A two-individual evolutionary algorithm for cumulative capacitated vehicle routing with single and multiple depots

Yuji Zou, Jin-Kao Hao* , Qinghua Wu,

IEEE Transactions on Evolutionary Computation 29(2): 505-518, 2025

***Abstract*—The cumulative capacitated vehicle routing problem with single (CCVRP) or multiple depots (MDCCVRP) is a variant of the popular capacitated vehicle routing problem. Instead of minimizing the total travel time, the objective here is to minimize the sum of all customers' waiting times. This problem has a variety of real-world applications, especially in post-disaster humanitarian relief. To solve the challenging CCVRP and MDCCVRP, we propose a unified two-individual evolutionary algorithm that follows the memetic search framework. The algorithm integrates several key features: a two-individual population mechanism to accelerate convergence while maintaining population diversity, a dedicated edge assembly crossover to generate high-quality offspring and an adaptive feasible and infeasible local search to achieve a balanced exploration between feasible and infeasible solutions. The algorithm is evaluated on 39 CCVRP instances and 78 MDCCVRP instances commonly used in the literature. Computational results show that for the CCVRP, the algorithm outperforms the leading algorithms by achieving improved best results (new upper bound) for 13 instances and matching the best results for 23 other instances. For the MDCCVRP, the algorithm achieves 9 improved best results and matches the best results for the remaining instances. The critical components of the algorithm are investigated to understand their contributions.**

***Index Terms*—Vehicle routing; Cumulative capacitated routing; Heuristics; Evolutionary algorithm; Combinatorial optimization.**

I. Introduction

In the field of transport optimization, the classical capacitated vehicle routing problem (CVRP) is one of the most important problems. In the CVRP, each capacitated vehicle starts from the depot to serve the customers with demands and returns to the depot. Each customer is visited exactly once by one vehicle, and the sum of customer demand served by a vehicle does not exceed the vehicle's capacity. The objective is to minimize the total traveling time of all vehicles. The CVRP is known to be a general model to formulate a number of practical problems. To better accommodate more real-world scenarios, a family of routing problems known as rich vehicle routing problems [1] has been introduced, extending the conventional CVRP.

This work is partially supported by the National Natural Science Foundation Program of China (Grant No. 72122006) and the China Scholar Council (Grant No. 202106050037). (*Corresponding author*: Jin-Kao Hao)

Y. Zou and J.-K Hao are with the Department of Computer Science, LERIA, Université d'Angers, 2 bd Lavoisier, 49045 Angers Cedex 01, France (e-mails: yujizou6@gmail.com, jin-kao.hao@univ-angers.fr).

Q. Wu is with the School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China (e-mail: qinghuawu1005@gmail.com)

The cumulative capacitated vehicle routing problem (CCVRP) is one of those problem variants. As a customer-centric model [2], this problem aims to minimize the sum of customers' waiting time. The CCVRP naturally occurs in several applications. For example, when a disaster happens, it is important for rescue teams to reach the disaster site as soon as possible to limit the loss of life and suffering [3]. The school bus routing problem is another typical application, where the goal is to minimize the total travel time of all passengers [4].

The general CCVRP can be defined on a complete undirected graph $G = (V, E)$ with $V = D \cup C$ and $E = \{(i, j) : i, j \in V\}$, where D is the set of depots ($|D| \geq 1$) and $C = \{C_1, C_2, ...C_m\}$ is the set of customers. Moreover, a symmetric non-negative matrix $Y = (d_{ij})$ for the edges (i, j) is associated with E, where d_{ij} represents the travel time (or equivalently the distance) between two vertices. There is a fleet R of K vehicles with a given capacity. Each customer $i \in C$ has a demand q_i that is served when a vehicle visits the customer. A candidate solution to the CCVRP with single depot is a set of k ($k \leq K$) disjoint Hamiltonian tours starting and ending at the depot such that each customer is served exactly once by a vehicle, and the sum of customer demands served by a vehicle does not exceed its capacity.

For a given solution, let t_i^k be the arrival time of vehicle k at customer i (i.e., t_i^k is the waiting time of the customer i; $t_i^k = 0$ if i is not served by k). Then the goal of the CCVRP is to find a solution S that minimizes the sum of the waiting times of all customers.

$$Minimize \quad f(S) = \sum_{k \in R} \sum_{i \in C} t_i^k, \quad S \in \Omega \tag{1}$$

where Ω is the search space including all feasible candidate solutions for a CCVRP instance. A detailed mathematical formulation of the CCVRP can be found in [2].

The MDCCVRP is a generalization of the CCVRP with multiple depots ($|D| > 1$) [5]. In this problem, each depot in D is supposed to supply enough goods to serve all customers. Compared to the CCVRP, it is necessary to additionally decide the depot at which each route starts and ends.

