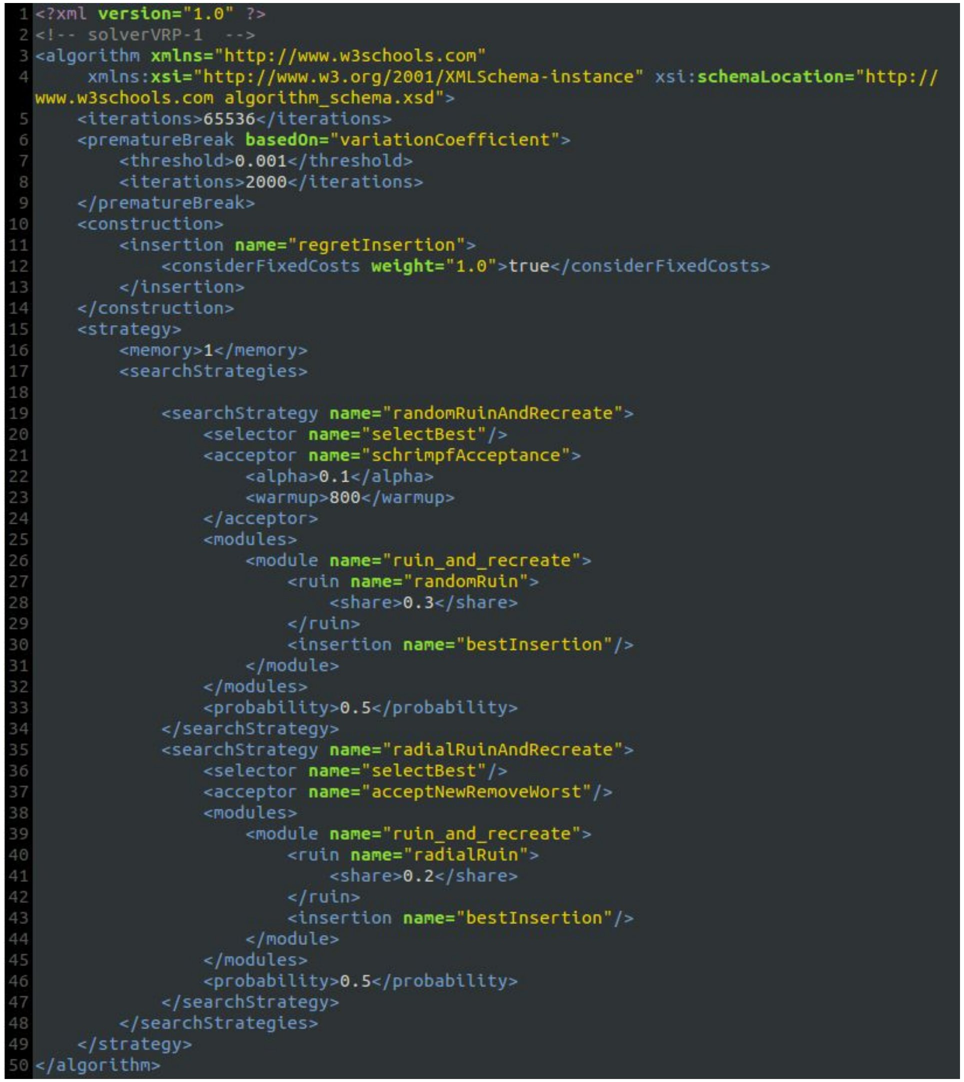
Various parameters



```xml
<?xml version="1.0" ?>

<algorithm xmlns="http://www.w3schools.com"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
www.w3schools.com algorithm_schema.xsd">

    <iterations>4096</iterations>

    <construction>
        <insertion name="bestInsertion">
            <considerFixedCosts weight="1.0">true</considerFixedCosts>
        </insertion>
    </construction>

    <strategy>
        <memory>1</memory>
        <searchStrategies>

            <searchStrategy name="strategy1">
                <selector name="selectBest"/>
                <acceptor name="acceptNewRemoveWorst"/>
                <modules>
                    <module name="ruin_and_recreate">
                        <ruin name="randomRuin">
                            <share>0.5</share>
                        </ruin>
                        <insertion name="bestInsertion"/>
                    </module>
                </modules>
                <probability>0.5</probability>
            </searchStrategy>

            <searchStrategy name="strategy2">
                <selector name="selectBest"/>
                <acceptor name="acceptNewRemoveWorst"/>
                <modules>
                    <module name="ruin_and_recreate">
                        <ruin name="radialRuin">
                            <share>0.3</share>
                        </ruin>
                        <insertion name="bestInsertion"/>
                    </module>
                </modules>
                <probability>0.5</probability>
            </searchStrategy>
        </searchStrategies>
    </strategy>


</algorithm>
```

```xml
<acceptor name="schrimpfAcceptance">
    <alpha>0.4</alpha>
    <warmup>100</warmup>
</acceptor>
```

Our config



```xml
<?xml version="1.0" ?>
<!-- solverVRP-1  -->
<algorithm xmlns="http://www.w3schools.com"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.w3schools.com algorithm_schema.xsd">
    <iterations>65536</iterations>
    <prematureBreak basedOn="variationCoefficient">
        <threshold>0.001</threshold>
        <iterations>2000</iterations>
    </prematureBreak>
    <construction>
        <insertion name="regretInsertion">
            <considerFixedCosts weight="1.0">true</considerFixedCosts>
        </insertion>
    </construction>
    <strategy>
        <memory>1</memory>
        <searchStrategies>

            <searchStrategy name="randomRuinAndRecreate">
                <selector name="selectBest"/>
                <acceptor name="schrimpfAcceptance">
                    <alpha>0.1</alpha>
                    <warmup>800</warmup>
                </acceptor>
                <modules>
                    <module name="ruin_and_recreate">
                        <ruin name="randomRuin">
                            <share>0.3</share>
                        </ruin>
                        <insertion name="bestInsertion"/>
                    </module>
                </modules>
                <probability>0.5</probability>
            </searchStrategy>
            <searchStrategy name="radialRuinAndRecreate">
                <selector name="selectBest"/>
                <acceptor name="acceptNewRemoveWorst"/>
                <modules>
                    <module name="ruin_and_recreate">
                        <ruin name="radialRuin">
                            <share>0.2</share>
                        </ruin>
                        <insertion name="bestInsertion"/>
                    </module>
                </modules>
                <probability>0.5</probability>
            </searchStrategy>
        </searchStrategies>
    </strategy>
</algorithm>
```

Details

- Main blocks
  - Number of iterations: 65536
    - We tried growing powers of 2, starting from 1024, until we exceeded the maximum time constraint.
    - 65536 was upper limit that we found in our tests, given the other two conditions: maximum number of threads and CPU power.

Details

- ➔Main blocks
  - ◆PrematureBreak
    - ●It breaks the run if after a given number of iterations the solution has not improved of at least the given amount
    - ●Very lax threshold (0.001) so it does not have side effects on the algorithm, but it just detects if the run is stalling.

Details

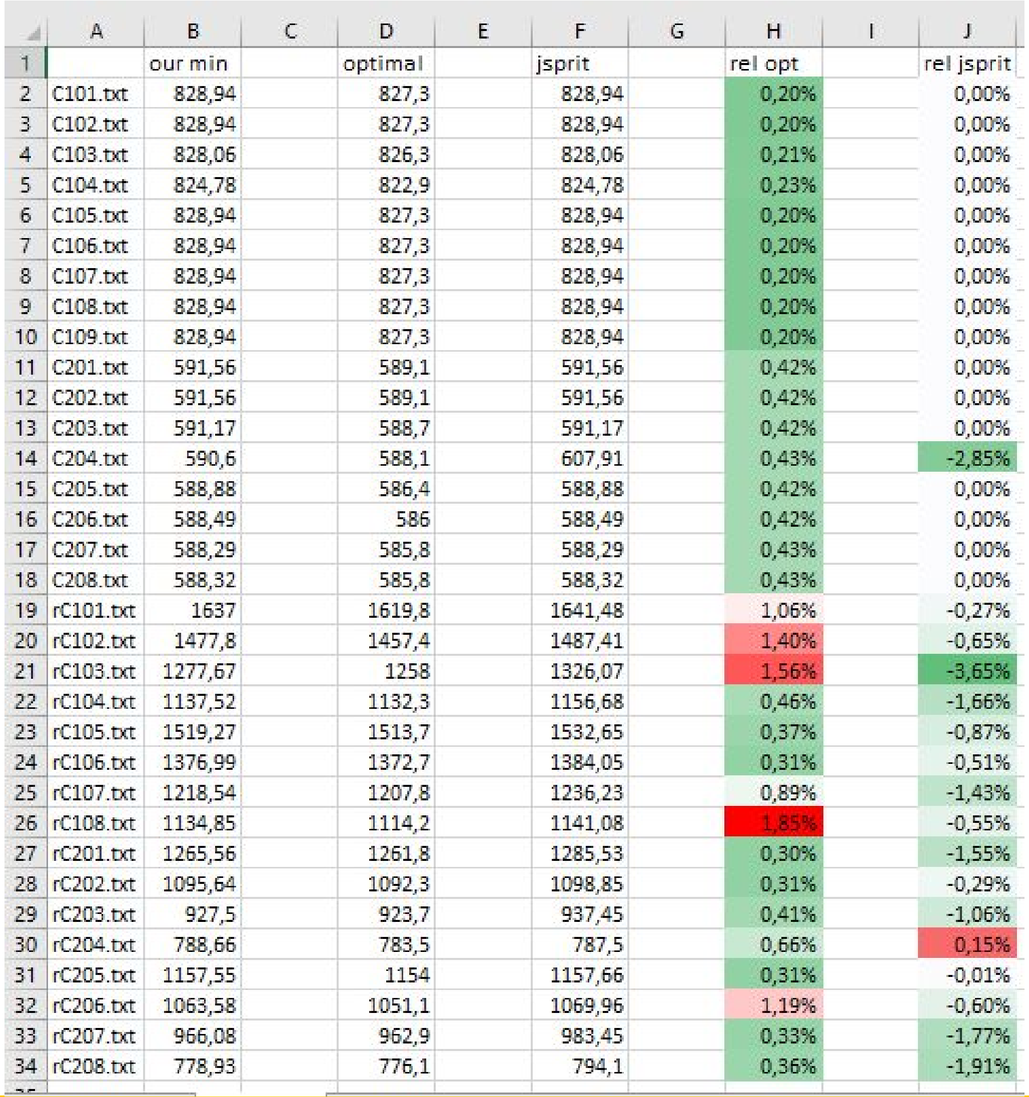
- ➔Main blocks
  - ◆Strategy 1: SchrimpfAcceptance
    - At the beginning it accepts also worse solutions over a certain threshold, then, over time it converges to a greedy approach
    - Suitable for problems with several local minima
    - Warmup is set to 800, a number that ensures a fairly deep exploration of the search space without taking too much from the execution time
    - Alpha 0.1 granted the best threshold through iterations
    - The 0.3 randomRuin additionally helped getting out of local minima

