

Supplementary Materials for TEVC-00434-2020 “Novel Memetic Algorithms with Smart Initialisation for Real-world Large-scale Waste Collection Problems”

Wenxing Lan, Ziyuan Ye, Peijun Ruan, Jialin Liu, *Senior Member, IEEE*, Peng Yang, *Member, IEEE*, and Xin Yao, *Fellow, IEEE*

In our paper “Novel Memetic Algorithms with Smart Initialisation for Real-world Large-scale Waste Collection Problems” [1] under review by the IEEE Transactions on Evolutionary Computation (manuscript id: TEVC-00434-2020), we have formulated the multi-depot multi-disposal-facility multi-trip capacitated vehicle routing problem (M3CVRP). To tackle the problem, we have proposed an heuristic-assisted smart initialisation (HaSI) method for generating feasible solutions, an extended local search (ELS) composed of three new region-constrained search operators and a non-greedy facility insertion method, and a novel memetic algorithm composed of HaSI and ELS, namely MAELS. In the paper, our approaches have been compared to the state-of-the-art Clarke and Wright (CW) Savings Algorithm, some state-of-the-art operators (2-opt, swap and insertion), greedy facility insertion method, local search and memetic algorithm that don’t use our proposed operators or methods, respectively.

To further verify our approaches, in the supplementary materials, we compare our approaches to some state-of-the-art methods, including a popular commercial solver, IBM ILOG CPLEX Optimizer (Section II), and a state-of-the-art memetic algorithm with competition (MAC) [2] (Section III).

I. CLARKE AND WRIGHT SAVINGS ALGORITHM

II. COMPARISON WITH CPLEX

We have used CPLEX to solve our real-world M3CVRP instance that has been described in the paper. However, due to the high dimensionality (3,000 collection sites), CPLEX solver is not able to solve our problem instance in a reasonable time. At least 12TB of memory and several days of runtime are required to find a feasible solution by the CPLEX solver. For comparison, smaller M3CVRP instances have been generated by randomly sampling tasks (i.e., collection sites) from our real-world problem instance. The size of generated instances, in terms of number of tasks, are summarised in the first column of Table I. CPLEX uses dynamically multiple threads. Therefore, for each problem size, the total CPU time consumed by CPLEX with multiple threads to find a solution are reported in Table I, as well as the solution quality. The quality of solutions found by our ELS and MAELS with initial population of high and low diversity (denoted as MAELS_High and MAELS_Low, respectively) in 30 independent trials, given the

Algorithm 1 Clarke and Wright Savings Algorithm [3]. $d_{i,j}$ denotes the Euclidean distance between two vertices i and j .

Input: A set of sites V_C , a depot p and their locations

Output: A solution with inserted depots and disposal facilities

- 1: Initial a savings list $\mathcal{L}$ as empty
 - 2: **for** $\forall$ customers $i, j \in V_C$ **do**
 - 3: Calculate *savings value* $s_{i,j}$ of edge (i, j) with

$$s_{i,j} = d_{p,i} + d_{j,p} - d_{i,j} \quad (1)$$
 - 4: Add the pair $\langle (i, j), s_{i,j} \rangle$ into $\mathcal{L}$
 - 5: **end for**
 - 6: $\mathcal{L}$ is sorted in the decreasing order of savings value
 - 7: **for** $\langle (i, j), s_{i,j} \rangle \in \mathcal{L}$ **do**
 - 8: Include edge (i, j) in a route if no route constraint will be violated through the inclusion of i and j
 - 9: **end for**
 - 10: **for** $\forall$ non-routed site $c \in V_C$ **do**
 - 11: Set its route as $p \rightarrow c \rightarrow p$
 - 12: **end for**
-

same amount of CPU time as CPLEX used, are provided in the same table. The way of generating initial population of high and low diversity is the same as presented in Section TODO of the main manuscript.

