Online Supplement For "Combination of Meta-Heuristic and Exact Algorithms for Solving Set Covering-Type Optimization Problems"

İbrahim Muter, Ş. İlker Birbil and Güvenç Şahin
Faculty of Engineering and Natural Sciences, Sabancı University, 34956 Istanbul, Turkey,
{imuter@su.sabanciuniv.edu, sibirbil@sabanciuniv.edu, guvencs@sabanciuniv.edu}

1. Implementation Details and Additional Results

The details of the modified version of the tabu search algorithm proposed by Potvin et al. (1996) is explained in this Online Supplement. Let d_{ij} denote the distance between customers i and j, and s^2_{jk} (s^{Or}_{jk}) denote the saving in the travel distance for a 2-Opt (Or-Opt) move oriented at customers j and k. In both procedures, q customers are selected randomly to restrict the possible exchanges. For each of the q customers, only h nearest neighbors are considered for possible exchanges. Suppose we select customer k and $j \in N^h(k)$ (i.e. j is one of the h neighbors of customer k). For a 2-Opt move, the customers j and k should not be in the same route. Let s^2_{jk} be the saving in traveled distance when the routes, on which k and j reside, are changed as follows:

- The route, on which j resides, visits customer k after customer j, and continues with the subsequent customers on the route, on which k resides, in the same order.
- The route, on which k resides, visits customer $j+1$ after customer $k-1$, and continues with the subsequent customers of the route, on which j resides, in the same order.

For an Or-Opt move, k and j can be in either the same route or different routes. Let s^{Or}_{jk} be the saving in traveled distance when the routes, on which k and j reside, are changed according to the first feasible move listed as follows:

- Insert customers j, $j+1$ and $j+2$ between customers k and $k+1$, and let the route, on which j resides, visit customer $j+3$ after $j-1$.
- Insert customers j and $j+1$ between customers k and $k+1$, and let the route, on which j resides, visit customer $j+2$ after $j-1$.

- Insert customer j between customers k and $k+1$, and let the route, on which j resides, visit customer $j+1$ after $j-1$.

For both 2-Opt and Or-Opt exchanges, the inverse move of a feasible selected exchange is declared tabu for a number of iterations determined by the size of the tabu list.

Algorithm 4: Meta-heuristic Component for VRPTW

```
1:  i = 1
2:  while i ≤ I_tot do
3:      for i_2opt=1 to I_2opt do
4:          Select q customers randomly
5:          foreach customer k = 1 to q do
6:              2-Opt(k)
7:          end
8:          k* = arg min_{k∈{1,...,q}} {s^2(j*(k))}
9:          Delete (k*,k*+1) and (j*(k),j*(k)+1)
10:         Add (k*,j*+1) and (j*(k),k*+1)
11:         The inverse move is declared tabu
12:         i ← i+1
13:     end
14:     for i_Oropt=1 to I_Oropt do
15:         Select q customers randomly
16:         foreach customer k = 1 to q do
17:             Or-Opt(k)
18:         end
19:         k* = arg min_{k∈{1,...,q}} {s^Or(j*(k))}
20:         Delete (k*,k*+1), (j*(k)-1,j*(k)) and
            (j*(k)+u(k*,j*(k)),j*(k)+u(k*,j*(k))+1)
21:         Add (k*,j*(k)), (j*(k)+u(k*,j*(k)),k*+1) and
            (j*(k)-1,j*(k)+u(k*,j*(k))+1)
22:         The inverse move is declared tabu
23:         i ← i+1
24:     end
25: end
```

Algorithm 5: 2-Opt(k)

2-Opt(k)
foreach $j \in N^h(k)$ **do**
  if *(j and k are in different routes)* **and** *(2-Opt move oriented at j and k is feasible)* **then**
    $s^2(j) = (d_{j,j+1} + d_{k,k+1}) - (d_{j,k+1} + d_{k,j+1})$
  else
    $s^2(j) = -\infty$
  end
end
$j^*(k) = \arg\min_{j \in N^h(k)}\{s^2(j)\}$

