MD-mfcmTSP algorithm

Panagiotis Zachos

May 2024

1 Pseudocode for the MD-mfcmTSP

At the end of Algorithm 1, a local optimization function (3) is called which in addition to moving nodes in different places in the same route, also moves nodes between routes of different depots and different vehicle types.

Algorithm 1: MD-mfcmTSP heuristic

Input: $G_T, G_M, ..., G_D$
Output: $Sol = \{Sol^i = \{R_T^i, R_M^i, ..., R_D^i\}, Sol^{i+1}, ..., Sol^m\}$ for each $i \in D$

Create clusters K^i of customer nodes for each depot $d^i \in D$
by assigning each customer to the closest possible depot
for *each* $d^i \in D$ **do**
  Call *Initialization*(d^i, K^i)
  while $(M_T^i > M_M^i \parallel M_T^i > M_D^i)$ && $stop \neq true$ **do**
    $diff_M = M_T^i - M_M^i$
    $diff_D = M_T^i - M_D^i$
    if $diff_M \geq diff_D$ **then**
      $vt = M$
      $cap =$ Motorbike's capacity
    else
      $vt = D$
      $cap = 1$
    end if
    $M_{min} = M_T^i$
    $r_{best} = \emptyset$
    for $j = 1$ *to* $|R_T^i| - cap$ **do**
      $successive\_nodes = \emptyset$
      $load = 0$
      while $load + v_j^{demand} \leq cap$ && $v_j \in G_{vt}$ **do**
        $successive\_nodes$ += v_j
      end while
      if $|successive\_nodes| == cap$ **then**
        $r_{new} = R_T^i[0] + \{successive\_nodes\} + R_T^i[0]$
        $R'^i_{vt} = R^i_{vt} + r_{new}$
        $M'_{vt} = R'^i_{vt}$'s makespan
        $R'^i_T = R^i_T - \{successive\_nodes\}$
        $M'_T = R'^i_T$'s makespan
        $M_{max} = MAX(M'_T, M'_{vt})$
        if $M_{max} < M_{min}$ **then**
          $M_{min} = M_{max}$
          $r_{best} = r_{new}$
        end if
        $r_{new} = \emptyset$
      end if
      j += 1
    end for
    if $M_{min} < R_T^i$ **then**
      $R_T^i = R_T^i - \{r_{best}^{customers}\}$
      $M_T = R_T^i$'s makespan
      R_{vt}^i += r_{best}
      $M_{vt} = R_{vt}^i$'s makespan
      Call *local_optimization*(R_T^i)
      Call *local_optimization*(R_{vt}^i)
    else
      $stop = true$
    end if
  end while
  $Sol^i = \{R_T^i, R_M^i, ..., R_D^i\}$
end for
$M_T = MAX(M_T^i, M_T^{i+1}, ..., M_T^m)$
$M_M = MAX(M_M^i, M_M^{i+1}, ..., M_M^m)$
$M_D = MAX(M_D^i, M_D^{i+1}, ..., M_D^m)$
$M_{total} = MAX(M_T, M_M, ..., M_D)$

Algorithm 2: Initialization(d^i, K^i)

while $\{K^i\} \cap \{G_T\} \neq \emptyset$ **do**
  R_T^i += $NearestNeighbour(\{K^i\} \cap \{G_T\})$
end while
$M_T^i = R_T^i$'s makespan
$v_{free} = \{K^i\} - \{G_T\}$
if $v_{free} = \emptyset$ **then**
  return R_T^i
else
  while $v_{free} \neq \emptyset$ **do**
    if $M_T - M_M \geq M_T - M_D \parallel G_D = \emptyset$ **then**
      R_M^i += $NearestNeighbour(\{K^i\} \cap \{G_M\})$
      $v_{free} = v_{free} - \{R_M^i\}$
      $M_M^i = R_M^i$'s makespan
    else
      R_D^i += *closest*$(\{K^i\} \cap \{G_D\})$
      $M_D^i = R_D^i$'s makespan
    end if
  end while
end if
return Sol^i

Algorithm 3: $local\_opt\_full(Sol, n_{max})$

Call *vt_optimization*$(Sol, n_{max} = 2)$
for *each vt* **do**
  for *each* $i \in D$ **do**
    Call *local_optimization*$(R_{vt}^i, n_{max} = 2)$
  end for
  $M_{vt} = MAX(R_{vt}^i, R_{vt}^{i+1}, ..., R_{vt}^m)$
  Call *mutual_optimization*$(R_{vt}, n_{max} = 2)$
end for
Call *vt_optimization*$(Sol, n_{max} = 2)$

