Online Supplement for “An Improved Multi-directional Local Search Algorithm for the Multi-objective Consistent Vehicle Routing Problem” by “Kunlei Lian, Ashlea B. Milburn and Ronald L. Rardin”

1 MDLS workflow

Algorithm 1 shows the framework of multi-directional local search. Algorithm 2 shows the workflow of the initial solution generation process.

Algorithm 1 Multi-directional local search (Tricoire, 2012)

```
1: Input: A set of non-dominated solutions  $\mathcal{F}$ 
2: repeat
3:    $x \leftarrow \text{select\_a\_solution}(F)$ 
4:    $G \leftarrow \emptyset$ 
5:   for  $k \leftarrow 1$  to  $K$  do
6:      $G \leftarrow G \cup \{LS_k(x)\}$ 
7:   end for
8:    $\text{update}(F, G)$ 
9: until stopping criterion is met
10: Output:  $\mathcal{F}$ 
```

2 Initial solution generation procedure

Algorithm 2 shows the initial generation scheme.

3 Detailed coverage comparison results

Tables 1, 2 and 3 give the pairwise coverage comparison between different multi-objective algorithms.

Algorithm 2 Solution generation scheme

```
1: Input: customer list  $C$  and customer neighbor size  $\alpha$ 
2: randomly shuffle the list  $C$ 
3: create a solution  $s$  with an empty route for each day  $d \in D$ 
4: while  $C$  is not empty do
5:   take the first customer  $c$  in  $C$ , let  $C \leftarrow C \setminus \{c\}$ 
6:   for  $d \leftarrow 1$  to  $D$  do
7:     if  $w_{cd} > 0$  then
8:       identify neighbor routes of  $c$  from existing routes on day  $d$  regarding to  $\alpha$ 
9:       put all neighbor routes of  $c$  to  $R_d$ 
10:      if  $R_d = \emptyset$  then
11:        create a new route  $r$ , let  $R_d \leftarrow r$ 
12:      end if
13:    end if
14:  end for
15:  determine the driver  $r^*$  who works on most days over  $D$ 
16:  for  $d \leftarrow 1$  to  $D$  do
17:    if  $w_{cd} > 0$  then
18:      if there exists route  $r \in R_d$  whose driver is  $r^*$  then
19:        insert customer  $c$  to route  $r^*$  with least travel cost increase
20:      else
21:        insert customer  $c$  to route  $r'$  on day  $d$  with least travel cost increase
22:      end if
23:    end if
24:  end for
25: end while
26: Output: solution  $s$ 
```

		NSGAII	NSGAIII	MOEA/D			NNIA	SPEA2	MDLS	IMDLS
				WS	TCH	PBI				
MOEA/D	NSGAII	0.00%	42.84%	36.29%	37.05%	75.56%	60.50%	47.90%	99.15%	22.60%
	NSGAIII	41.75%	0.00%	39.81%	39.71%	76.54%	58.02%	45.94%	99.39%	24.11%
	WS	22.83%	21.35%	0.00%	20.73%	62.21%	31.97%	22.52%	89.19%	12.00%
	TCH	42.07%	42.70%	39.70%	0.00%	78.12%	54.87%	45.52%	99.24%	22.92%
	PBI	6.99%	6.48%	11.81%	6.33%	0.00%	13.59%	8.70%	78.88%	5.06%
	NNIA	27.73%	27.11%	29.67%	24.79%	63.98%	0.00%	33.20%	98.34%	15.56%
	SPEA2	40.36%	36.49%	38.63%	33.92%	72.15%	55.33%	0.00%	98.54%	20.92%
	MDLS	0.04%	0.00%	0.70%	0.27%	19.80%	0.37%	0.05%	0.00%	0.26%
	IMDLS	58.37%	56.25%	53.08%	47.40%	71.27%	71.80%	61.78%	98.65%	0.00%

Table 1: Coverage results for $Group_{0.5}$

4 Description of comparison algorithms

Algorithm 3 describes the workflow of NSGAII. NSGAIII shares similar workflow with NSGAII except steps 16-20 in Algorithm 3. The steps unique for NSGAIII are given in Algorithm 4. Algorithms 5, 6, 7 show the workflows for MOEA/D, NNIA and SPEA2, respectively.