Algorithm 6: Or-Opt(k)

foreach $j \in N^h(k)$ **do**
  if *(it is feasible to insert segment $\{j, j+1, j+2\}$ between k and $k+1$)* **then**
    $s^{Or}(j) = (d_{j,j-1} + d_{k,k+1} + d_{j+2,j+3}) - (d_{k,j} + d_{j+2,k+1} + d_{j-1,j+3})$
    $u(k, j) = 2$
  else if *(it is feasible to insert segment $\{j, j+1\}$ between k and $k+1$)* **then**
    $s^{Or}(j) = (d_{j,j-1} + d_{k,k+1} + d_{j+1,j+2}) - (d_{k,j} + d_{j+1,k+1} + d_{j-1,j+2})$
    $u(k, j) = 1$
  else if *(it is feasible to insert j between k and $k+1$)* **then**
    $s^{Or}(j) = (d_{j,j-1} + d_{k,k+1} + d_{j,j+1}) - (d_{k,j} + d_{j,k+1} + d_{j-1,j+1})$
    $u(k, j) = 0$
  else
    $s^{Or}(j) = -\infty$
  end
end
$j^*(k) = \arg\min_{j \in N^h(k)}\{s^{Or}(j)\}$

Algorithm 7: Column Pool Management for VRPTW

Let y be a newly generated column
foreach *column x in column pool* **do**
    if $x = y$ **then**
        Delete y
    else if *(customers in x = customers in y)* **and** *(distance(x) $\prec$ distance(y))* **then**
        Delete y
    else if *(customers in x = customers in y)* **and** *(distance(x) $\succ$ distance(y))* **then**
        Delete x
    else
        Add y to the column pool
    end
end

Algorithm 8: Exact Algorithm for VRPTW

if $((Z_{UB}(k) - Z_{PLB}(k))/Z_{PLB}(k)) > \alpha$ **then**
    $Z_{PLB}(k) \leftarrow$ `Solve` IP
else
    $Z_{PLB}(k) \leftarrow$ `Solve` LPR
end

Table 6: Using dual information for the Solomon test set - instances R1, RC1 and C1

Instances	MetaOptD		MetaOptD-A		MetaOptR		MetaOptR-A		MetaOptD+R		MetaOptD+R-A		MetaOpt		MetaOpt-A	
	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV
R101	1642.880	20	1642.880	20	1642.880	20	1642.880	20	1643.370	20	1644.230	20	1642.880	20	1642.880	20
R102	1474.110	18	1476.820	18	1472.810	18	1472.810	18	1474.430	18	1475.530	18	1472.810	18	1472.810	18
R103	1220.070	14	1221.200	14	1216.080	14	1217.020	14	1220.110	14	1220.470	14	1215.000	14	1216.740	14
R104	989.181	11	1005.590	11.33	985.918	11	996.300	11	981.919	11	999.120	11	976.764	11	980.540	11
R105	1360.780	15	1363.940	15	1360.780	15	1360.780	15	1360.780	15	1362.110	15	1360.780	15	1360.780	15
R106	1252.630	13	1255.690	13	1243.080	13	1246.930	13	1249.990	13	1252.410	13	1240.880	13	1243.210	13
R107	1082.130	12	1084.790	12	1075.200	11	1077.530	11	1079.190	12	1081.270	12	1074.310	11	1074.640	11
R108	958.874	10	965.370	10.67	956.047	10	963.960	10	957.987	10	964.050	10	948.388	10	961.280	10
R109	1154.350	13	1156.440	13	1152.820	13	1153.450	13	1158.330	13	1158.710	13	1151.840	13	1153.300	13
R110	1086.370	12	1093.230	12	1073.460	12	1077.560	12	1077.410	12	1089.190	12	1072.410	12	1078.320	12
R111	1068.690	12	1074.760	12	1057.490	12	1066.630	11.67	1057.550	12	1060.150	12	1054.950	12	1067.720	12
R112	967.960	10	970.790	10.33	964.732	10	972.530	10.66	978.427	11	984.420	11	962.303	10	973.170	10.33
RC101	1630.840	15	1645.220	15	1629.970	15	1632.840	15	1624.060	15	1643.790	15	1624.930	15	1632.660	15
RC102	1491.440	15	1492.640	15	1467.260	14	1478.280	14.33	1462.300	14	1483.570	14.33	1468.370	14	1485.600	14.67
RC103	1271.750	11	1287.020	11.67	1280.400	11	1290.640	11.33	1285.660	11	1288.120	11.67	1267.090	11	1267.240	11
RC104	1155.510	10	1158.260	10.33	1145.660	10	1151.790	10.67	1146.970	10	1152.920	10.33	1151.420	11	1159.160	11
RC105	1523.800	15	1537.200	15.33	1522.510	16	1522.510	16	1543.880	16	1562.670	16	1522.510	16	1522.510	16
RC106	1380.290	13	1385.550	13	1377.350	13	1386.110	13	1378.690	13	1388.950	13	1393.760	13	1394.480	13
RC107	1215.080	12	1217.300	12	1217.050	12	1225.730	12	1213.780	12	1214.910	12	1212.830	12	1214.760	12
RC108	1142.160	11	1147.920	11.33	1145.480	12	1150.870	11.67	1121.800	11	1133.020	11	1126.440	11	1132.680	11
C101	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10
C102	830.165	10	830.600	10	828.937	10	828.937	10	828.937	10	829.870	10	828.937	10	828.937	10
C103	828.839	10	829.760	10	828.065	10	829.370	10	831.199	10	832.000	10	828.065	10	828.065	10
C104	826.723	10	838.600	10	825.649	10	830.150	10	827.451	10	833.140	10	825.649	10	834.580	10
C105	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10
C106	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10
C107	828.937	10	829.35	10	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10
C108	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10	828.937	10
C109	828.937	10	829.470	10	828.937	10	828.937	10	828.937	10	829.470	10	828.937	10	828.937	10