It’s observed that with the increment of the number of tasks, the time cost for CPLEX to find a solution increases rapidly. More importantly, when solving an instance with 60 tasks, CPLEX consumed more than 10 hours in real time to find the solution. When solving an instance with 70 tasks, CPLEX failed to find a solution after 50 hours in real time with an *Intel(R) Xeon(R) CPU E5-2650 v4* working at 2.20GHz on Linux 3.10.0. In conclusion, CPLEX dominates when solving small instances with no more than 60 tasks. However, it’s not able to solve instances with 70 tasks or more in a reasonable time. Our ELS and MAELS initialised with solutions of high diversity is competitive when solving the real-world M3CVRP instance with 3,000 tasks as previously shown in Tables VII

TABLE I

COMPARISON BETWEEN CPLEX, OUR ELS AND MAELS ON SMALLER M3CVRP INSTANCES GENERATED FROM REAL DATA. THE DEPOTS AND DISPOSAL FACILITIES REMAIN UNCHANGED, THUS $|V_P| = 3$ AND $|V_D| = 3$. THE BEST (“BEST”) AND AVERAGED (“AVG”) DISTANCES OF RECOMMENDED SOLUTIONS OVER 30 INDEPENDENT OPTIMISATION TRIALS OF ELS ARE REPORTED, AS WELL AS THE STANDARD DEVIATION (“STD”).

Instance $ V_C $	CPLEX		ELS				MAELS_High				MAELS_Low			
	CPU time (mins)	Distance (km)	CPU time (mins)	Distance (km)			CPU time (mins)	Distance (km)			CPU time (mins)	Distance (km)		
				BEST	AVG	STD		BEST	AVG	STD		BEST	AVG	STD
30	2.0	232.2	2.0	237.0	250.7	12.3	2.0	235.6	236.9	0.6	2.0	235.6	237.0	0.5
40	49.3	322.1	49.3	331.0	350.4	10.7	49.3	330.5	332.8	2.8	49.3	331.0	341.0	5.5
50	379.8	350.5	379.8	375.1	382.8	3.1	379.8	351.0	367.0	6.4	379.8	351.6	369.7	6.6
60	1737.6	360.0	1737.6	388.6	395.7	5.5	1737.6	380.9	385.6	2.5	1737.6	382.8	386.6	1.7
70	No solution after 50 hours		2000.0 ¹	426.8	445.5	16.0	2000.0	423.0	432.5	4.4	2000.0	427.0	436.4	3.9
3,000	–	–	240 ²	5568.5	5628.8	22.9	240	5505.4	5628.5	29.7	240	5660.8	5719.9	25.3

and VIII of our paper.

III. COMPARISON WITH MAC

To our best knowledge, no existing work has considered or formalised such a capacitated vehicle routing problem with multi-depot, multi-disposal-facility, multi-trip and time constraint at the same time. As a consequence, there is no existing method that can be directly applied to our problem. Therefore, we have taken the state-of-the-art memetic algorithm with competition (MAC) [2] as the closest baseline algorithm. MAC was designed for capacitated green vehicle routing problems which involve multiple fuel stations [2], similar to multiple disposal facilities in our case but there is one single depot. We adapted MAC to the M3CVRP and perform comparative study on our real-world M3CVRP instance. Since MAC has its own solution initialisation method, we compare it to HaSI, our initialisation method, first in Section III-A, and then compare MAC, MAC using HaSI with our ELS and MAELS in Section III-B.

The experimental details are the same as the ones reported in our paper, reminded as follows. All programs have been written in *C++ 11* and executed on an *Intel(R) Xeon(R) CPU E5-2650 v4* CPU working at 2.20GHz on Linux 3.10.0, using a single thread. All the experiments have been run on the same machine. Experiments that involve randomness have been repeated 30 times independently. Statistical tests have been performed for those experiments.

A. Initialisation Phase

In our paper, we have proposed heuristic-assisted smart initialisation (HaSI) method for generating feasible solutions, which can be used as initial solutions of meta-heuristics, such as local search and memetic algorithms. HaSI is compared to the initialisation method of MAC [2], which is a stochastic algorithm. Consequently, 30 independent trials have been performed with the initialisation phase in MAC. Consequently, Table II reports the best and worst solutions generated in the 30

independent trials of MAC’s initialisation method. Although HaSI is deterministic, it can generate different feasible solutions by varying the number of available vehicles at each depot. More details can be found in the paper (Section IV). For comparison, the experimental results of HaSI, M3CWSA-I and M3CWSA-II in Table IV of the main manuscript are reminded in Table II.

