Supplementary Material for: New Heuristic and Evolutionary Operators for the Multi-Objective Urban Transit Routing Problem, CEC 2013

Christine L Mumford
School of Computer Science & Informatics,
Cardiff University

April 6, 2016

1 Introduction

These notes provide additional material to support the above paper, describing the following:

- Visualization
- Data Set Generation
- Lower Bounds for passenger costs and operator costs
- Timings in relation to the size of the transit network (i.e.,(number of routes)*(average number of nodes on each route)
- File formats for the data files for instances Mandl, Mumford0, Mumford1, Mumford2 and Mumford3
- Link to online python program to evaluate route sets, to allow researcher to validate their results
- File formats for the results files

2 Visualization

The data sets can be visualized by plotting the supplied coordinates files, <instance>Coords.txt (Fig. 1).

3 Data Set Generation

The data generation software was written to create instances according to input parameters supplied by the user. Each instance consists of a connected transport network with roads and bus stops (or other vehicle stopping nodes), but no routes, because it it the function of the software described in the CEC paper to construct these routes. Each instance is constrained within the boundaries of an enclosing square region. To maintain a similar density of points, a side length proportional to the square root of the number of nodes is used. The coordinates for the nodes are generated within these boundaries from a uniform

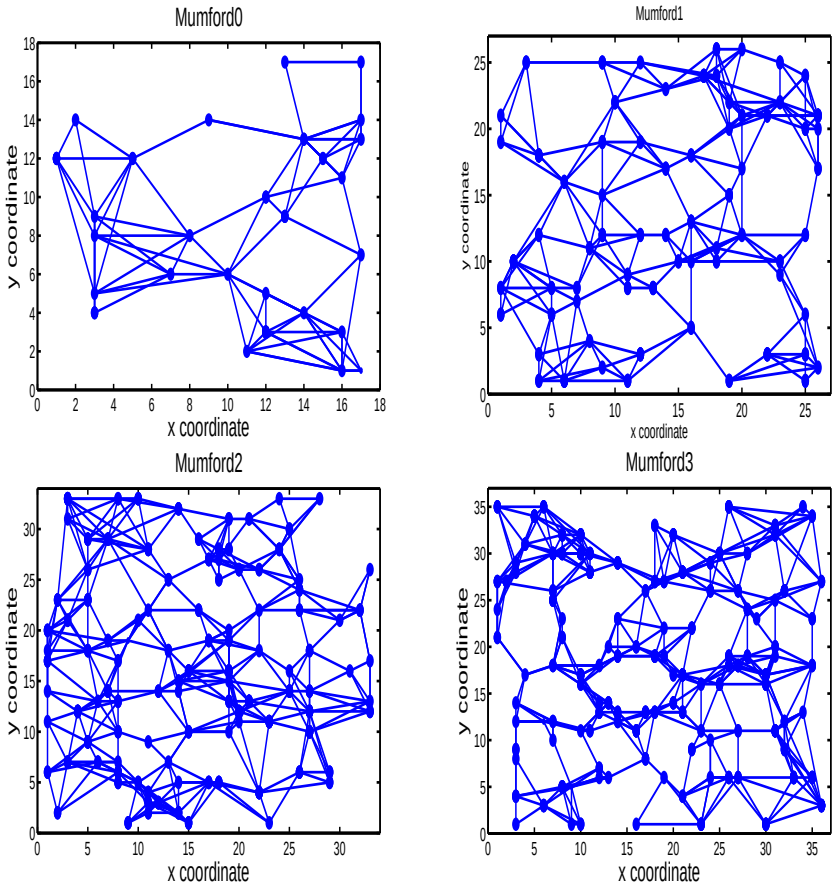

Figure 1: Transport (road) Networks for the new Data Sets

random distribution, with the number of nodes and links input by the user, giving control over the size and the complexity of the network, but leaving the software to decide exactly where the nodes and links will be placed within the enclosing area. For convenience, the Euclidean distances between the generated nodes corresponds to the travel time in minutes (the size of the enclosing square is chosen to ensure that these travel times are reasonable).