The CCVRP and the MDCCVRP studied in this work are computationally challenging NP-hard problems [2, 6]. Due to their practical relevance and computational challenge, several methods have been proposed to tackle these problems in the past. However, as the review in Section II shows, some of the existing methods are designed for one problem only. When

they are designed for both problems, their performance lacks stability and may vary considerably on different benchmark instances. In this work, we propose a two-individual evolutionary algorithm (TIEA) that is able to effectively solve almost all CCVRP and MDCCVRP instances in the literature. Our contributions can be summarized as follows.

- The proposed TIEA algorithm incorporates several complementary search components. First, TIEA maintains a small population of only two individuals, which overcomes the disadvantage of a large population whose management induces computational costs. This reduced population still preserves the key features of a typical memetic algorithm including crossover and local optimization. Second, TIEA employs a special edge assembly crossover to generate high-quality offspring from the two individuals of the population. Third, TIEA uses a feasible and infeasible variable neighborhood descent to adaptively explore both feasible and infeasible search spaces, which helps to find high-quality solutions lying on the feasible-infeasible boundary.
- We conduct thorough experiments on benchmarks to evaluate the performance of TIEA. The results show that TIEA is highly competitive against the state-of-the-art algorithms. In particular, TIEA is able to achieve 13 record-breaking results (new upper bounds) for the 39 CCVRP instances and 9 new best results for the 78 MDCCVRP instances. In addition, TIEA can match all but three of the best-known results. TIEA achieves these results with a competitive computation time compared to the state-of-the-art methods.

The rest of the paper is organized as follows. Section II reviews the related works. Section III presents the proposed algorithm. Section IV shows experimental results and comparisons with the state-of-the-art methods. Section V provides investigations of the main components of the algorithm to shed light on their roles, followed by conclusions in Section VI.

II. Related works

TABLE I
Representative exact and heuristic algorithms for CCVRP and MDCCVRP

Literature	Method	Problem solved
Ngueveu *et al*. (2010) [2]	MA	CCVRP
Ribeiro and Laporte (2012) [7]	Adaptive LNS	CCVRP
Ke and Feng (2013) [8]	Two phase ILS	CCVRP
Vidal *et al.* (2014) [9]	HGS	CCVRP
Lysgaard and Wøhlk (2014) [10]	BCP	CCVRP
Sze *et al.* (2017) [11]	Adaptive VNS	CCVRP
Ke (2018) [12]	BSO	CCVRP
Nucamendi-Guillén *et al.* (2018) [13]	MIP and ILS	CCVRP
Smiti *et al.* (2020) [14]	SVNS	CCVRP
Kyriakakis *et al.* (2021) [15]	ACS-VND and MMAS-VND	CCVRP
Lalla-Ruiz and Voß (2020) [5]	POPMUSIC	MDCCVRP
Wang *et al.* (2019) [6]	PLS	MDCCVRP
Niu *et al.* (2021) [16]	MMAS	MDCCVRP
Nucamendi-Guillén *et al.* (2022) [17]	MIP	MDCCVRP&LLRP
Osorio-Mora *et al.* (2023) [18]	ILS	MDCCVRP&LLRP&MDk-TRP
Damião *et al.* (2021) [19]	BCP	CCVRP&MDCCVRP
Liu and Jiang (2019) [20]	GA-LNS	CCVRP&CCVRPTW
Kyriakakis *et al.* (2022) [21]	TS-VND	CCVRP&CCVRPTW
Kyriakakis *et al.* (2022) [22]	GRASP-VND	CCVRP&CUAVRP

A. Literature review of the CCVRP

The CCVRP was initially studied by Ngueveu *et al*. [2], who proposed a memetic algorithm (MA) that encodes a solution of multiple routes as a giant tour of all customers, which is suitable for standard crossover operators. To split the chromosome into separate routes during the local search phase, they employed the technique suggested in [23]. Three move operators were used to improve the solution during the local search phase. In their study, a set of instances comprising 50 to 199 nodes (referred to as CMT) was used to evaluate the performance of their algorithm. These instances were originally proposed by [24] for the VRP. In those instances, the fleet size of the vehicles was set to the minimum number necessary to handle the capacity constraints, while the distance limitation was ignored.

Ribeiro and Laporte [7] later introduced an adaptive large neighborhood search (ALNS) that utilizes seven removal and three insertion heuristics to solve the problem. They adapted a set of instances (referred to as GWKC) comprising 200 to 483 nodes from [25], using the same method as in [2]. Their experimental results showed that ALNS outperformed MA of [2] by producing better solutions in less computational time.

Vidal *et al.* [9] introduced a unified hybrid genetic search (UHGS) approach to solve a class of routing problems, collectively known as multi-attribute vehicle routing problems, including the CCVRP. UHGS integrates several critical components, including a local search with efficient route evaluation, a crossover based on the giant tour representation combined with the fast split procedure of [23], and an advanced population management method. The results obtained on CCVRP benchmark instances demonstrate excellent performance by achieving numerous best-known solutions.