Details

➔Main blocks

- ◆Strategy 2: acceptNewRemoveWorst
  - ●In opposition with SchrimpfAcceptance it involves more of a greedy approach
  - ●The selectBest selector avoids taking a non optimal solution
  - ●The 0.2 radialRuin further focuses on local solutions
  - ●This strategy and the previous one are complementary, thus they are picked with a 50% chance each

Results



	A	B	C	D	E	F	G	H	I	J
1		our min		optimal		jsprit		rel opt		rel jsprit
2	C101.txt	828,94		827,3		828,94		0,20%		0,00%
3	C102.txt	828,94		827,3		828,94		0,20%		0,00%
4	C103.txt	828,06		826,3		828,06		0,21%		0,00%
5	C104.txt	824,78		822,9		824,78		0,23%		0,00%
6	C105.txt	828,94		827,3		828,94		0,20%		0,00%
7	C106.txt	828,94		827,3		828,94		0,20%		0,00%
8	C107.txt	828,94		827,3		828,94		0,20%		0,00%
9	C108.txt	828,94		827,3		828,94		0,20%		0,00%
10	C109.txt	828,94		827,3		828,94		0,20%		0,00%
11	C201.txt	591,56		589,1		591,56		0,42%		0,00%
12	C202.txt	591,56		589,1		591,56		0,42%		0,00%
13	C203.txt	591,17		588,7		591,17		0,42%		0,00%
14	C204.txt	590,6		588,1		607,91		0,43%		-2,85%
15	C205.txt	588,88		586,4		588,88		0,42%		0,00%
16	C206.txt	588,49		586		588,49		0,42%		0,00%
17	C207.txt	588,29		585,8		588,29		0,43%		0,00%
18	C208.txt	588,32		585,8		588,32		0,43%		0,00%
19	rC101.txt	1637		1619,8		1641,48		1,06%		-0,27%
20	rC102.txt	1477,8		1457,4		1487,41		1,40%		-0,65%
21	rC103.txt	1277,67		1258		1326,07		1,56%		-3,65%
22	rC104.txt	1137,52		1132,3		1156,68		0,46%		-1,66%
23	rC105.txt	1519,27		1513,7		1532,65		0,37%		-0,87%
24	rC106.txt	1376,99		1372,7		1384,05		0,31%		-0,51%
25	rC107.txt	1218,54		1207,8		1236,23		0,89%		-1,43%
26	rC108.txt	1134,85		1114,2		1141,08		1,85%		-0,55%
27	rC201.txt	1265,56		1261,8		1285,53		0,30%		-1,55%
28	rC202.txt	1095,64		1092,3		1098,85		0,31%		-0,29%
29	rC203.txt	927,5		923,7		937,45		0,41%		-1,06%
30	rC204.txt	788,66		783,5		787,5		0,66%		0,15%
31	rC205.txt	1157,55		1154		1157,66		0,31%		-0,01%
32	rC206.txt	1063,58		1051,1		1069,96		1,19%		-0,60%
33	rC207.txt	966,08		962,9		983,45		0,33%		-1,77%
34	rC208.txt	778,93		776,1		794,1		0,36%		-1,91%

Final notes

- We eventually chose the Schrimpf Acceptance because this method offered better solutions than the others.
- The warmup number has been set to 800 in order to improve the search operation.
- We noticed that it is not efficient to set a warmup too high because otherwise the process would require too much time.

Assignment 2

Ideas

➔Main objectives

- ◆Setting the upper limit of the fleet size
- ◆Setting the lower limit of the fleet size
- ◆Implementing the hard and soft constraints
- ◆Have balanced solutions even when fleet size changes (increase of 10%, 20%, 30%)

Run the project

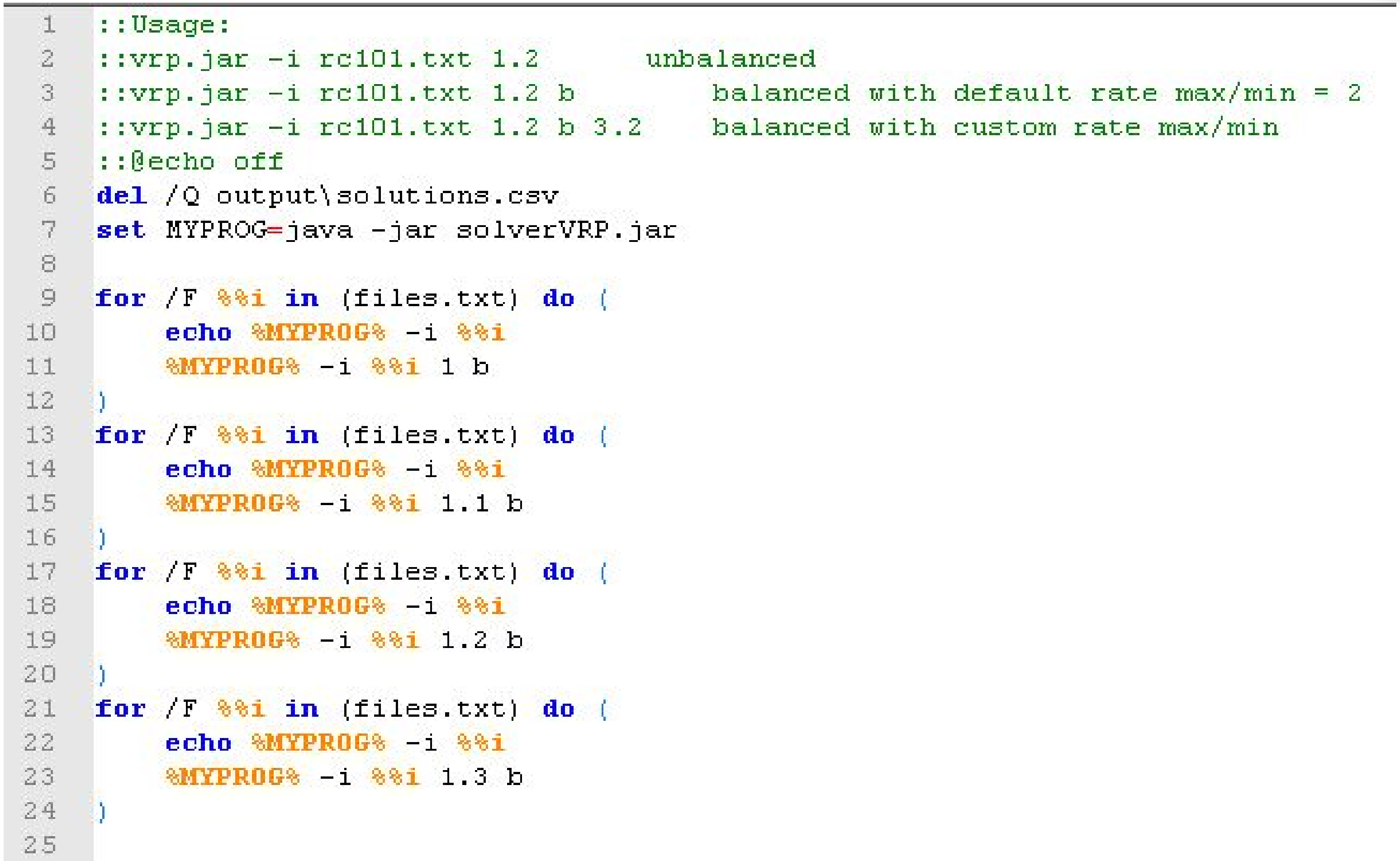
➔Command line parameters that have to be passed to the jar