The general level of demand is also determined by the user, but the actual demand values between each pair of nodes is randomly generated by our software using the input parameters. The user will supply an upper and a lower bound (i.e., a demand range), and this range will apply to every node pair in the network. As mentioned in the paper, the data sets Mumford1, Mumford2 and Mumford3 were based on information manually extracted from bus route network maps obtained for real cities: one in China (Yubei) and two in the UK (Brighton and Cardiff). The approximate properties we used to help us construct the data can be seen in Table 1, and the table defining the actual properties of the test data is repeated from the paper for convenience (Table 2).

Careful attention was paid to the generation of links in the design of our software. Firstly we require that all transport networks are connected; i.e., that at least one path will exist between every pair of nodes in the network, so that each part of the network is reachable from any given starting point.

Furthermore, we wish to ensure that network connectivity resembles a real road network, to avoid the excessive crossings of links. With these criteria in mind, a two stage approach to generating the links was adopted:

1. The construction a minimum spanning tree (MST), according to the Euclidean distances between each pair of nodes, followed by
2. The addition of further links to match user requirements, choosing the shorter links first.

The MST is constructed using Kruskal's algorithm, and then the remaining links are selected as follows:

1. A node is chosen at random;
2. The shortest unused link out of this node is selected and added to the network;
3. Repeat from 1. until the network has the required number of links.

Table 1: Properties of Real Data Sets

Location	Number of Nodes and Links	Number of Routes	Nodes /Route	Mean Links /Node	Mean Frequency of Routes Visiting Each Node
Yubei	70 & 210	15	10 - 30	3	4.29
Brighton	110 & 385	56	10 - 22	3.5	8.15
Cardiff	127 & 425	60	12 - 25	3.35	8.98

Table 2: Our Data Sets: Corrections of errors published in [?] indicated in red

Instance	Number of Nodes and Links	Number of Routes	Nodes /Route	LB_{pass} (mins)	LB_{op} = Minimum R(L) (mins)
Mandl	15 & 20	4-8	2-8	10.0058	63
Mumford0	30 & 90	12	2-15	13.0121	94
Mumford1	70 & 210	15	10 - 30	19.2695	345*
Mumford2	110 & 385	56	10 - 22	22.1689	864*
Mumford3	127 & 425	60	12 - 25	24.7453	982*

4 Lower Bounds for passenger costs and operator costs

Lower bounds can be useful for estimating the solution quality of minimization problems when optima are not known. The lower bound for the passenger cost was well covered in the main paper. It is computed by simply assuming that each passenger can travel from source to destination by travelling along the shortest path in the transport network (with no transfers).

$$C_P(R) = \frac{\sum_{i,j=1}^{n} d_{ij}\alpha_{ij}(R)}{\sum_{i,j=1}^{n} d_{ij}} \tag{1}$$

where α_{ij} is the shortest journey time from i to j using route set R. Recall that the operator cost is evaluated by adding together the lengths (travel time) of all the routes in a particular route set. Thus, a lower bound for operator cost is given by:

$$LB_{op} = min[R(L)] \tag{2}$$

A value for $min[R(L)]$ can be obtained by computing a minimum spanning tree on the transport network. However, a better lower bound is found by considering the number of routes in a route set and the allowable route lengths. If we specify the minimum number of nodes allowed in a route as MIN, it follows that the minimum number of edges in a given route within a route set, R, is $MIN-1$. Thus the minimum number of edges that can be found in a route set is $max(r \times (MIN-1), (n-1))$. (For the route set to be connected, there must be at least $(n-1)$ edges to form a minimum spanning tree). $min(R(L))$ is calculated in two stages:

1. compute the sum of the $n-1$ minimum spanning tree edges in the transport network,
2. if $(MIN-1) > (n-1)$ add to the aforementioned MST length the sum of the lengths of all further edges that need to be added, in order to bring the total number of edges to $r \times (MIN-1)$.