Algorithm 3 Non-dominated sorting genetic algorithm II: NSGAII (Deb et al., 2002)

```
1: Input: population size  $N$ 
2: Set  $t = 0$ .
3: Generate an initial population  $P_t$  and assign dominance rank to each solution
4: Create a offspring population  $Q_t$  of size  $N$ 
5: repeat
6:    $S_t = \emptyset, i = 1$ 
7:   Combine  $P_t$  and  $Q_t$  to get  $R_t = P_t \cup Q_t$ 
8:    $(F_1, F_2, \dots) = \text{Non-dominated-sort}(R_t)$ 
9:   repeat
10:     $S_t = S_t \cup F_i$  and  $i = i + 1$ 
11:   until  $|S_t| \geq N$ 
12:   Last front to be included  $F_l = F_i$ 
13:   if  $|S_t| = N$  then
14:      $P_{t+1} = S_t$ 
15:   else
16:      $P_{t+1} = \cup_{j=1}^{l-1} F_j$ 
17:     Compute crowding distance of  $F_l$  and sort  $F_l$  in non-increasing order
18:     repeat
19:        $P_{t+1} = P_{t+1} \cup s_1 \in F_l, F_l = F_l \setminus s_1$ 
20:     until  $|P_{t+1}| = N$ 
21:   end if
22:   Create offspring population  $Q_{t+1}$  based on  $P_{t+1}$ 
23:    $t = t + 1$ 
24: until stopping criterion is met
25: Output:  $Q_t$ 
```

Algorithm 4 Unique steps in NSGAIII compared with NSGAII (Deb and Jain, 2014)

```
16:  $P_{t+1} = \cup_{j=1}^{l-1} F_j$ 
17: Points to be chosen from  $F_l : K = N - |P_{t+1}|$ 
18: Normalize objectives and create reference set  $Z^r$ 
19: Associate each member  $s$  of  $S^t$  with a reference point
20: Compute niche count of reference point  $j \in Z^r : \rho_j = \sum_{s \in S_t/F_l} ((\pi(s) = j)?1 : 0)$ 
21: Choose  $K$  members one at a time from  $F_l$  to construct  $P_{t+1}$ 
```

Algorithm 5 Multi-objective evolutionary algorithm based on decomposition: MOEA/D (Qingfu and Hui, 2007)

```

1: Input:  $N$ : the number of subproblems considered in MOEA/D;  $\lambda^1, \dots, \lambda^N$ : a uniform spread
   of  $N$  weight vectors;  $T$ : the number of the weight vectors in the neighborhood of each weight
   vector
2: Initialize an external population  $EP = \emptyset$ 
3: Determine the  $T$  closest weight vectors to each weight vector based on Euclidean distance
4: For each  $i = 1, \dots, N$ , set  $B(i) = \{i_1, \dots, i_T\}$ 
5: Create initial population  $x^1, \dots, x^N$ . Set  $FV^i = F(x^i)$  as the scalarized objective value
6: Initialize  $z = (z_1, \dots, z_m)$  as the best value found so far for each objective
7: repeat
8:   for  $i = 1, \dots, N$  do
9:     Randomly select  $k, l \in B(i)$  and generate a new solution  $y$  from  $x^k$  and  $x^l$ 
10:    Apply improvement heuristic on  $y$  to get  $y'$ 
11:    For  $j = 1, \dots, m$ , set  $z_j = f_j(y')$  if  $z_j < f_j(y')$ 
12:    For  $j \in B(i)$ , set  $x^j = y'$  and  $FV^j = F(y')$  if  $g(y'|\lambda^j, z) \leq g(x^j|\lambda^j, z)$ 
13:    Update  $EP$ 
14:   end for
15: until stopping criterion is met
16: Output: External population  $EP$ 

```

Algorithm 6 Non-dominated neighbor immune algorithm: NNIA (Gong et al., 2008)

```

1: Input:  $n_D$ : dominant population size;  $n_A$ : active population size;  $n_C$ : clone population size
2: Create initial population  $B_0$  with size  $n_D$ 
3: Set  $D_0 = \emptyset, A_0 = \emptyset, C_0 = \emptyset, t = 0$ 
4: repeat
5:   Identify dominant solutions in  $B_t$  and copy them to  $DT_{t+1}$ .
6:   if  $|DT_{t+1}| < n_D$  then
7:      $D_{t+1} = DT_{t+1}$ 
8:   else
9:     Compute crowding distances for  $DT_{t+1}$  and sort it in descending order
10:    Choose the first  $n_D$  solutions in  $DT_{t+1}$  to form  $D_{t+1}$ 
11:   end if
12:   if  $|D_t| \leq n_A$  then
13:      $A_t = D_t$ 
14:   else
15:     Compute the crowding distances for  $D_t$  and sort it in descending order
16:     Choose the first  $n_A$  solutions in  $D_t$  to form  $A_t$ 
17:   end if
18:   Apply proportional cloning on  $A_t$  to get  $C_t$ 
19:   Perform recombination and hypermutation on  $C_t$  to get  $C'_t$ 
20:   Get  $B_t$  by combining  $C'_t$  and  $D_t$ , set  $t = t + 1$ 
21: until stopping criterion is met
22: Output  $D_t$ 