TABLE II

FEASIBLE SOLUTIONS GENERATED BY HaSI AND THE INITIALISATION PHASE IN MAC. N_V STANDS FOR THE NUMBER OF USED VEHICLES. RT STANDS FOR THE RUNNING TIME OF THE ALGORITHM IN SECONDS.

Algorithm	Solution	Distance (km)	N_V	RT (s)
HaSI	Best	6,347.3	29	< 1
	Worst	6,664.8	30	< 1
MAC	Best	6,687.1	20	< 1
	Worst	7,3057.9	21	< 1

According to Table II, the total distance of the best and worst solutions generated by HaSI are both shorter than the corresponding ones generated by MAC’s initialisation method. In particular, the distance of the worst solution generated by HaSI is about 270 kilometers shorter than the averaged distance of the 30 solutions generated by MAC (6,941.5). Moreover, we generate 300 solutions with MAC’s initialisation method and obtain an average of 6,931.8 kilometers with standard deviation 134.9, still significantly longer than the solutions generated by HaSI. Although the solutions generated by HaSI use more vehicles than the ones by MAC, it’s acceptable since the objective of M3CVRP is minimising the total travel distance. But for problems that consider the minimisation of the number of used vehicles, MAC’s initialisation method is better. In short, HaSI generates better solutions for our real-world M3CVRP instance in terms of the total travel distance than the initialisation method of the state-of-the-art MAC and the adaptations of Clarke and Wright Savings Algorithm (referred to as M3CWSA-I and M3CWSA-II in Table II).

B. Optimisation Phase

1) *A direct comparison:* During the optimisation phase of MAC, there are two important hyperparameters, the size of population and the number of executions in local intensification phase, denoted as PS and LS in [2]. According to the

¹Each experiment is repeated 30 times which takes a lot of time. Therefore, we provided 2000 minutes CPU time instead of 3000 as optimisation budget.

²When optimising the real-world problem instance, the real execution time is used, as in our main manuscript. When optimising smaller instances, the CPU time is used, as CPLEX solver uses multiple thread dynamically, it’s hard to calculate the total real time while the total CPU time is available.

parameter setting in [2], LS is set as 5 and two values of PS is tested, 200 and 30. Table III illustrates the comparison of MAC, our ELS and MAELS using initial population of high and low diversity (referred to as MAELS_Low and MAELS_High, respectively). ELS and MAELS significantly outperform MAC with $PS \in \{30, 200\}$ and $LS = 5$.

TABLE III

COMPARISON AMONG ELS, MAELS AND MAC ON THE REAL-WORLD INSTANCE, THUS $|V_P| = 3$, $|V_D| = 3$ AND $|V_C| = 3,000$. THE BUDGET IS 240 MINUTES AS USED IN OUR PAPER. "BEST" AND "AVG" STAND FOR THE BEST AND AVERAGE DISTANCES (km) OBTAINED FROM 30 INDEPENDENT TRIALS, RESPECTIVELY. "STD" STANDS FOR THE STANDARD DEVIATION. THE LOWEST AVERAGE DISTANCE IS MARKED WITH *. RESULTS IN BOLD (OR UNDERLINE) INDICATE THAT THE CORRESPONDING ALGORITHM IS BETTER (OR WORSE) THAN ELS BASED ON WILCOXON RANK-SUM TEST WITH THE LEVEL OF SIGNIFICANCE 0.05.

Algorithm	BEST	AVG	STD
ELS	5568.5	5628.8	22.9
MAELS_High	5505.4	5628.5*	29.7
MAELS_LOW	5660.8	5719.9	25.3
MAC ($PC = 30$)	6432.9	<u>6565.2</u>	62.1
MAC ($PC = 200$)	6468.3	<u>6556.7</u>	35.5

2) *A fairer comparison:* Besides MAC using its own initialisation method, we also provide MAC the identical initial population as ELS and MAELS_High for a fair comparison. In this case, the population size is 30 as in our main manuscript. First, we compare the quality of routes decoded by MAC and our approaches given the identical initial population, illustrated on the second row (with 0 runtime) of Table IV. The routes decoded by MAC are statistically worse than the ones decoded by our approaches. Then, the quality of optimised routes are reported and compared periodically in Table IV. The optimised routes by MAC are significantly worse than the ones by our approaches.