Assume that the $(n-1)$ MST edges can be spread evenly amongst the r routes, with each route being initially assigned $\lfloor \frac{n-1}{r} \rfloor$ edges and the remaining edges, $(n-1) \mod r$, being distributed one per route for as many as is needed, following integer division. For example, if a minimum spanning tree contains 14 edges, and you have 6 routes, then each route will initially receive 2 edges (= 12 edges), and the remaining 2 edges will be added to 2 of the routes, so that 4 routes will each contain a total of 2 edges each, and 2 routes will contain a total of 3 edges each.

A running total for $min(R(L))$ is initialized by adding together the lengths of all the edges in a minimum spanning tree of the transport network. The addition of further edges (if required) then proceeds by first sorting all the transport network edges in non-descending sequence of travel time. These edges are then selected in sequence, one at a time, starting with the smallest. The first selected edge will inevitably be an MST edge, and thus a copy will have been allocated already to one of the routes. However, the remaining $r-1$ routes will not contain this edge, so following selection of the first edge, the lengths of $r-1$ copies of the selected edge will be added (provided at least $r-1$ are required at this stage), and the running total of operator cost will be updated. The process then continues by selecting the next smallest edge from the sorted list. Once again, another $r-1$ copies of this second edge may be added if it is once more an MST edge. If it is not an MST edge then it can be assigned to all r routes. This process is repeated until $r \times (MIN-1)$ edges have been assigned, assigning less than r or $r-1$ copies of a selected edge, as needed, to complete the allocation.

5 Timings in relation to the size of the transit network

Figure 2 shows a graph for the average run times, in seconds, for the replicate runs for Mandl, Mumford0, Mumfor1, Mumford2 and Mumford3. The x axes gives the average size of the transit network (i.e.,(number of routes)*(average number of nodes on each route)), and the y axis gives the average run times in seconds. A quadratic trend provides a good fit, although this is doubtful to continue and the Floyd-Warshall algorithm runs in $O(N^3)$, which is a key component of the evaluation function for each route set.

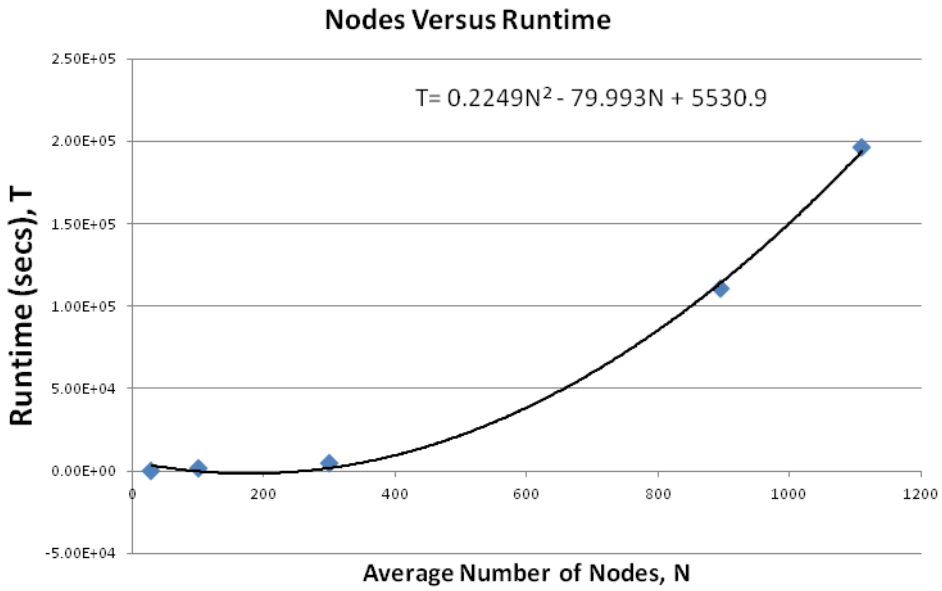

Figure 2: Average run times for MOEA versus number of nodes in the transit network

An Excel spread sheet is provided with the results giving the raw run times of all the replicate runs and showing the computation used for the graph. The sections below outline the formats for the other files.