Table 7: Using dual information for the Solomon test set - instances R2, RC2 and C2

Instances	MetaOptD		MetaOptD-A		MetaOptR		MetaOptR-A		MetaOptD+R		MetaOptD+R-A		MetaOpt		MetaOpt-A	
	TD	**NV**	**TD**	**NV**	**TD**	**NV**	**TD**	**NV**	**TD**	**NV**	**TD**	**NV**	**TD**	**NV**	**TD**	**NV**
R201	1157.870	8	1161.730	7.67	1158.230	8	1160.530	7.67	1196.270	8	1202.590	7.33	1155.710	9	1156.930	8.67
R202	1060.080	7	1062.700	6.67	1054.440	7	1065.680	6	1076.480	7	1085.220	6	1057.610	6	1065.030	6.67
R203	897.065	6	902.430	5.67	890.662	6	895.060	6	924.131	6	937.880	5.33	891.709	6	897.422	6
R204	759.352	5	767.050	5	756.179	5	760.260	5	821.800	4	835.670	3.67	755.582	4	760.023	4.67
R205	964.250	6	969.840	6	959.787	6	963.000	6	1006.300	6	1022.240	4.67	958.450	6	960.770	6
R206	919.095	5	922.560	5	904.370	5	908.040	5	938.338	4	960.860	4	899.741	5	903.284	5
R207	837.759	4	852.600	4.67	821.303	4	826.700	4	876.170	4	909.210	3.33	810.507	4	820.510	4
R208	754.243	4	756.900	3.33	728.130	4	732.230	3.67	771.307	3	793.710	3	738.044	4	741.660	4
R209	867.839	5	874.400	5	864.807	5	865.730	5	872.281	5	901.270	4.67	862.990	5	865.276	5
R210	943.950	5	947.510	5.33	925.976	6	930.450	6	985.902	4	999.680	4	922.305	6	926.150	6
R211	775.414	4	776.690	4.67	764.579	4	771.890	4.33	809.990	4	814.420	4.67	764.650	4	771.128	4.33
RC201	1270.090	9	1275.650	8.67	1276.430	8	1277.660	8	1278.170	9	1280.710	9	1272.490	8	1278.690	8.67
RC202	1110.550	8	1113.950	8	1106.650	8	1108.740	7.67	1115.700	8	1120.660	8	1102.730	8	1106.850	8
RC203	940.876	5	948.500	5	940.820	5	942.640	5	950.598	5	957.070	5	938.428	5	943.740	5.33
RC204	819.846	4	827.920	3.67	806.834	4	912.960	4.33	823.289	5	833.260	4.67	797.498	4	809.880	4
RC205	1176.980	8	1180.420	8	1164.980	8	1166.000	7.67	1167.350	7	1181.120	7.67	1162.210	7	1164.430	7
RC206	1089.990	5	1092.340	5.33	1074.580	7	1077.190	6.33	1078.930	7	1085.860	6.33	1068.670	6	1073.920	5.67
RC207	977.283	6	985.780	6	969.054	6	979.930	6	973.873	6	979.760	6	978.605	6	990.640	6.33
RC208	792.944	5	804.590	5.00	787.855	5	794.530	4.67	801.145	5	810.310	5	787.972	5	791.778	5
C201	642.222	3	642.222	3	591.557	3	591.557	3	642.222	3	642.222	3	591.557	3	591.557	3
C202	624.640	3	650.800	3	591.557	3	591.557	3	638.415	3	642.040	3	591.557	3	591.557	3
C203	615.579	3	616.380	3	604.549	3	605.400	3	615.423	3	616.090	3	591.173	3	601.060	3
C204	613.658	3	626.250	3	606.133	3	608.260	3	609.467	3	616.160	3	602.658	3	608.100	3
C205	615.976	3	629.530	3	592.573	3	593.810	3	626.899	4	635.390	4	590.190	3	591.618	3
C206	610.038	3	629.190	3	591.343	3	591.740	3	628.278	3	632.830	3.67	590.878	3	591.882	3
C207	608.031	3	622.880	3	591.263	3	592.450	3	602.537	3	624.410	3	589.860	3	590.500	3
C208	631.443	3	636.600	3	589.949	3	591.040	3	629.348	3	638.360	3	589.307	3	590.600	4
Average*	**992.952**	**8.661**	**998.813**	**8.708**	**984.961**	**8.696**	**990.255**	**8.673**	**998.901**	**8.661**	**1006.925**	**8.613**	**983.215**	**8.643**	**987.226**	**8.720**