Sze *et al*. [11] introduced an adaptive variable neighborhood search (AVNS) to solve the cumulative vehicle routing problem with min-sum and min-max objectives. The AVNS algorithm was divided into two stages. In the first stage, the best improvement method was applied, and each move operator was evaluated based on its contribution. In the second stage, a subset of move operators was chosen based on their performance in the first stage, and a so-called k-improvement acceptance criteria was used. Large neigbhorhood search (LNS) was used along with several destroy and repair methods to diversify the search. To evaluate the algorithm, the authors introduced a new set of large-scale instances (referred to as L), comprising 560 to 1040 nodes sourced from [26]. This powerful algorithm was able to find several new best solutions on the three instance sets.

Ke [12] proposed a brain storm optimization (BSO) approach for the CCVRP. This algorithm uses convergent and divergent operations to allow BSO, which was initially developed for continuous problems, to handle discrete problems. The algorithm also employs a special divide and conquer method to decompose the problem into smaller, easier-to-solve subproblems. The proposed method achieved some new best solutions on the three instance sets.

Smiti *et al*. [14] proposed a skewed general variable neighborhood search (SVNS) approach to address the CCVRP. SVNS relies on the VNS and incorporates an acceptance criterion based on the distance between the local optimal solution and the global best solution to accept worse solutions. Furthermore, SVNS employs three move operators to enhance

the solution quality, and a semi-random perturbation procedure with three operators to find a new starting point when the algorithm is trapped in a local optimum.

The aforementioned methods represent the current state-of-the-art approaches for solving the CCVRP. Other methods include the two-phase metaheuristic algorithm by [8] and the branch-and-cut-and-price (BCP) algorithm by [10]. Some recent studies also tested their methods on small-sized instances denoted as A, B, E, M, and P, which were originally proposed for the VRP [24]. These studies include ant colony system-variable neighborhood descent (ACS-VND) and max-min ant system-variable neighborhood descent (MMAS-VND) [15], improved mixed-integer formulations and iterated local search (ILS) [13]. Other related studies include a hybrid genetic algorithm for the cumulative capacitated vehicle routing problem with time-window (CCVRPTW) and the CCVRP [20], a hybrid tabu search-variable neighborhood descent (TS-VND) algorithm for these two problems [21], and a GRASP-VND algorithm for the cumulative unmanned aerial vehicle routing problem (CUAVRP) and the CCVRP [22].

B. Literature review of the MDCCVRP

Lalla-Ruiz and Voß [5] proposed the first work on the MDCCVRP, in which they presented a mathematical formulation and solved the problem using a metaheuristic approach called partial optimization metaheuristic under special intensification condition (POPMUSIC). This method reduces the original problem to a sub-problem that can be solved by an approximate or exact approach. In addition, the authors introduced two instance sets to evaluate their algorithm. The first set named "lr," was based on instances in [27]. The second set, called "p" and "pr", was originally proposed by [27] for the MDVRP. The fleet size for these instances was set to 35.

Wang *et al*. [6] presented a perturb-based local search (PLS) algorithm for the MDCCVRP. The PLS algorithm starts from a feasible initial solution and iteratively improves the solution using six operators under the VND framework. When a local optimum is reached, two types of perturbations are used to continue the search. The authors tested their algorithm on the instance sets from [5], and showed that for most instances, PLS achieved better results in less time.

Niu *et al*. [16] introduced a max-min ant system (MMAS) based on decomposition for the MDCCVRP. The original problem was decomposed into a series of smaller problems that were solved using the max-min ant system. To avoid getting stuck in local optima, a perturbation mechanism consisting of increasing the probability of searching for solutions with low pheromone and employing two random move operators was used. The authors reported some record-breaking results while it should be noted that some results were found to be even better than the optimal results established in [17].

Damião *et al*. [19] used position indices to determine the contribution of an edge to the arrival of the remaining customers. They then solved the resulting model using the branch-cut-and-price (BCP) algorithm of the VRPSolver package. The proposed approach was evaluated on small-sized CCVRP instances and all MDCCVRP instances, and achieved several new best results for MDCCVRP instances.

Nucamendi-Guillén *et al*. [17] introduced two novel mixed-integer formulations for the latency location routing problem (LLRP), which were further adapted to the MDCCVRP. Their results showed significant improvements in obtaining optimal solutions compared with POPMUSIC.