6 File Formats for the Data Files

The file formats for instances Mandl, Mumford0, Mumford1, Mumford2 and Mumford3 are illustrated through the smallest instance: Mandl's 15 node Swiss network. For each instance there are 3 files:

- <instance>Coords.txt
- <instance>TravelTimes.txt
- <instance>Demand.txt

First of all the coordinates file begins with the number of nodes on the first line (15 in this case). The following 15 lines contain the (x,y) coordinates of the problem. Please note that in the case of Mandl's problem, the coordinates were not supplied in the literature, so the ones included here are just approximate so that a picture can be drawn. However, all the other instances (Mumford0, Mumford1, Mumford2 and Mumford3) have accurate (x, y) coordinates from which all the other input parameters were computed (for the Demand matrix and the Travel Times matrix).

MandlCoords.txt

```
15
1     9
3     8
4.5    7.75
2.75    6.2
0.8    6.6
4.6    6
7    4.5
5.5    5
8.5    6.8
5.8    3.25
3.8    2.25
1.3    3.5
5.25   1
6.7    1.75
6.75    5.8
```

The travel times matrix gives the travel times in minutes between nodes i and j. For all the instances used in the paper, these matrices are symmetrical. Note that the travel times between each node and itself along the diagonal are all zeros. Inf indicates that there is no direct link between node i and j

MandlTravelTimes.txt

```
0    8    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf

8    0    2    3    6    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf

Inf    2    0    Inf    Inf    3    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf

Inf    3    Inf    0    4    4    Inf    Inf    Inf    Inf    Inf    10   Inf    Inf    Inf

Inf    6    Inf    4    0    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf

Inf    Inf    3    4    Inf    0    Inf    2    Inf    Inf    Inf    Inf    Inf    Inf    3

Inf    Inf    Inf    Inf    Inf    Inf    0    Inf    Inf    7    Inf    Inf    Inf    Inf    2

Inf    Inf    Inf    Inf    Inf    2    Inf    0    Inf    8    Inf    Inf    Inf    Inf    2

Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    0    Inf    Inf    Inf    Inf    Inf    8

Inf    Inf    Inf    Inf    Inf    Inf    7    8    Inf    0    5    Inf    10   8    Inf

Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    5    0    10   5    Inf    Inf

Inf    Inf    Inf    10   Inf    Inf    Inf    Inf    Inf    Inf    10   0    Inf    Inf    Inf

Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    10   5    Inf    0    2    Inf

Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    Inf    8    Inf    Inf    2    0    Inf

Inf    Inf    Inf    Inf    Inf    3    2    2    8    Inf    Inf    Inf    Inf    Inf    0
```

Finally, the demand matrix gives the daily demand for travelling between nodes i and j as the number of passengers. Note again that the matrix is symmetrical, and that each passenger therefore performs a return journey (round trip). One interesting feature of Mandl's problem is that there is no demand either to or from node 15, yet it is always included in the route sets because the rules that are used insist on every node being included.

MandlDemand.txt

0	400	200	60	80	150	75	75	30	160	30	25	35	0	0
400	0	50	120	20	180	90	90	15	130	20	10	10	5	0
200	50	0	40	60	180	90	90	15	45	20	10	10	5	0
60	120	40	0	50	100	50	50	15	240	40	25	10	5	0
80	20	60	50	0	50	25	25	10	120	20	15	5	0	0
150	180	180	100	50	0	100	100	30	880	60	15	15	10	0
75	90	90	50	25	100	0	50	15	440	35	10	10	5	0
75	90	90	50	25	100	50	0	15	440	35	10	10	5	0
30	15	15	15	10	30	15	15	0	140	20	5	0	0	0
160	130	45	240	120	880	440	440	140	0	600	250	500	200	0
30	20	20	40	20	60	35	35	20	600	0	75	95	15	0
25	10	10	25	15	15	10	10	5	250	75	0	70	0	0
35	10	10	10	5	15	10	10	0	500	95	70	0	45	0
0	5	5	5	0	10	5	5	0	200	15	0	45	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