C. Discussion

ELS and MAELS outperforms MAC on our real-world M3CVRP instance no matter if MAC uses the identical initial population of solutions or its own initialisation method. The possible reasons are as follows:

- Given the same amount of time, MAC iterates much fewer times than ELS and MAELS do since the time cost of its customer adjustment operator used for mutating permutations of tasks is large. The time complexity of its customer adjustment operator is $O(n^2)$ [2]. For more details of this operator, please refer to [2].
- The solutions decoded by MAC use fewer vehicles than the ones decoded by MAELS and ELS, which leads to fewer but longer trips. It may be difficult for MAC to find better and feasible solutions from those long trips during the local intensification phase.

IV. SUMMARY

Our solution approaches for M3CVRP use the framework of memetic algorithms, but with a novel local search procedure and new initialisation methods, which are shown to outperform the closest state-of-the-art algorithm for CGVRP [2] and the commercial solver CPLEX on our real-world problem instance.

REFERENCES

- [1] W. Lan, Z. Ye, P. Ruan, J. Liu, P. Yang, and X. Yao, "Novel memetic algorithms with smart initialisation for real-world large-scale waste collection problems," *Under review by IEEE Transactions on Evolutionary Computation*, 2021.
- [2] L. Wang and J. Lu, "A memetic algorithm with competition for the capacitated green vehicle routing problem," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 516–526, 2019.
- [3] G. Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, vol. 12, no. 4, pp. 568–581, 1964.

TABLE IV

THE COMPARISON AMONG ELS, MAELS AND MAC. MAC IS INITIALISED WITH THE IDENTICAL POPULATION OF MAELS_High. “BEST” AND “AVG” STAND FOR THE BEST AND AVERAGE DISTANCES (km) OBTAINED FROM 30 INDEPENDENT TRIALS. “STD” STANDS FOR THE STANDARD DEVIATION. THE LOWEST AVERAGE DISTANCE IS MARKED WITH *. RESULTS IN BOLD (OR UNDERLINE) INDICATE THAT THE CORRESPONDING ALGORITHM IS BETTER (OR WORSE) THAN ELS BASED ON WILCOXON RANK-SUM TEST WITH THE LEVEL OF SIGNIFICANCE 0.05. “+/-/ $\approx$ ” INDICATE THE NUMBER OF TIMES THAT THE RESULTS ARE STATISTICALLY BETTER/WORSE/SIMILAR TO THOSE OBTAINED BY ELS, RESPECTIVELY.

Time (mins)	ELS			MAELS_High			MAELS_Low			MAC		
	BEST	AVG	STD	BEST	AVG	STD	BEST	AVG	STD	BEST	AVG	STD
0	6148.0	6148.0*	0.0	6148.0	6148.0*	0.0	6148.0	6148.0*	0.0	6726.3	<u>6726.3</u>	0.0
30	5601.7	5649.3*	20.6	5574.7	<u>5696.8</u>	52.5	5704.2	<u>5776.0</u>	28.4	6587.7	<u>6669.9</u>	36.9
75	5601.5	5641.7*	22.0	5505.4	<u>5664.9</u>	53.6	5660.8	<u>5749.4</u>	26.4	6499.3	<u>6613.3</u>	48.6
120	5599.3	5639.1*	22.5	5505.4	5653.5	47.4	5660.8	<u>5736.9</u>	28.6	6474.3	<u>6599.3</u>	55.3
165	5569.2	5634.7*	23.4	5505.4	5636.5	35.1	5660.8	<u>5732.3</u>	27.5	6441.5	<u>6588.3</u>	60.7
195	5568.5	5633.2*	24.0	5505.4	5633.1	32.8	5660.8	<u>5724.7</u>	25.7	6441.5	<u>6578.5</u>	63.1
240	5568.5	5628.8	22.9	5505.4	5628.5*	29.7	5660.8	<u>5719.9</u>	25.3	6432.9	<u>6565.2</u>	62.1
+/- / $\approx$	-/-/-			0/2/5			0/6/1			0/7/0		