*Average values over all instances in Table 6 and Table 7.

2. Review on Performance Profiles

Dolan and Moré (2002) propose the performance profiles to evaluate and compare the performances of a set of solvers on a set of test instances. We also use this tool to evaluate the performance of MetaOpt and its extensions in comparison with other algorithms.

Suppose that we have a set of algorithms denoted by S and a set of instances denoted by I whose cardinalities are n_s and n_i, respectively. Our performance measure is the total distance. For each instance $i \in I$ and algorithm $s \in S$, $t_{i,s}$ is defined as the total distance found for instance i by algorithm s. The performance on instance i by algorithm s is compared to the best performance among all algorithms applied on the same problem. To plot the performance profile figures, we follow these steps: First, the performance ratio, given by

$$r_{i,s} = \frac{t_{i,s}}{\min(t_{i,s} : s \in S)},$$

is computed. Then, to assess the overall performance of an algorithm we compute

$$\rho_s(\tau) = \frac{1}{n_i}|\{i \in I : r_{i,s} \leq \tau\}|,$$

where $\tau \in \mathbb{R}$ and $|\cdot|$ denotes the cardinality of a set. The value $\rho_s(\tau)$ shows the probability for algorithm $s \in S$ such that a performance ratio $r_{i,s}$ is within a factor τ of the best possible ratio. The function ρ_s is the (cumulative) distribution function for the performance ratio. Therefore, the value of $\rho_s(1)$ is the probability that the algorithm s dominates the rest of the algorithms.

References

Dolan, E. D. and J. J. Moré. 2002. "Benchmarking Optimization Software with Performance Profiles." *Math. Programming* 91:201–213.

Potvin, J.Y., T. Kervahut, B.L. Garcia and J.M. Rousseau. 1996. "The Vehicle Routing Problem with Time Windows Part I: Tabu Search." *INFORMS J. Comput.* 8(2):158–164.