7 Python Program for Evaluating Route Sets

Click here to use an online evaluation tool for evaluating your own route sets, according to the constraints we use in our papers defined in Table 2. All our results assume a 5 minute waiting time / transfer penalty. Please note that there is a runtime limit of 30 seconds on the website, so this program will only work on relatively small instances. For example, it will work on route sets for Mandl, Mumford0 and Mumford1, but probably time out for route sets derived from larger networks (e.g., Mumford2 and Mumford3). (A Python program for download is in preparation and will be posted soon). The evaluation software gives researchers the opportunity to ensure that their results are comparable with ours. Over a period of time we have found that errors in the evaluations procedure can be very tricky to detect, and we believe that we have now ironed them all out. This has been achieved by different researchers independently coding the evaluation procedure, followed by cooperation to debug and validate.

Click here to download Mandl test files to try out the program (TravelTimes, Demand and a 6 route solution).

8 File formats for the results files

There are TWO folders of results: *MandlCompareFac2009*, which gives the results for that go with Table 2 in the CEC paper, and MainResults, which provides the results that go with Table 3 in the paper. Both of these folders include README files to explain the included results in detail. In Table 2, the results are compared with the [Fan 2009] paper, and the results files consist of 10 files for each version of Mandl's instance with 4, 6, 7 and 8 routes in the route set. These files are numbered outfile1_1 to outfile1_10. The routes published in the paper are also repeated in the file called *MandlRoutes.docx*.

In the main results folder, for each instance, Mandl, Mumford0, Mumford1, Mumford2 and Mumford3, the SEAMO2 algorithm is run for 200 generations on populations of 200. For these results, 20 individual sets of non-dominated solutions extracted from the replicate runs are stored for each instance in the results files, numbered *outfile1_1..20*. The route sets for the best passenger objective and the best operator objective are reproduced in the file called *Best Route Sets.docx* (these results are not included in the paper because of lack of space.) Finally, for each set of 20 replicate runs the non-dominated solutions have been extracted from the combined set of 20 files (Outfile1_1, Outfile1_2, ..., Outfile1_20). These files are called ParetoMandl.txt, ParetoMumford0.txt, ParetoMumford1.txt, ParetoMumford2.txt and ParetoMumford3.txt. The format of the output files is illustrated below using examples from the results for Mandl's instance.
The file below shows the non dominated dual objective solutions from run number 1. The lefthand column contains the passenger objective, C_p, and the righthand column the operator objective, C_o.

Outfile1_1

```
1.0477200e+001  2.0400000e+002
1.0496468e+001  1.9200000e+002
1.0620424e+001  1.8900000e+002
1.0711625e+001  1.8100000e+002
1.0770713e+001  1.7900000e+002
1.0799615e+001  1.5400000e+002
1.0867694e+001  1.4400000e+002
1.0916506e+001  1.4300000e+002
1.0984586e+001  1.3700000e+002
1.1203597e+001  1.3300000e+002
1.1247913e+001  1.2800000e+002
1.1341040e+001  1.2600000e+002
1.1472704e+001  1.0900000e+002
1.1711625e+001  8.9000000e+001
1.2502890e+001  8.6000000e+001
1.3120745e+001  8.2000000e+001
1.3131021e+001  7.8000000e+001
1.3396917e+001  7.1000000e+001
1.3831728e+001  6.9000000e+001
1.4059730e+001  6.8000000e+001
1.4262042e+001  6.5000000e+001
1.4819525e+001  6.4000000e+001
```

ParetoMandl.txt contains the non dominated solutions extracted from ALL 20 results files.

ParetoMandl.txt

```
10.3301 224
```

```
10.3687 165
10.6519 161
10.7071 159
10.7951 150
10.8677 144
10.9165 143
10.9261 141
10.9396 130
11.097 129
11.0976 124
11.1574 106
11.4631 103
11.5228 102
11.7116 89
12.0963 76
12.6821 73
13.0546 72
13.1496 70
13.3301 69
13.4046 66
13.5478 65
14.2267 64
15.1304 63
```