```

	NSGAII	NSGAIII	MOEA/D			NNIA	SPEA2	MDLS	IMDLS
			WS	TCH	PBI				
NSGAII	0.00%	42.06%	46.45%	46.21%	62.49%	63.35%	58.23%	99.64%	27.95%
NSGAIII	41.27%	0.00%	45.00%	43.17%	59.84%	63.31%	57.25%	99.59%	29.10%
WS	14.81%	15.19%	0.00%	23.79%	44.67%	27.55%	21.88%	92.71%	11.73%
TCH	30.44%	28.74%	37.28%	0.00%	60.90%	45.14%	38.34%	99.02%	21.11%
PBI	5.83%	3.06%	10.40%	9.22%	0.00%	9.38%	6.81%	72.38%	4.60%
NNIA	25.22%	23.28%	32.03%	25.90%	41.20%	0.00%	34.74%	99.48%	16.00%
SPEA2	26.56%	27.01%	36.51%	29.75%	52.05%	50.83%	0.00%	99.49%	20.67%
MDLS	0.07%	0.17%	0.46%	0.10%	4.56%	0.11%	0.14%	0.00%	0.14%
IMDLS	49.25%	49.03%	48.70%	41.27%	45.99%	65.08%	58.23%	98.22%	0.00%

Table 2: Coverage results for $Group_{0.7}$

	NSGAII	NSGAIII	MOEA/D			NNIA	SPEA2	MDLS	IMDLS
			WS	TCH	PBI				
NSGAII	0.00%	41.90%	55.57%	54.49%	52.49%	64.62%	54.98%	99.36%	44.80%
NSGAIII	37.17%	0.00%	53.21%	52.04%	48.94%	62.77%	53.55%	99.44%	41.21%
WS	17.13%	15.76%	0.00%	35.77%	38.23%	27.17%	21.87%	94.68%	18.44%
TCH	21.82%	18.71%	38.88%	0.00%	45.06%	34.43%	32.35%	97.05%	24.61%
PBI	12.67%	10.17%	17.10%	24.58%	0.00%	14.21%	13.63%	78.98%	12.38%
NNIA	21.27%	24.44%	40.90%	40.77%	41.68%	0.00%	36.69%	99.36%	28.04%
SPEA2	29.30%	31.27%	45.27%	42.64%	42.29%	47.11%	0.00%	98.17%	35.63%
MDLS	0.25%	0.21%	0.59%	0.94%	3.26%	0.21%	1.22%	0.00%	0.34%
IMDLS	34.82%	38.11%	45.09%	45.93%	39.26%	55.03%	49.31%	97.25%	0.00%

Table 3: Coverage results for $Group_{0.9}$

References

- K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *Evolutionary Computation, IEEE Transactions on*, 18(4):577–601, 2014. ISSN 1089-778X. doi: 10.1109/TEVC.2013.2281535.
- K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.
- Maoguo Gong, Licheng Jiao, Haifeng Du, and Liefeng Bo. Multiobjective immune algorithm with nondominated neighbor-based selection. *Evol. Comput.*, 16(2):225–255, 2008. p.
- Eckart Zit. Ler, Marco Laumanns, and Lothar Thiele. *SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization*, pages 19–26. CIMNE, Spain, 2002.
- Zhang Qingfu and Li Hui. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, 11(6):712–731, 2007. ISSN 1089-778X. doi: 10.1109/TEVC.2007.892759.

Algorithm 7 Strength Pareto Evolutionary Algorithm: SPEA2 (Ler et al., 2002)

- 1: **Input:** N : population size; $\bar{N}$: archive size; T maximum number of iterations
 - 2: Create initial population P_0 and empty archive $\bar{P} = \emptyset$, set $t = 0$
 - 3: Compute fitness values of solutions in P_t and $\bar{P}_t$
 - 4: Set $\bar{P}_{t+1} = P_t \cup \bar{P}_t$.
 - 5: **if** $|\bar{P}_{t+1}| > \bar{N}$ **then**
 - 6: Truncate $\bar{P}_{t+1}$ to size $\bar{N}$
 - 7: **else**
 - 8: Fill $\bar{P}_{t+1}$ to size $\bar{N}$
 - 9: **end if**
 - 10: Using binary tournament selection on $\bar{P}_{t+1}$ to fill the mating pool
 - 11: Apply recombination and mutation to the mating pool and set P_{t+1} as the resulting population
 - 12: $t = t + 1$, stop if $t \geq T$; otherwise, go to line 3
 - 13: **Output** $\bar{P}_t$
-

Fabien Tricoire. Multi-directional local search. *Computers & Operations Research*, 39(12):3089–3101, 2012. p.