Algorithm 4: $local\_optimization(r, n_{max})$

1 **for** $n = 1$ *to* n_{max} **do**
2   **for each** *combination of n successive nodes on the route*
3   move the node(s) to a different place on the same route
4   evaluate the new route
5   **if** *this route is better than the original and all constraints are satisfied* **then**
6   replace the original route with the new one
7   continue in point 3 unless all possible places in the route have been evaluated
8   **end for**
9 **end for**
10 **return** r

Algorithm 5: *mutual_optimization*(R_{vt}, n_{max})

1 **for** $n = 1$ *to* n_{max} **do**
2   **for each** *possible pair of depots* $c1$ *and* $c2$
3   **for each** *combination of* n *successive nodes in the route of c1*
4   remove the nodes from the route of c1 and insert them into c2
5   evaluate the newly-created routes
6   **if** $MAX(|{R'}^{c1}_{vt}|, |{R'}^{c2}_{vt}|) < MAX(|R^{c1}_{vt}|, |R^{c2}_{vt}|)$ *and all constraints are satisfied* **then**
7   replace the original routes with the new ones
8   continue in point 4 unless all possible places in c2 have been evaluated
9   **end for**
10   **end for**
11 **end for**
12 **return** R_{VT}

Algorithm 6: *vt_optimization*(Sol, n_{max})

1 **for** $n = 1$ *to* n_{max} **do**
2   **for each** *depot* $i \in D$
3   **for each** *possible pair of vehicle types* $t1, t2 \in VT$
4   **for each** *combination of* n *successive nodes in* R^i_{t1}
5   remove the nodes from R^i_{t1} and insert them in R^i_{t2}
6   **if** $MAX(|{R'}^i_{t1}|, |{R'}^i_{t2}|) < MAX(|R^i_{t1}|, |R^i_{t2}|)$ *and all constraints are satisfied* **then**
7   replace the original routes with the new ones
8   continue in point 5 unless all possible places in R^i_{t2} have been evaluated
9   **end for**
10   **end for**
11   **end for**
12 **end for**
13 **return** Sol