- ◆Multiplicative factor to be applied for number of vehicles
  - Used to increase the fleet size by a certain specified percentage
  - Defined as a double and has a default value of 1.0
- ◆Balanced/unbalanced solution
  - Default to unbalanced, which means that the balancing soft constraints are not applied
- ◆Balance factor
  - It defines the maximum number of jobs served in a route, divided by the minimum number of jobs served in a route
  - Used only when a balanced solution is required by the user, and it has a default value of 2

Automatic execution

- ➔ The run.bat file is formed of four main loops
  - ◆ Normal fleet size
  - ◆ 10% more vehicles
  - ◆ 20% more vehicles
  - ◆ 30% more vehicles

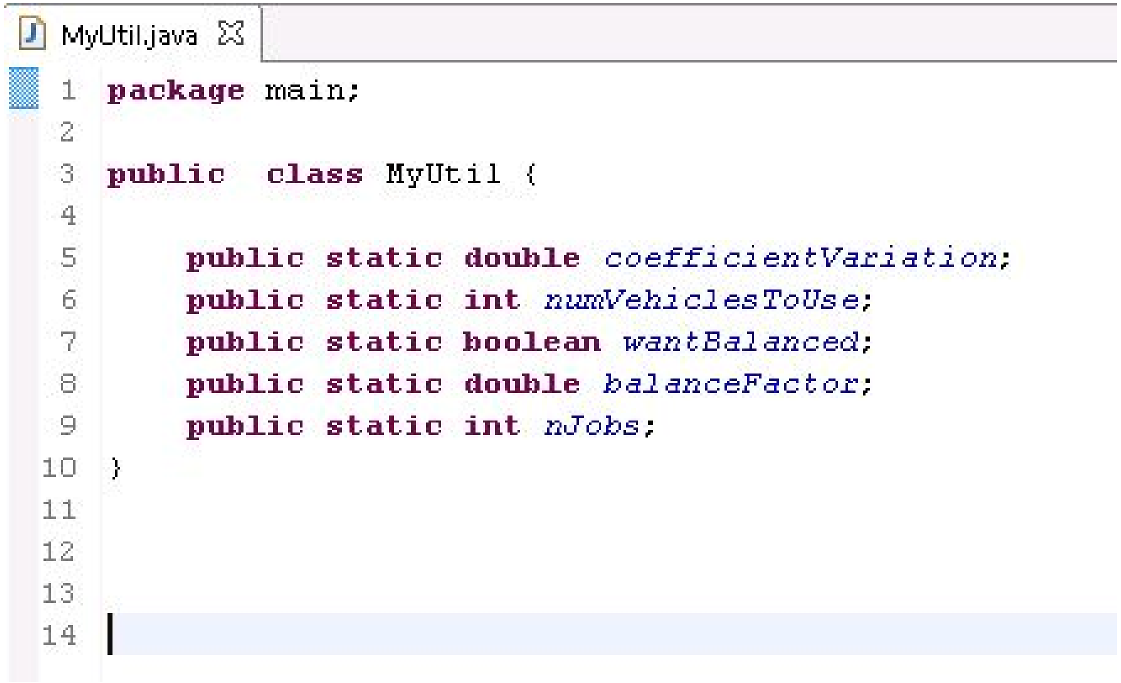
Automatic execution



```
::Usage:
::vrp.jar -i rc101.txt 1.2      unbalanced
::vrp.jar -i rc101.txt 1.2 b        balanced with default rate max/min = 2
::vrp.jar -i rc101.txt 1.2 b 3.2    balanced with custom rate max/min
::@echo off
del /Q output\solutions.csv
set MYPROG=java -jar solverVRP.jar

for /F %%i in (files.txt) do (
    echo %MYPROG% -i %%i
    %MYPROG% -i %%i 1 b
)
for /F %%i in (files.txt) do (
    echo %MYPROG% -i %%i
    %MYPROG% -i %%i 1.1 b
)
for /F %%i in (files.txt) do (
    echo %MYPROG% -i %%i
    %MYPROG% -i %%i 1.2 b
)
for /F %%i in (files.txt) do (
    echo %MYPROG% -i %%i
    %MYPROG% -i %%i 1.3 b
)

```

Project structure

- ➔ Some files have been added to the project
  - ◆ MyUtils.java in /main
    - contains shared global variables



```java
package main;

public  class MyUtil {

	public static double coefficientVariation;
	public static int numVehiclesToUse;
	public static boolean wantBalanced;
	public static double balanceFactor;
	public static int nJobs;
}




```

Project structure

- ➔ Some files have been added to the project
  - ◆ instanceVehicles.txt in /input
    - contains the name of the instance and fleet size of all 33 instance files used in the first assignment (C101-C109, C201-C208, RC101-RC108, RC201-208) and also the “R” instances (R101-R112,R201-R211)
    - E.g.: RC101.txt,15

Project structure

- ➔ Some files have been modified to fit our needs
  - ◆ algorithmConfig.xml in /input
    - the same configuration file used in assignment 1
  - ◆ OROutils.java in /main
    - added a method that writes three more columns inside solutions.csv: number of unassignedJobs, maxJobsInRoute, minJobsInRoute
  - ◆ Main.java in /main
    - added hard constraint and new CostCalculator
  - ◆ SolomonReader.java in /jsprit/instance/reader
    - the FleetSize was set to FINITE and the addition of vehicles is based on instanceVehicles.txt

Upper limit on fleet size

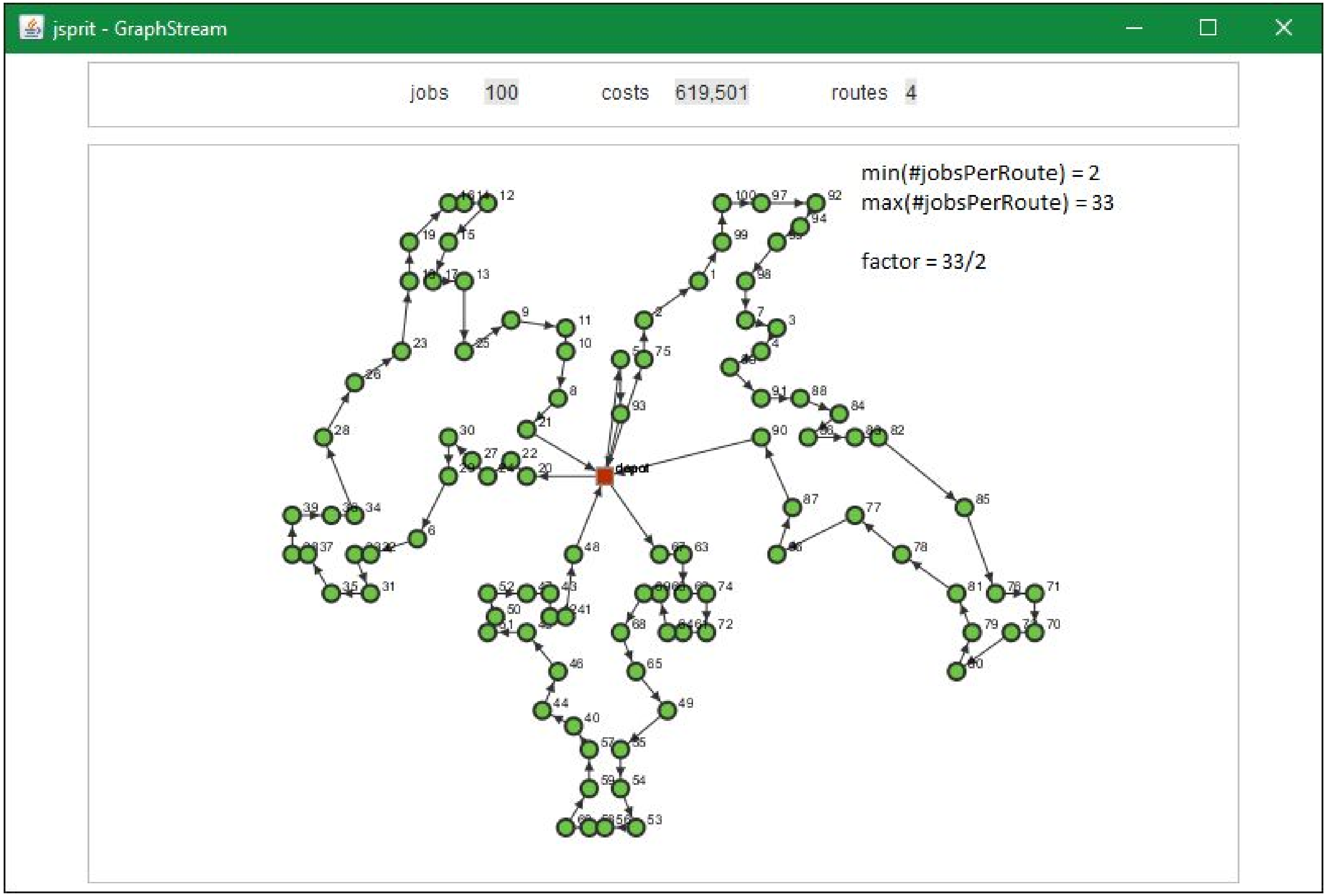
- ➔ It is the first step in order to try to fix the number of vehicles
- ➔ Modifications into the SolomonReader class
  - ◆ vrpBuilder.setFleetSize(FleetSize.FINITE)
  - ◆ Only add the desired number of vehicles VehicleImpl.Builder. newInstance("solomonVehicle" + i)
  - ◆ i goes from 1 to the number read from instanceVehicles.txt

- ➔ However, this is an upper limit, so we need to find a way to use all vehicles.