Osorio-Mora *et al*. [18] proposed an iterated local search for the MDCCVRP. ILS consists of three procedures: perturbation, local search (LS), and simulated annealing cooperated with variable neighborhood descent (SA-VND). The LS procedure allows infeasible solutions to be visited using a fixed parameter to penalize capacity violations. In the SA-VND procedure, a VND procedure is applied after the SA search. The ILS algorithm successfully found feasible solutions for nine instances of the MDCCVRP that were previously unsolvable by the BCP algorithm [19]. However, it is worth mentioning that the ILS algorithm only managed to match a subset of the remaining results achieved by the BCP algorithm, which represents the best-known results. ILS was also tested on the related LLRP and the multiple depots k-traveling repairman problem.

Table I shows the representative algorithms for solving the CCVRP and MDCCVRP. See [28] for a comprehensive review on vehicle routing with cumulative objectives.

III. Two-individual evolutionary algorithm for CCVRP and MDCCVRP

The proposed two-individual evolutionary algorithm is based on the framework of the memetic algorithm (MA) [29], which combines population-based genetic search with neighborhood-based local search. The basic idea of an MA is to take advantage of these complementary search methods. In fact, it is generally believed that population-based search provides more opportunities for exploration, while neighborhood search provides more opportunities for exploitation. When combined in an appropriate way the resulting hybrid method can provide, a good balance between exploitation and exploration, thus ensuring high search performance.

The performance of an MA depends on the design of two key search components: crossover and local search. Their design should incorporate useful problem-specific knowledge of the given problem to ensure intensified search of specific search regions and extensive exploration of promising areas.

MAs have been successfully used to solve many NP-hard combinatorial optimization problems, including several routing problems. However, maintaining a population (even with some tens of solutions) in an MA can be time-consuming. Recently, some algorithms based on a small population of two or three individuals have proven to be highly successful [30, 31, 32]. Inspired by this idea, our TIEA algorithm uses a population of only two individuals to speed up the search process. In addition to this feature, TIEA incorporates a dedicated edge assembly crossover (dEAX) and a powerful local optimization based on adaptive feasible and infeasible variable neighborhood descent (AFIVND). Other algorithmic components that contribute to its performance include a population initialization procedure, an ejection chain based mutation procedure, and a diversity preserving population updating method.

As shown in Algorithm 1. The TIEA algorithm first creates a population of two-individuals (line 3), then these individuals

undergo improvement by the AFIVND procedure (line 5). At each generation (one *while* loop), a new offspring solution S is generated by the dEAX crossover with the two individuals (line 9). A mutation procedure is then applied with probability m_p to slightly change the offspring for diversification (line 10). The offspring is further improved by the AFIVND procedure (line 11), and the search information is updated according to the obtained feasible local optimum (lines 12-16). Meanwhile, this solution is recorded in an adaptive memory M (line 17). Finally, the population is updated with the improved offspring solution (line 18). If the best-found objective does not change during I_r successive generations, the worse individual of the population is replaced by a random solution from the adaptive memory M (lines 19-21). The algorithm stops and returns the best recorded solution when a given time limit (or an allowed number of generations) is reached.

Algorithm 1: Pseudo-code of TIEA

```
Input: Input graph G = (V, E), time limit t_max, frequency of
       updating the penalty parameter u_p, penalty parameter β,
       frequency of renewing the population I_r
Output: The best solution S_b found.
1  I ← 0; // Iteration counter
2  I_n ← 0; // Counter of consecutive loops f_b is not improved
3  P ← IniPool; // Initial two-individual pop, see Section III-A
4  for i ← 1 to 2 do
5      P_i ← AFIVND(P_i, u_p); // See Section III-D
6      if f(P_i) < f(S_b) then
7          S_b ← P_i; // Update the best solution ever found
8  while t_max is not reached do
9      S ← dEAX(P_1, P_2); // See Section III-B
10     S ← Mutation(S, m_p); // See Section III-C
11     S' ← AFIVND(S, u_p);// Output the local optimum
12     if f(S') < f(S_b) then
13         I_n ← 0;
14         S_b ← S'; // Record the best solution ever found
15     else
16         I_n ← I_n + 1;
17     Update adaptive memory M with S';
18     UpdatingPop(P, S'); // See section III-E
19     if I_n ≥ I_r then
20         Replace the worse solution in P with a solution in M; //
             See section III-E
21         I_n ← 0;
22     I ← I + 1;
23 return S_b; // Return the best feasible solution ever found.
```

A. Population initialization

One notable characteristic of the CCVRP is that the weight (travel time) of an edge in a tour impacts the waiting time of all customers that follow the edge. Furthermore, the edge that returns to the depot in a tour has no contribution to the objective value of the tour. As such, given a solution, we can reduce its objective value by moving the last customer from any existing tours to a new tour, since this reduces the waiting time for that customer. Thus, any solution can be improved by this observation if the number of tours is less than K (the fleet size) [2]. Consequently, our initialization method builds initial solutions with K vehicles according to a greedy or random construction method. With the random method, the unvisited customer nodes are randomly assigned to the K tours. With the greedy method, we try to assign the customer nodes by minimizing the objective function, which is explained below.