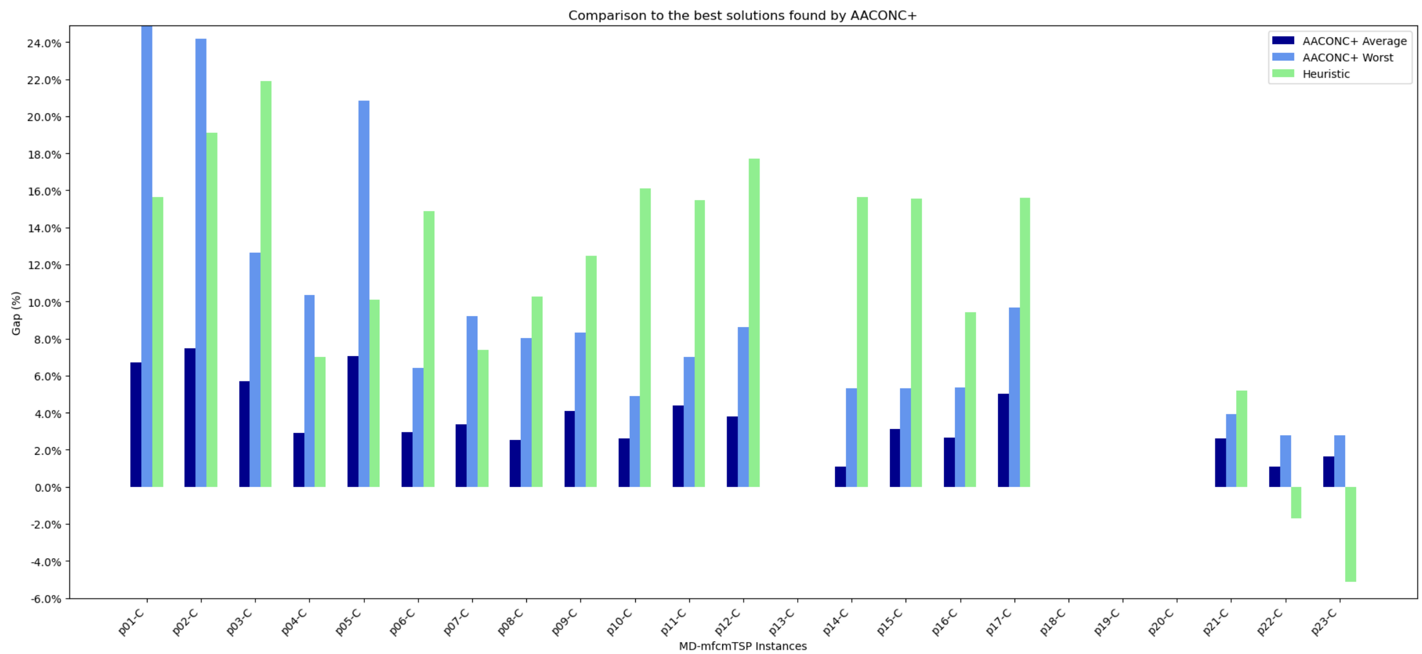
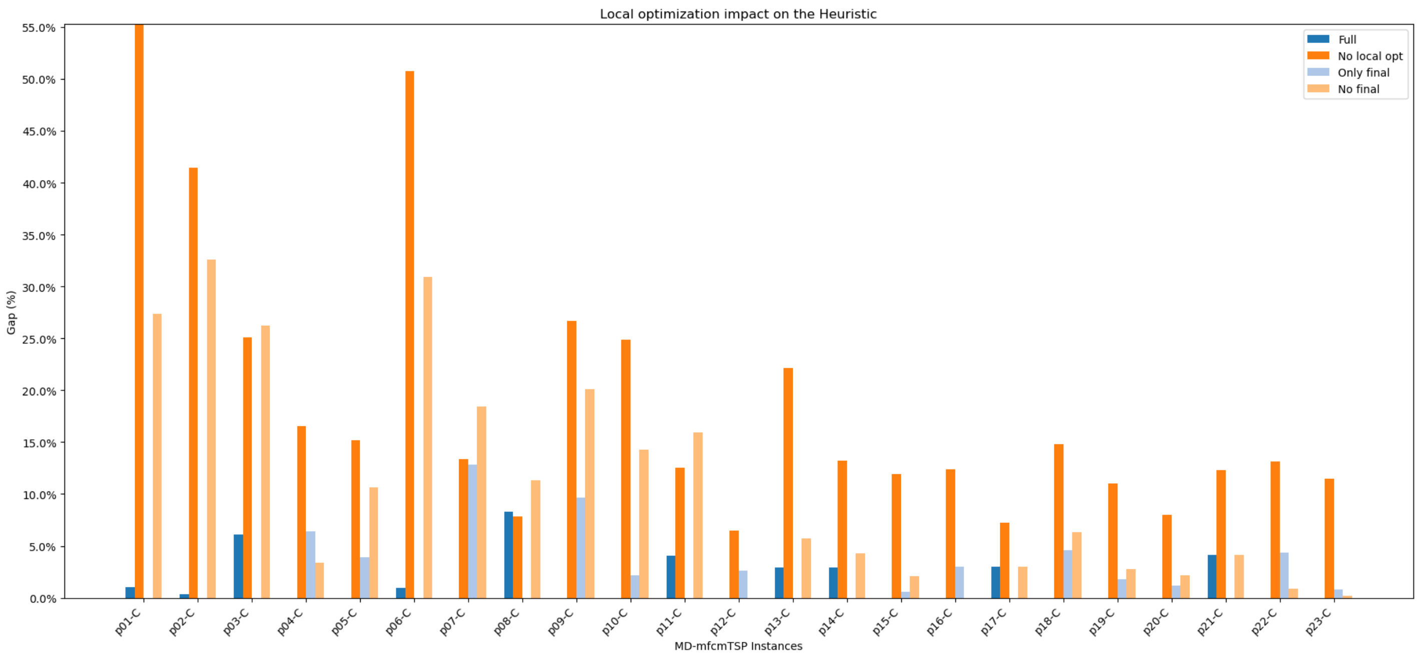

Comparison to the best solutions found by AACONC+
AACONC+ Average
AACONC+ Worst
Heuristic
Gap (%)
MD-mfcmTSP Instances