Soft constraint

- ➔ The main idea is to assign penalties for solutions we don’t like
- ➔ VRP is a min problem, it will select solution with lowest cost
  - ◆ lowest penalties if a sufficient number of iterations is done
- ➔ Penalties will be applied in the following cases
  - ◆ some vehicles are not used
  - ◆ there are some unassigned jobs
  - ◆ the solution is unbalanced (defined a maximum ratio between maxJobsInRoute/minJobsInRoute, which has a default value of 2 but can be customized)
- ➔ The penalties are proportional to the violations

Example: C201 unbalanced

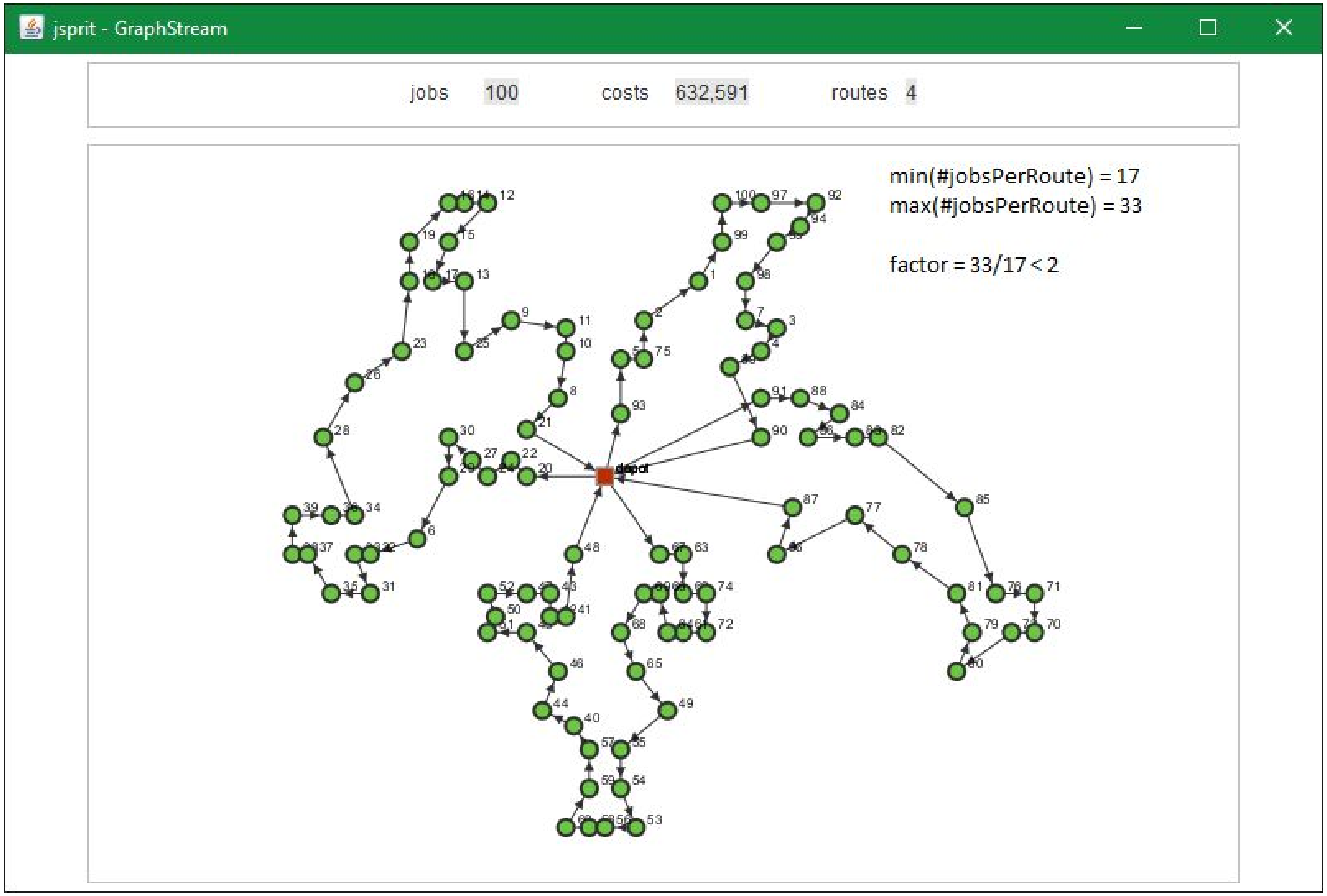


jobs 100 costs 619,501 routes 4

min(#jobsPerRoute) = 2
max(#jobsPerRoute) = 33

factor = 33/2

Example: C201 balanced

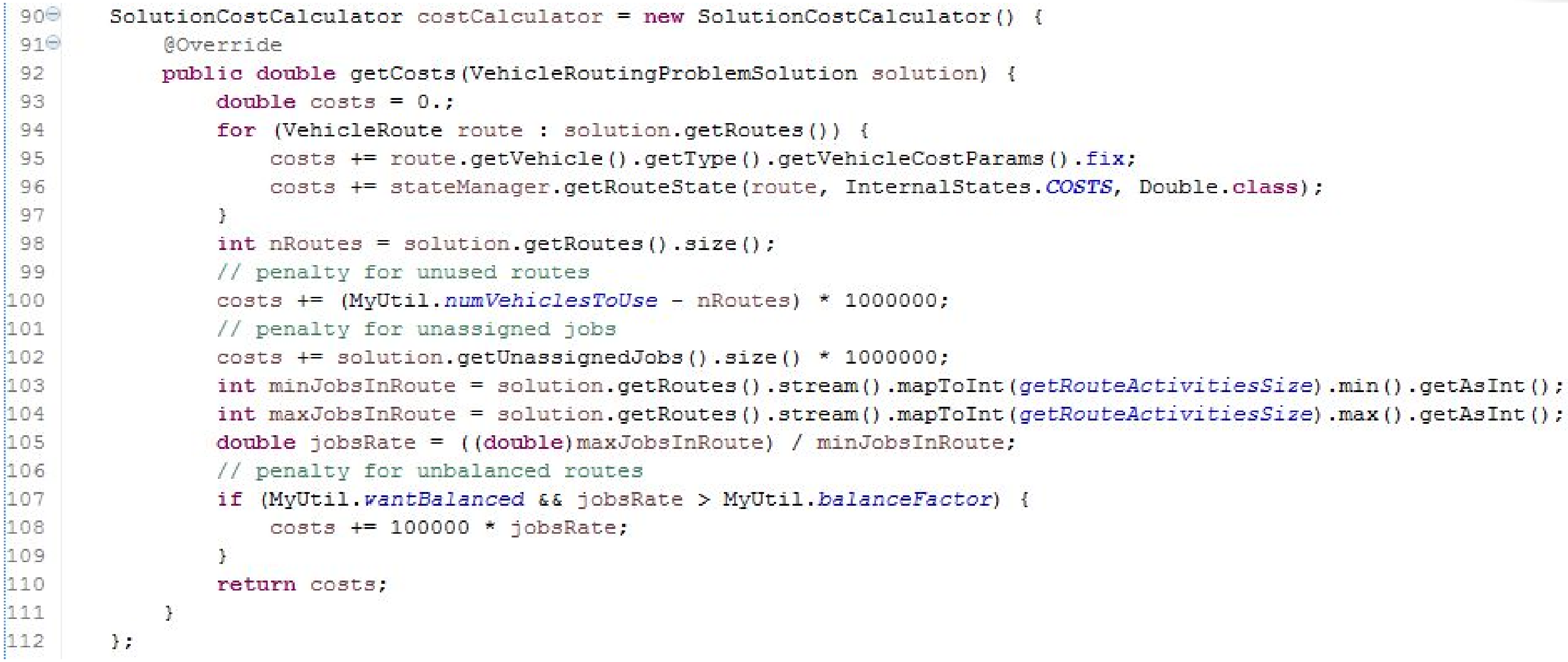


jobs 100 costs 632,591 routes 4

min(#jobsPerRoute) = 17
max(#jobsPerRoute) = 33

factor = 33/17 < 2

Soft constraint implementation

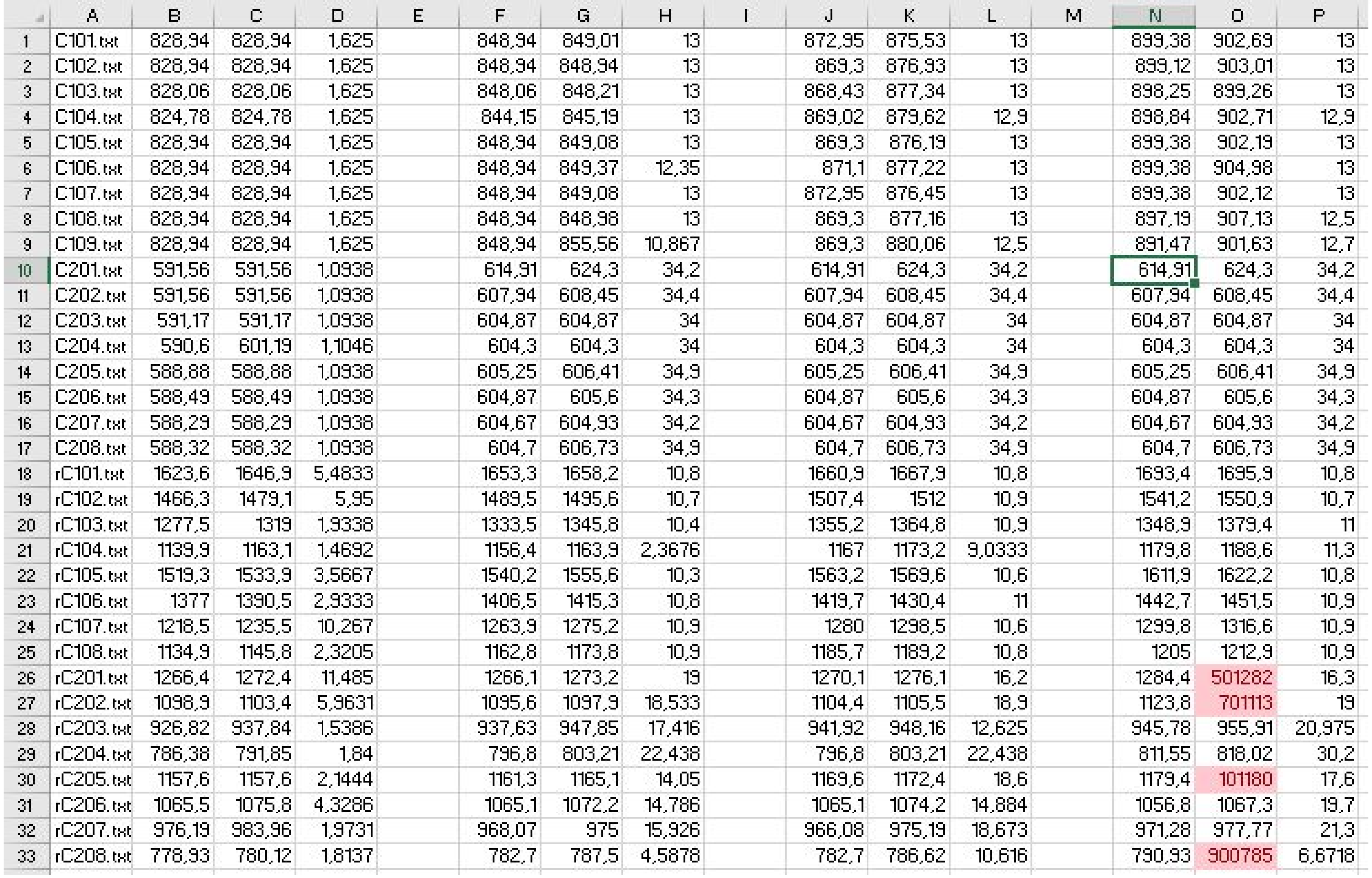


```java
SolutionCostCalculator costCalculator = new SolutionCostCalculator() {
    @Override
    public double getCosts(VehicleRoutingProblemSolution solution) {
        double costs = 0.;
        for (VehicleRoute route : solution.getRoutes()) {
            costs += route.getVehicle().getType().getVehicleCostParams().fix;