For the single depot CCVRP, the greedy method first assigns the K customer nodes closest to the depot as the first node of each tour. Then, it extends each tour sequentially by adding to the tour the closest unvisited customer from the last node of the tour. This is based on the fact that minimizing the cost of the edges preceding a node favors the objective minimization of the resulting solution. This process continues until all customers have been added to a tour.

For the MDCCVRP, the greedy method first selects a depot for a new tour from the available depots. To do this, it determines the depot and the first node of the new tour based on the minimum distance between all depots and the unselected customers. Specifically, it chooses a depot i and an unvisited customer j with the smallest distance d_{ij}. This process of depot selection and new tour creation is repeated until the desired number of tours (K) has been reached. The remaining procedure follows the same greedy method for the CCVRP described above.

To obtain the two solutions of the initial population, one solution is created randomly, while the other solution is built using the aforementioned greedy method. Once obtained, each of these solutions is submitted to the AFIVND procedure for local optimization (Section III-D3). It is important to note that AFIVND accepts infeasible solutions as input because the capacity constraint is ignored during the greedy and random construction methods. After the solution has been improved by AFIVND, the first solution is always inserted into the population, while the second solution is inserted only if the distance between the two solutions, defined as the number of non-common edges (as described in Section III-E), exceeds ten percent of the total number of edges in a solution. Otherwise, a new initial solution is generated using the random construction method. This initial population procedure ensures a sufficient level of population diversity.

B. Offspring generation based on dEAX

In memetic algorithms, the crossover plays a crucial role as a key mechanism for exploring the search space. A well-designed crossover is expected to allow offspring to inherit valuable characteristics from the parent solutions while also introducing diversity into the offspring [33]. The popular edge assembly crossover operator (EAX) was originally designed for the traveling salesman problem (TSP) [34, 35] and later adapted to capacitated vehicle routing [36], vehicle routing with time windows [37], and split delivery vehicle routing [38]. EAX is based on the insight that high-quality solutions share common edges, which are likely to be part of the optimal solution. Additionally, in the offspring generated by EAX, the majority of edges are inherited from the parents, with only a few extra edges introduced to eliminate sub-tours, resulting in high-quality offspring with fewer long-distance edges. This characteristic enables efficient search, saving expensive computational effort for local search to improve the offspring solution. In this work, we adapt the idea of EAX to the CCVRP and MDCCVRP, and we use dEAX to denote the resulting crossover to distinguish it from the original EAX crossover.

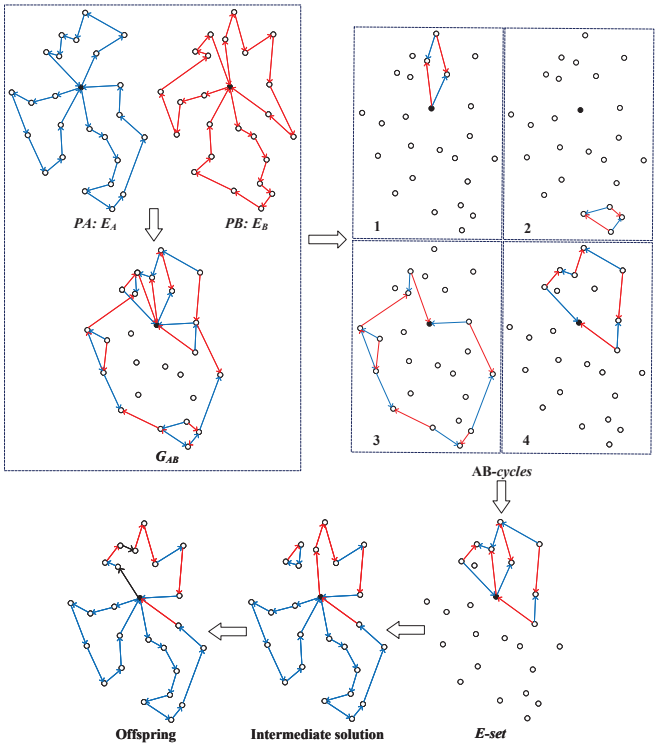


Fig. 1. Illustration of the dEAX crossover for the CCVRP. The black point is the depot.