Table 1: Local optimization impact on heuristic

Instance	**Best**	**Full Local Opt**	**gap(%)**	**No Local Opt**	**gap(%)**	**No Swap Local Opt**	**gap(%)**	**No Final Local Opt**	**gap(%)**
p01-C	215.15	217.42	1.06	334.04	55.26	**215.15**	0.00	173.96	27.33
p02-C	218.4	219.20	0.37	308.83	41.41	**218.40**	0.00	289.58	32.59
p03-C	202.43	214.84	6.13	253.18	25.07	**202.43**	0.00	255.52	26.23
p04-C	668.81	**668.81**	0.00	779.34	16.53	711.80	6.43	691.62	3.41
p05-C	630.78	**630.78**	0.00	726.62	15.19	655.4	3.90	697.86	10.63
p06-C	431.32	435.42	0.95	650.32	50.77	**431.32**	0.00	564.79	30.94
p07-C	309.48	**309.48**	0.00	350.77	13.34	349.19	12.83	366.51	18.43
p08-C	3079.58	333.93	8.26	3321.63	7.86	**3079.58**	0.00	3428.15	11.32
p09-C	1917.82	**1917.82**	0.00	2429.24	26.67	2102.98	9.65	2303.70	20.12
p10-C	1493.13	**1493.13**	0.00	1864.23	24.85	1525.37	2.16	1706.14	14.27
p11-C	1101.33	1145.75	4.03	1239.36	12.53	**1101.33**	0.00	1276.59	15.91
p12-C	1273.77	**1273.77**	0.00	1356.09	6.46	1307.11	2.62	**1273.77**	0.00
p13-C	1191.96	1226.63	2.91	1455.87	22.14	**1191.96**	0.00	1259.97	5.71
p14-C	1248.53	1284.73	2.90	1413.87	13.24	**1248.53**	0.00	1302.34	4.31
p15-C	1347.67	**1347.67**	0.00	1508.50	11.93	1355.08	0.55	1375.95	2.10
p16-C	1289.29	**1289.29**	0.00	1448.53	12.35	1328.00	3.00	**1289.29**	0.00
p17-C	1282.38	1320.91	3.00	1375.55	7.27	**1282.38**	0.00	1320.91	3.00
p18-C	1260.24	**1260.24**	0.00	1446.92	14.81	1317.54	4.55	1339.83	6.32
p19-C	1266.92	**1266.92**	0.00	1406.13	10.99	1289.95	1.82	1301.62	2.74
p20-C	1323.1	**1323.10**	0.00	1429.00	8.00	1338.42	1.16	1351.39	2.14
p21-C	1309.14	1363.18	4.13	1470.03	12.29	**1309.14**	0.00	1363.18	4.13
p22-C	1295.67	**1295.67**	0.00	1465.74	13.13	1351.96	4.34	1306.6	0.84
p23-C	1252.36	**1252.36**	0.00	1396.11	11.48	1262.65	0.82	1254.42	0.16
AVG	1113.44	**1134.39**	**1.46**	1279.56	18.85	1138.07	2.34	1199.72	10.54

Table 2: AACONC+ Results

Instance	Best	Average	gap(%)	Worst	gap(%)	Average time(s)
p01-C	188.00	200.66	6.73	234.78	24.88	73
p02-C	184.01	197.78	7.48	228.47	24.16	47
p03-C	176.26	186.33	5.71	198.56	12.65	202
p04-C	624.93	642.98	2.89	689.6	10.35	1173
p05-C	573.03	613.47	7.06	692.44	20.84	884
p06-C	379.10	390.34	2.96	403.39	6.41	1316
p07-C	288.24	297.91	3.35	314.78	9.21	798
p08-C	3023.77	3100.23	2.53	3265.97	8.01	1635
p09-C	1705.03	1774.86	4.10	1846.92	8.32	3391
p10-C	1286.28	1319.78	2.60	1349.47	4.91	3549
p11-C	992.16	1035.63	4.38	1061.7	7.01	3600
p12-C	1082.12	1123.30	3.81	1175.39	8.62	446
p13-C	1099.02	1135.30	3.30	1191.96	8.46	491
p14-C	1111.12	1123.33	1.10	1170.17	5.31	423
p15-C	1166.08	1202.57	3.13	1228.27	5.33	2471
p16-C	1178.51	1209.60	2.64	1241.67	5.36	2146
p17-C	1142.74	1199.88	5.00	1253.47	9.69	2628
p18-C	1231.96	1263.90	2.59	1296.42	5.23	3368
p19-C	-	-	-	-	-	
p20-C	-	-	-	-	-	
p21-C	1296.19	1330.11	2.62	1347.09	3.93	3600
p22-C	1317.94	1332.07	1.07	1354.61	2.78	3600
p23-C	1319.93	1341.50	1.63	1356.62	2.78	3600
Average	1036.46	1068.44	-	1111.66	-	1904


Local optimization impact on the Heuristic
Full
No local opt
Only final
No final
Gap (%)
55.0%
50.0%
45.0%
40.0%
35.0%
30.0%
25.0%
20.0%
15.0%
10.0%
5.0%
0.0%
p01-C
p02-C
p03-C
p04-C
p05-C
p06-C
p07-C
p08-C
p09-C
p10-C
p11-C
p12-C
p13-C
p14-C
p15-C
p16-C
p17-C
p18-C
p19-C
p20-C
p21-C
p22-C
p23-C
MD-mfcmTSP Instances