            costs += stateManager.getRouteState(route, InternalStates.COSTS, Double.class);
        }
        int nRoutes = solution.getRoutes().size();
        // penalty for unused routes
        costs += (MyUtil.numVehiclesToUse - nRoutes) * 1000000;
        // penalty for unassigned jobs
        costs += solution.getUnassignedJobs().size() * 1000000;
        int minJobsInRoute = solution.getRoutes().stream().mapToInt(getRouteActivitiesSize).min().getAsInt();
        int maxJobsInRoute = solution.getRoutes().stream().mapToInt(getRouteActivitiesSize).max().getAsInt();
        double jobsRate = ((double)maxJobsInRoute) / minJobsInRoute;
        // penalty for unbalanced routes
        if (MyUtil.wantBalanced && jobsRate > MyUtil.balanceFactor) {
            costs += 100000 * jobsRate;
        }
        return costs;
    }
};
```

Soft constraint test

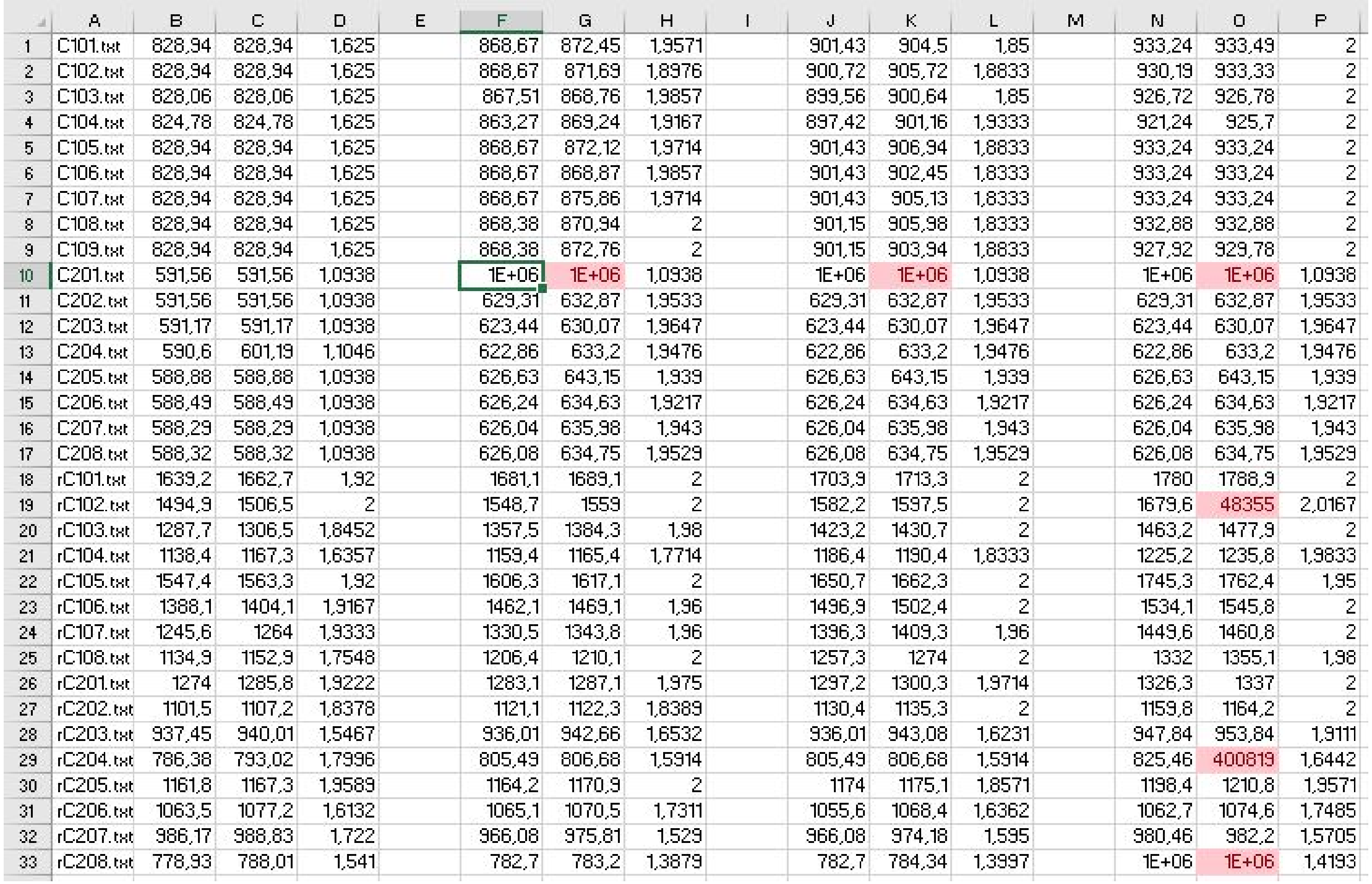
- ➔ All instances were run only with soft constraint (and with different requirements on balance)
- ➔ Some solution costs were penalized because unable to use all the vehicles
- ➔ The search space was too large to find solutions without penalties, because soft constraint only acts when a complete solution is created
- ➔ Need to add some hard constraints to the problem in order to exclude solutions not enough good during the recreate phase of the algorithm