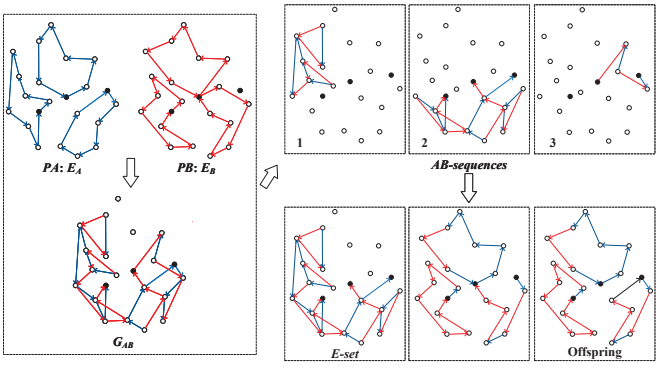


Fig. 2. Illustration of the dEAX crossover for the MDCCVRP. The black points are the depots.

When the visit sequence of vertices in a CCVRP solution is reversed, it can lead to a solution with a significantly different objective value. Particularly, in high-quality solutions, altering the relative visit order of vertices can substantially degrade the solution quality. Therefore, compared to the CVRP, when applying dEAX, the edge orientation should be taken into accounted. Let P_A and P_B be the parent solutions for the crossover. We define two graphs, $G_A = \{V, E_A\}$ and $G_B = \{V, E_B\}$, where E_A is the edge set of solution P_A, and E_B is the edge set of solution P_B. The dEAX procedure consists of the following steps.

- Generation of a joint graph $G_{AB} = \{V, (E_A \cup E_B) \setminus (E_A \cap E_B)\}$.
- The edges in G_{AB} are partitioned into several AB-*cycles*. An AB-*cycle* is a cycle that begins with a randomly selected node that has connected edges. Next, an adjacent edge is chosen randomly with respect to this node, and edges from G_A and G_B are alternately linked with different directions until a cycle is formed. Once an AB-*cycle* is formed, its edges are removed from G_{AB}. This process is repeated until there are no edges left in G_{AB}.
- The *E-set* is generated using the block strategy, which involves randomly selecting an AB-*cycle*, and then selecting the AB-*cycles* that share at least one node with the chosen cycle to form the *E-set*.
- The intermediate solution is generated by removing the edges from G_A that are included in the *E-set* (i.e., *E-set*$\cap E_A$), and adding the edges from G_B that are included in the *E-set* (i.e., *E-set*$\cap E_B$). In this step, G_A serves as the base solution.
- If there are sub-tours, the sub-tours are connected to the existing routes with the fewest number of nodes using 2-opt*. This involves removing two edges: one from a sub-tour and one from the existing routes. Then, two new edges with the same directions as the deleted edges are inserted, while keeping the directions of the other edges unchanged. During this step, all possible combinations are explored, and the two edges that result in the minimum difference between the added edges and the deleted edges are selected. This process is repeated until all sub-tours are eliminated.

Fig. 1 provides an illustrative example. In this example, AB-*cycle* 1 is selected as the central AB-*cycle*, which shares a node with AB-*cycle* 4. The *E-set* is formed by these two AB-*cycles*. In the intermediate solution, there exists a sub-route that contains three arcs. This sub-tour is eliminated by replacing two arcs with two new arcs indicated by black lines, leading to the offspring solution.

It should be mentioned that in the original EAX crossover [34, 35], multiple *E-sets* can be generated, resulting in numerous intermediate solutions and offsprings. However, in our dEAX crossover, only one *E-set* is randomly produced, leading to the generation of a single offspring. Using one offspring offers several advantages. Firstly, it helps avoid the production of a large number of similar offspring, thus preserving the diversity of the algorithm. Secondly, feeding only one offspring as the input for the expensive local search can speed up the convergence of the algorithm, especially in large-scale instances where a number of possible *E-sets* can be generated, and each local search iteration demands considerable computation time. Finally, the dEAX crossover ignores the capacity constraint and thus can generate infeasible offspring. It is less expensive to repair one infeasible offspring solution than to repair multiple infeasible solutions.

For the MDCCVRP, dEAX performs the same steps as for the CCVRP, except the generation of "AB-*cycles*". Due to the presence of multiple depots, the number of edges with the same or opposite direction connected with the same depot in G_{AB} may be odd, making it impossible to form an AB-*cycle*. To address this issue, all depots are treated as a single node in our approach. Since it may not be a *cycle* any more, we use the term AB-*sequence*. Specifically, when selecting an edge

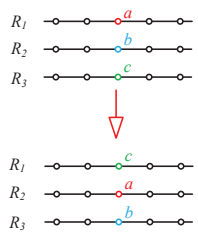


Fig. 3. Illustration of the ejection chain for the mutation.

connected to a depot during the generation of AB-*sequences*, the subsequent edge can be any unselected edge connected to any depot, forming an AB-*sequence*. However, this may lead to an invalid solution such that some routes may not form a closed tour, as depicted in the middle sub-figure between the "E-set" and "offspring" sub-figures in Fig. 2. To resolve this problem, we begin from the edges departing from the depot, and when we reach a node that is linked to another depot, we delete this involved edge and connect this node with the starting depot. As depicted in Fig. 2, we consider all depots as one node when generating AB-*sequence* 2 and AB-*sequence* 3. AB-*sequence* 1 is selected as the central AB-*sequence*, then AB-*sequence* 1 and AB-*sequence* 2 constitute the *E-set*. Additionally, we ensure that the route forms a closed tour before generating a valid intermediate solution. Since there are no sub-tours, the intermediate solution obtained serves as the offspring through this dEAX crossover. dEAX allows us to obtain a legal offspring without deviating from the fundamental concept of EAX.