Results for unbalanced



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	C101.txt	828,94	828,94	1,625		848,94	849,01	13		872,95	875,53	13		899,38	902,69	13
2	C102.txt	828,94	828,94	1,625		848,94	848,94	13		869,3	876,93	13		899,12	903,01	13
3	C103.txt	828,06	828,06	1,625		848,06	848,21	13		868,43	877,34	13		898,25	899,26	13
4	C104.txt	824,78	824,78	1,625		844,15	845,19	13		869,02	879,62	12,9		898,84	902,71	12,9
5	C105.txt	828,94	828,94	1,625		848,94	849,08	13		869,3	876,19	13		899,38	902,19	13
6	C106.txt	828,94	828,94	1,625		848,94	849,37	12,35		871,1	877,22	13		899,38	904,98	13
7	C107.txt	828,94	828,94	1,625		848,94	849,08	13		872,95	876,45	13		899,38	902,12	13
8	C108.txt	828,94	828,94	1,625		848,94	848,98	13		869,3	877,16	13		897,19	907,13	12,5
9	C109.txt	828,94	828,94	1,625		848,94	855,56	10,867		869,3	880,06	12,5		891,47	901,63	12,7
10	C201.txt	591,56	591,56	1,0938		614,91	624,3	34,2		614,91	624,3	34,2		614,91	624,3	34,2
11	C202.txt	591,56	591,56	1,0938		607,94	608,45	34,4		607,94	608,45	34,4		607,94	608,45	34,4
12	C203.txt	591,17	591,17	1,0938		604,87	604,87	34		604,87	604,87	34		604,87	604,87	34
13	C204.txt	590,6	601,19	1,1046		604,3	604,3	34		604,3	604,3	34		604,3	604,3	34
14	C205.txt	588,88	588,88	1,0938		605,25	606,41	34,9		605,25	606,41	34,9		605,25	606,41	34,9
15	C206.txt	588,49	588,49	1,0938		604,87	605,6	34,3		604,87	605,6	34,3		604,87	605,6	34,3
16	C207.txt	588,29	588,29	1,0938		604,67	604,93	34,2		604,67	604,93	34,2		604,67	604,93	34,2
17	C208.txt	588,32	588,32	1,0938		604,7	606,73	34,9		604,7	606,73	34,9		604,7	606,73	34,9
18	rC101.txt	1623,6	1646,9	5,4833		1653,3	1658,2	10,8		1660,9	1667,9	10,8		1693,4	1695,9	10,8
19	rC102.txt	1466,3	1479,1	5,95		1489,5	1495,6	10,7		1507,4	1512	10,9		1541,2	1550,9	10,7
20	rC103.txt	1277,5	1319	1,9338		1333,5	1345,8	10,4		1355,2	1364,8	10,9		1348,9	1379,4	11
21	rC104.txt	1139,9	1163,1	1,4692		1156,4	1163,9	2,3676		1167	1173,2	9,0333		1179,8	1188,6	11,3
22	rC105.txt	1519,3	1533,9	3,5667		1540,2	1555,6	10,3		1563,2	1569,6	10,6		1611,9	1622,2	10,8
23	rC106.txt	1377	1390,5	2,9333		1406,5	1415,3	10,8		1419,7	1430,4	11		1442,7	1451,5	10,9
24	rC107.txt	1218,5	1235,5	10,267		1263,9	1275,2	10,9		1280	1298,5	10,6		1299,8	1316,6	10,9
25	rC108.txt	1134,9	1145,8	2,3205		1162,8	1173,8	10,9		1185,7	1189,2	10,8		1205	1212,9	10,9
26	rC201.txt	1266,4	1272,4	11,485		1266,1	1273,2	19		1270,1	1276,1	16,2		1284,4	501282	16,3
27	rC202.txt	1098,9	1103,4	5,9631		1095,6	1097,9	18,533		1104,4	1105,5	18,9		1123,8	701113	19
28	rC203.txt	926,82	937,84	1,5386		937,63	947,85	17,416		941,92	948,16	12,625		945,78	955,91	20,975
29	rC204.txt	786,38	791,85	1,84		796,8	803,21	22,438		796,8	803,21	22,438		811,55	818,02	30,2
30	rC205.txt	1157,6	1157,6	2,1444		1161,3	1165,1	14,05		1169,6	1172,4	18,6		1179,4	101180	17,6
31	rC206.txt	1065,5	1075,8	4,3286		1065,1	1072,2	14,786		1065,1	1074,2	14,884		1056,8	1067,3	19,7
32	rC207.txt	976,19	983,96	1,9731		968,07	975	15,926		966,08	975,19	18,673		971,28	977,77	21,3
33	rC208.txt	778,93	780,12	1,8137		782,7	787,5	4,5878		782,7	786,62	10,616		790,93	900785	6,6718

Results for balanced, with default value



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	C101.txt	828,94	828,94	1,625		868,67	872,45	1,9571		901,43	904,5	1,85		933,24	933,49	2
2	C102.txt	828,94	828,94	1,625		868,67	871,69	1,8976		900,72	905,72	1,8833		930,19	933,33	2
3	C103.txt	828,06	828,06	1,625		867,51	868,76	1,9857		899,56	900,64	1,85		926,72	926,78	2
4	C104.txt	824,78	824,78	1,625		863,27	869,24	1,9167		897,42	901,16	1,9333		921,24	925,7	2
5	C105.txt	828,94	828,94	1,625		868,67	872,12	1,9714		901,43	906,94	1,8833		933,24	933,24	2
6	C106.txt	828,94	828,94	1,625		868,67	868,87	1,9857		901,43	902,45	1,8333		933,24	933,24	2
7	C107.txt	828,94	828,94	1,625		868,67	875,86	1,9714		901,43	905,13	1,8333		933,24	933,24	2
8	C108.txt	828,94	828,94	1,625		868,38	870,94	2		901,15	905,98	1,8333		932,88	932,88	2
9	C109.txt	828,94	828,94	1,625		868,38	872,76	2		901,15	903,94	1,8833		927,92	929,78	2
10	C201.txt	591,56	591,56	1,0938		1E+06	1E+06	1,0938		1E+06	1E+06	1,0938		1E+06	1E+06	1,0938
11	C202.txt	591,56	591,56	1,0938		629,31	632,87	1,9533		629,31	632,87	1,9533		629,31	632,87	1,9533
12	C203.txt	591,17	591,17	1,0938		623,44	630,07	1,9647		623,44	630,07	1,9647		623,44	630,07	1,9647
13	C204.txt	590,6	601,19	1,1046		622,86	633,2	1,9476		622,86	633,2	1,9476		622,86	633,2	1,9476
14	C205.txt	588,88	588,88	1,0938		626,63	643,15	1,939		626,63	643,15	1,939		626,63	643,15	1,939
15	C206.txt	588,49	588,49	1,0938		626,24	634,63	1,9217		626,24	634,63	1,9217		626,24	634,63	1,9217
16	C207.txt	588,29	588,29	1,0938		626,04	635,98	1,943		626,04	635,98	1,943		626,04	635,98	1,943
17	C208.txt	588,32	588,32	1,0938		626,08	634,75	1,9529		626,08	634,75	1,9529		626,08	634,75	1,9529
18	rC101.txt	1639,2	1662,7	1,92		1681,1	1689,1	2		1703,9	1713,3	2		1780	1788,9	2
19	rC102.txt	1494,9	1506,5	2		1548,7	1559	2		1582,2	1597,5	2		1679,6	48355	2,0167
20	rC103.txt	1287,7	1306,5	1,8452		1357,5	1384,3	1,98		1423,2	1430,7	2		1463,2	1477,9	2
21	rC104.txt	1138,4	1167,3	1,6357		1159,4	1165,4	1,7714		1186,4	1190,4	1,8333		1225,2	1235,8	1,9833
22	rC105.txt	1547,4	1563,3	1,92		1606,3	1617,1	2		1650,7	1662,3	2		1745,3	1762,4	1,95
23	rC106.txt	1388,1	1404,1	1,9167		1462,1	1469,1	1,96		1496,9	1502,4	2		1534,1	1545,8	2
24	rC107.txt	1245,6	1264	1,9333		1330,5	1343,8	1,96		1396,3	1409,3	1,96		1449,6	1460,8	2
25	rC108.txt	1134,9	1152,9	1,7548		1206,4	1210,1	2		1257,3	1274	2		1332	1355,1	1,98
26	rC201.txt	1274	1285,8	1,9222		1283,1	1287,1	1,975		1297,2	1300,3	1,9714		1326,3	1337	2
27	rC202.txt	1101,5	1107,2	1,8378		1121,1	1122,3	1,8389		1130,4	1135,3	2		1159,8	1164,2	2
28	rC203.txt	937,45	940,01	1,5467		936,01	942,66	1,6532		936,01	943,08	1,6231		947,84	953,84	1,9111
29	rC204.txt	786,38	793,02	1,7996		805,49	806,68	1,5914		805,49	806,68	1,5914		825,46	400819	1,6442
30	rC205.txt	1161,8	1167,3	1,9589		1164,2	1170,9	2		1174	1175,1	1,8571		1198,4	1210,8	1,9571
31	rC206.txt	1063,5	1077,2	1,6132		1065,1	1070,5	1,7311		1055,6	1068,4	1,6362		1062,7	1074,6	1,7485
32	rC207.txt	986,17	988,83	1,722		966,08	975,81	1,529		966,08	974,18	1,595		980,46	982,2	1,5705
33	rC208.txt	778,93	788,01	1,541		782,7	783,2	1,3879		782,7	784,34	1,3997		1E+06	1E+06	1,4193

Hard constraint

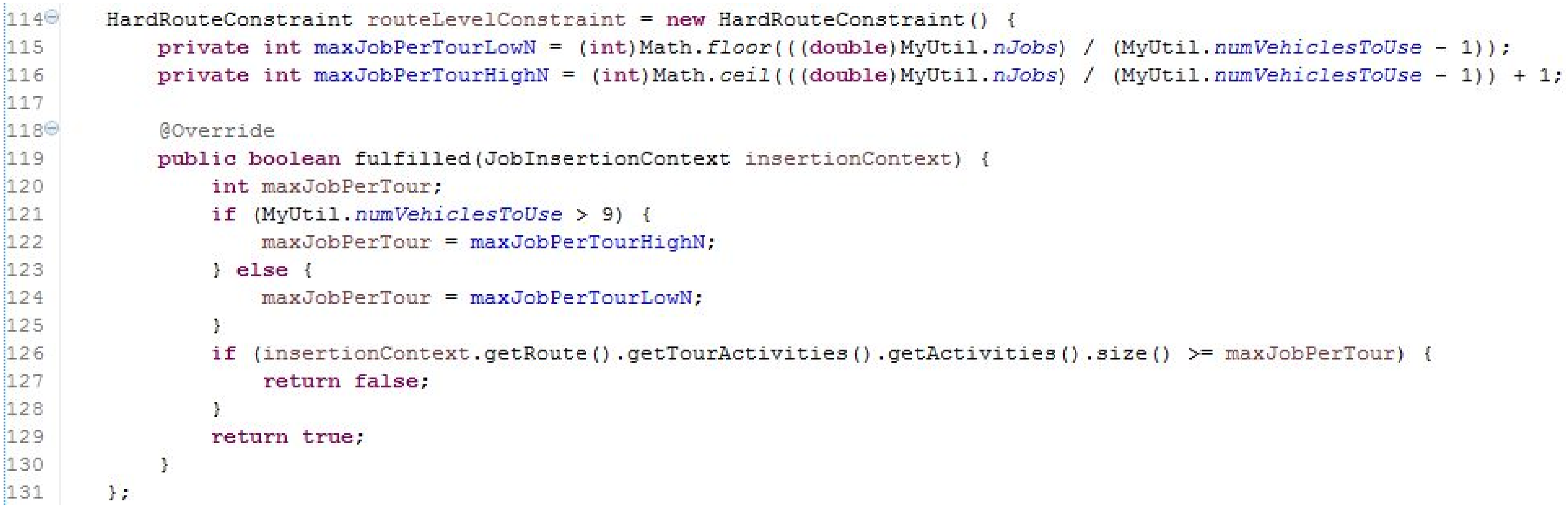
- Different options
  - Limit the capacity of vehicles
  - Limit the number of jobs that can be served by each vehicle
  - Set vehicles’ EarliestStart and LatestArrival
- The second approach is the most naive

Hard constraint

- ➔ Need to find the proper value
  - ◆ Too strict: routes may cover a lot of distance to reach their jobs in case they are spatially unbalanced. This can also cause unassigned jobs because of time-window
    - E.g.: rc101, rc104
  - ◆ Too large: unable to force some instances to use all the vehicles
    - E.g.: c201, rc102, rc204, rc208

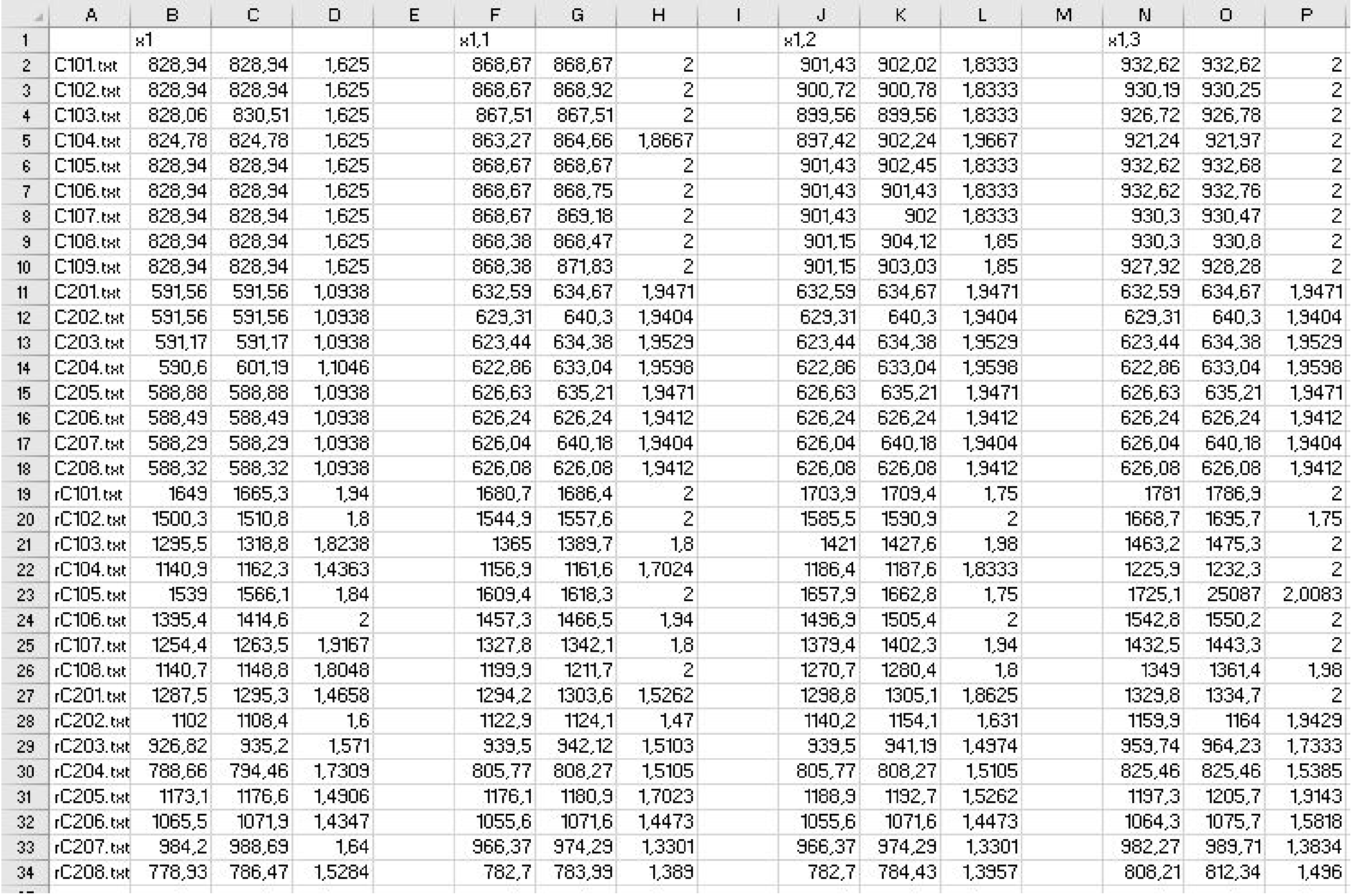
Hard constraint implementation

➔ Used two different formulas because when the fleet size is high, the first formula is too constraining.