C. Mutation for the individual

An offspring from the crossover typically inherit many edges from the parents. As a result, the offspring may be too similar to the parents. To introduce diversity into the offspring, we apply with probability m_p a mutation procedure to modify the solution. Specifically, three customers in different routes are selected, then their positions are changed with an ejection chain manner. As shown in Fig. 3, the nodes a, b and c from routes R_1, R_2 and R_3 are repositioned ($a \rightarrow b \rightarrow c \rightarrow a$). This procedure is applied m_l times (a parameter named mutation length) and introduces a few edges that were not present in the parents, creating a diversified offspring.

D. Adaptive feasible and infeasible variable neighborhood descent

Adaptive feasible and infeasible variable neighborhood descent (AFIVND) is the key local optimization component of the proposed TIEA algorithm. AFIVND adopts variable neighborhood descent (VND) [39] and reinforces VND with an adaptive feasible and infeasible search strategy, with seven neighborhoods $N_i, i = 1, \ldots, 7$ (see Section III-D4).

Algorithm 2 summarizes the general scheme of AFIVND. The input solution is first repaired if it is infeasible (lines 6-7). Then the improvement procedure starts with the first

Algorithm 2: Pseudo-code of AFIVND

Input: Input graph $G = (V, E)$, input solution S, frequency of updating the penalty parameter u_p
Output: The best feasible solution S' found during the search.

1 $\gamma \leftarrow 1$;
2 $\beta \leftarrow 10^{10}$; // Set β to be a large enough number
3 $I_f \leftarrow 0$; // Counter of consecutive iterations of feasible solutions
4 $I_i \leftarrow 0$; // Counter of consecutive iterations of infeasible solutions
5 $\epsilon \leftarrow 10^{-4}$; // ϵ is a positive decimal close to 0
6 **if** *Is_infeasible(S)* **then**
7 Repair the solution S; // See Section III-D1
8 $\beta \leftarrow \frac{f(S)}{\sum_{i \in C} q_i}$; // The initial value of β
9 $S' \leftarrow S$;
10 **while** *stopping condition is not met* **do**
11 $S'' \leftarrow S \oplus N_\gamma$; // Perform the first improvement with N_γ
12 **if** $f^p(S'') < f^p(S)$ **then**
13 **if** *Is_infeasible(S'')* **then**
14 $I_i \leftarrow I_i + 1$;
15 $I_f \leftarrow 0$;
16 **if** $I_i > u_p$ **then**
17 **if** $\beta < 0$ **then**
18 $\beta \leftarrow \epsilon$;
19 $\beta \leftarrow \beta \cdot (1.5 + rand(0, 1))$;
20 **else**
21 $I_f \leftarrow I_f + 1$;
22 $I_i \leftarrow 0$;
23 **if** $I_f > u_p$ **then**
24 **if** $\beta > 0$ **then**
25 $\beta \leftarrow \frac{\beta}{1.5 + rand(0,1)}$;
26 **if** $\beta < \epsilon$ **then**
27 $\beta \leftarrow -\epsilon$;
28 **else**
29 $\beta \leftarrow \beta \cdot (1.5 + rand(0, 1))$;
30 **if** $f(S'') < f(S')$ **then**
31 $S' \leftarrow S''$; // Record the best feasible solution
32 $S \leftarrow S''$; // Accept the better solution under f^p
33 $\gamma \leftarrow 1$; // Move to the first neighborhood
34 $\gamma \leftarrow \gamma + 1$; // Move to the next neighborhood
35 **return** S'; // Return the best feasible solution found during AFIVND

neighborhood N_1, and exploits it with the first-improvement strategy (line 11). Each time an improved solution under the penalty-based fitness function (see Eq. (2)) is found, the penalty parameter β is adaptively adjusted based on the search information (lines 12-29) (see Section III-D3). At the same time, the best feasible solution is recorded (lines 30-31). The solution is updated as the current solution (line 32) and the algorithm switches to the first neighborhood N_1 (line 33). Otherwise, when the algorithm reaches a local optimum within the current neighborhood, the algorithm switches to the next neighborhood (line 34). AFIVND terminates when no better solution can be found after exploring all the neighborhoods.