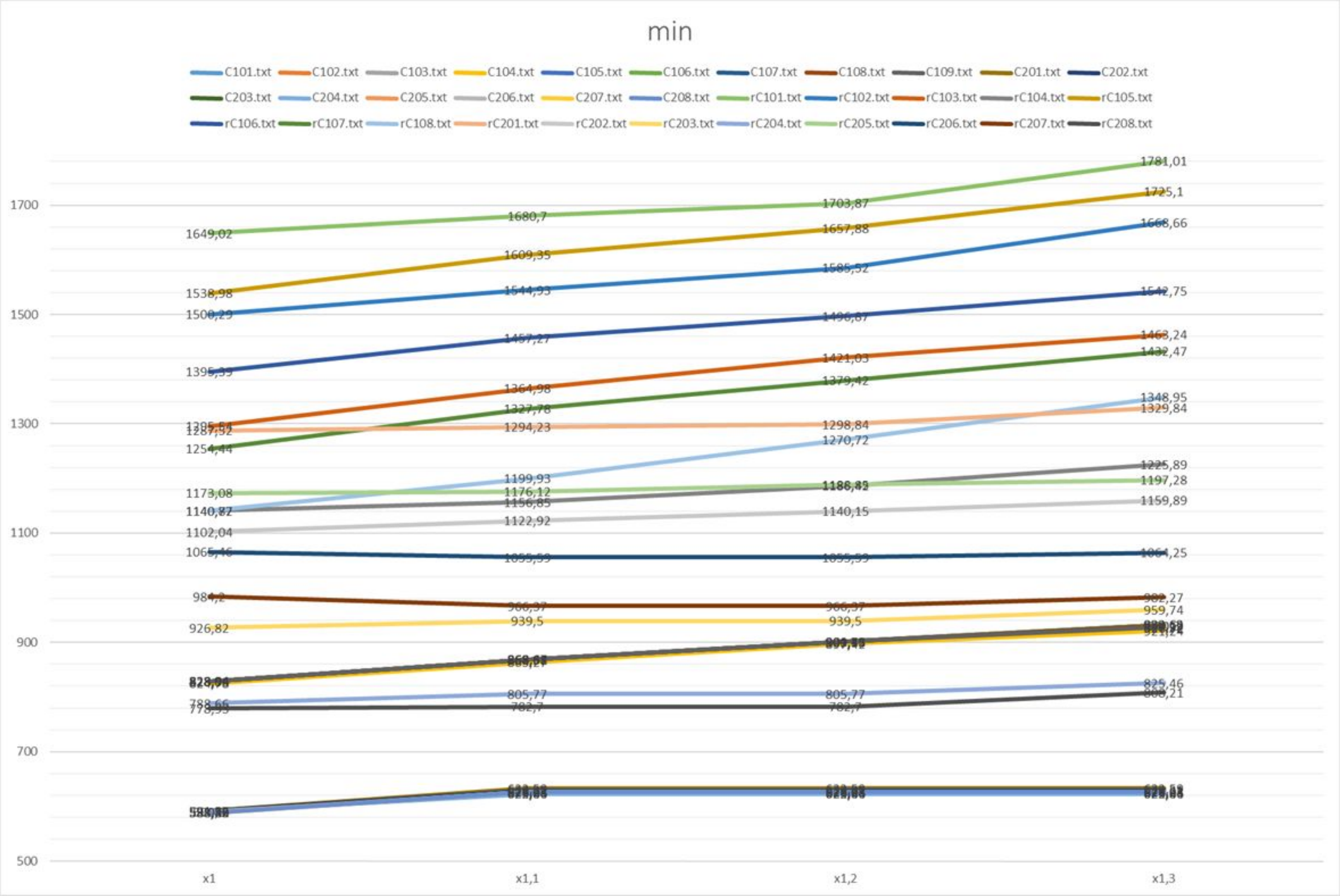
```
HardRouteConstraint routeLevelConstraint = new HardRouteConstraint() {
    private int maxJobPerTourLowN = (int)Math.floor(((double)MyUtil.nJobs) / (MyUtil.numVehiclesToUse - 1));
    private int maxJobPerTourHighN = (int)Math.ceil(((double)MyUtil.nJobs) / (MyUtil.numVehiclesToUse - 1)) + 1;

    @Override
    public boolean fulfilled(JobInsertionContext insertionContext) {
        int maxJobPerTour;
        if (MyUtil.numVehiclesToUse > 9) {
            maxJobPerTour = maxJobPerTourHighN;
        } else {
            maxJobPerTour = maxJobPerTourLowN;
        }
        if (insertionContext.getRoute().getTourActivities().getActivities().size() >= maxJobPerTour) {
            return false;
        }
        return true;
    }
};
```

Final results



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1		x1				x1,1				x1,2				x1,3		
2	C101.txt	828,94	828,94	1,625		868,67	868,67	2		901,43	902,02	1,8333		932,62	932,62	2
3	C102.txt	828,94	828,94	1,625		868,67	868,92	2		900,72	900,78	1,8333		930,19	930,25	2
4	C103.txt	828,06	830,51	1,625		867,51	867,51	2		899,56	899,56	1,8333		926,72	926,78	2
5	C104.txt	824,78	824,78	1,625		863,27	864,66	1,8667		897,42	902,24	1,9667		921,24	921,97	2
6	C105.txt	828,94	828,94	1,625		868,67	868,67	2		901,43	902,45	1,8333		932,62	932,68	2
7	C106.txt	828,94	828,94	1,625		868,67	868,75	2		901,43	901,43	1,8333		932,62	932,76	2
8	C107.txt	828,94	828,94	1,625		868,67	869,18	2		901,43	902	1,8333		930,3	930,47	2
9	C108.txt	828,94	828,94	1,625		868,38	868,47	2		901,15	904,12	1,85		930,3	930,8	2
10	C109.txt	828,94	828,94	1,625		868,38	871,83	2		901,15	903,03	1,85		927,92	928,28	2
11	C201.txt	591,56	591,56	1,0938		632,59	634,67	1,9471		632,59	634,67	1,9471		632,59	634,67	1,9471
12	C202.txt	591,56	591,56	1,0938		629,31	640,3	1,9404		629,31	640,3	1,9404		629,31	640,3	1,9404
13	C203.txt	591,17	591,17	1,0938		623,44	634,38	1,9529		623,44	634,38	1,9529		623,44	634,38	1,9529
14	C204.txt	590,6	601,19	1,1046		622,86	633,04	1,9598		622,86	633,04	1,9598		622,86	633,04	1,9598
15	C205.txt	588,88	588,88	1,0938		626,63	635,21	1,9471		626,63	635,21	1,9471		626,63	635,21	1,9471
16	C206.txt	588,49	588,49	1,0938		626,24	626,24	1,9412		626,24	626,24	1,9412		626,24	626,24	1,9412
17	C207.txt	588,29	588,29	1,0938		626,04	640,18	1,9404		626,04	640,18	1,9404		626,04	640,18	1,9404
18	C208.txt	588,32	588,32	1,0938		626,08	626,08	1,9412		626,08	626,08	1,9412		626,08	626,08	1,9412
19	rC101.txt	1649	1665,3	1,94		1680,7	1686,4	2		1703,9	1709,4	1,75		1781	1786,9	2
20	rC102.txt	1500,3	1510,8	1,8		1544,9	1557,6	2		1585,5	1590,9	2		1668,7	1695,7	1,75
21	rC103.txt	1295,5	1318,8	1,8238		1365	1389,7	1,8		1421	1427,6	1,98		1463,2	1475,3	2
22	rC104.txt	1140,9	1162,3	1,4363		1156,9	1161,6	1,7024		1186,4	1187,6	1,8333		1225,9	1232,3	2
23	rC105.txt	1539	1566,1	1,84		1609,4	1618,3	2		1657,9	1662,8	1,75		1725,1	25087	2,0083
24	rC106.txt	1395,4	1414,6	2		1457,3	1466,5	1,94		1496,9	1505,4	2		1542,8	1550,2	2
25	rC107.txt	1254,4	1263,5	1,9167		1327,8	1342,1	1,8		1379,4	1402,3	1,94		1432,5	1443,3	2
26	rC108.txt	1140,7	1148,8	1,8048		1199,9	1211,7	2		1270,7	1280,4	1,8		1349	1361,4	1,98
27	rC201.txt	1287,5	1295,3	1,4658		1294,2	1303,6	1,5262		1298,8	1305,1	1,8625		1329,8	1334,7	2
28	rC202.txt	1102	1108,4	1,6		1122,9	1124,1	1,47		1140,2	1154,1	1,631		1159,9	1164	1,9429
29	rC203.txt	926,82	935,2	1,571		939,5	942,12	1,5103		939,5	941,19	1,4974		959,74	964,23	1,7333
30	rC204.txt	788,66	794,46	1,7309		805,77	808,27	1,5105		805,77	808,27	1,5105		825,46	825,46	1,5385
31	rC205.txt	1173,1	1176,6	1,4906		1176,1	1180,9	1,7023		1188,9	1192,7	1,5262		1197,3	1205,7	1,9143
32	rC206.txt	1065,5	1071,9	1,4347		1055,6	1071,6	1,4473		1055,6	1071,6	1,4473		1064,3	1075,7	1,5818
33	rC207.txt	984,2	988,69	1,64		966,37	974,29	1,3301		966,37	974,29	1,3301		982,27	989,71	1,3834
34	rC208.txt	778,93	786,47	1,5284		782,7	783,99	1,389		782,7	784,43	1,3957		808,21	812,34	1,496

Final results (graphical)



Final notes

- ➔ For the majority of instances, the cost of solutions increases with the number of vehicles
- ➔ Instead, for rc206 and rc207 the cost decreases when increasing the fleet size of 10%

	5 vehicles	6 vehicles
rc206	1065,46	1055,59
rc207	984,2	966,37

- ➔ This happens because we chose the number of vehicles by using the minimum value of vehicles found in the results of the first assignment.