1) Repair procedure: Due to the relaxation of vehicle capacity during the initialization procedure and the dEAX crossover, an infeasible solution may be generated. In AFIVND, the penalty parameter (see below) is dynamically adjusted based on the feasibility information from the search history. When AFIVND improves an infeasible solution that is far from the feasible region, it may take several local search runs to make the solution feasible by significantly

increasing the penalty parameter. In such cases, the penalty parameter loses its ability to balance the feasible and infeasible exploration. Therefore, we repair the infeasible input solution before applying AFIVND to make the solution as close to feasible as possible. Our repair procedure uses the inter-route 2-opt* operator (see Section III-D4) to reassign customers. The evaluation of a solution is based on the modified objective function (Eq. (2)), with the penalty parameter β set to a sufficiently large positive value to strongly penalize capacity violations. The 2-opt* operator is applied iteratively until the solution becomes feasible or all neighborhood solutions induced by the move operator have been explored, at which point the repair procedure stops.

2) Neighborhood reduction: In our algorithm, we consider the neighborhood of each customer with a number of nearest vertices, called α-nearest ($\alpha < |V|$). The rationale behind this approach is that the neighborhood that includes customers with a long distance is less likely to yield a good solution. By focusing on the nearest vertices, the algorithm can avoid examining unpromising solutions, thus increasing the search efficiency. This approach has proven effective in solving other routing problems, as shown in previous studies [40, 41].

3) Adaptive feasible and infeasible search: During its search, AFIVND relaxes the capacity constraint and uses the penalty-based fitness function shown in Eq. (2) to guide the search process to visit both feasible and infeasible search spaces. The infeasibility degree allowed is controlled by dynamically adjusting the β parameter based on search information. This approach is consistent with the concept of strategic oscillation [42], which allows for exploration of infeasible search spaces to introduce more flexibility into the algorithm. Indeed, allowing transitions between infeasible and feasible regions can significantly improve the performance of neighborhood-based local search methods.

In AFIVND, how to adjust the penalty parameter β is especially important, because high-quality solutions may lie on the boundary between the feasible and infeasible search spaces. By controlling β, the algorithm is encouraged to explore both types of search spaces and not to remain trapped exclusively in the feasible or infeasible regions, facilitating the discovery of potentially high-quality solutions.

$$f^p(x) = f(x) + \beta \sum_{k=1}^{K} max(0, \sum_{i \in r_k} q_i - Q) \qquad (2)$$

Specifically, the penalty parameter β is initially set as the average cost per unit of demand for each route in the feasible solution. Then, the parameter is updated adaptively based on the search information. Two counters, I_f and I_i, are used to keep track of the number of consecutively accepted feasible and infeasible solutions in the previous iterations, respectively. Whenever a solution is accepted, these counters are updated accordingly. If I_f or I_i reaches a predetermined threshold u_p, the procedure for changing the penalty parameter is triggered. If the previous iterations have all resulted in feasible solutions, the penalty parameter is decreased to encourage the algorithm to move to infeasible regions. Conversely, if the algorithm is deemed to have spent too much time in the infeasible search space, the penalty parameter is increased to encourage exploration of the feasible search space. The specific calculation method can be observed in lines 19 and 25 of Algorithm 2. It is worth noting that we introduce some randomness (rand value) into the equation to add diversity and prevent possible cycling after neighborhood reduction. When the parameter is close to zero, we make it negative.

4) Neighborhood operators: In AFIVND, we use seven neighborhood operators N_1 to N_7 to make slight changes to the current solution. They include both intra-route and inter-route operators, with the exception of N_7, which is specifically an inter-route operator. It is worth noting that among the previous studies, only Sze *et al.* [11] applied N_7 for the CCVRP. However, N_7 leads to a considerably large neighborhood with a size of $O(\frac{n^2\alpha^2}{|R|^2})$, which is quite expensive to explore. To reduce the computational time needed for examining N_7, our algorithm uses a filter method to reduce this neighborhood. This is the first time the N_7 neighborhood has been exploited for the MDCCVRP.

N_1 (*Relocate*). A customer is removed from its original position and inserted into another position in the same route or a different route.

N_2 (*Swap*). This operator swaps the positions of two customers from the same or different routes. For the MDCCVRP, the swap of two depots is also considered.

N_3 (*2-relocate*). Two consecutive customers are removed from their original positions and inserted into another position in the same or different routes.

N_4 (*2-opt*). This operator is different depending on inter-route or intra-route. For the intra-route, two non-adjacent edges are deleted, and two new edges are added. Meanwhile, the edges between the two deleted edges are reversed. For the inter-route, also called *2-opt**, two edges are deleted, and two new edges are added.

N_5 (*Node-arc exchange*). This operator changes the position of a node and an arc. The node and the arc can be from the same or different routes.

N_6 (*Arc-arc exchange*). Two consecutive customers exchange their positions with two consecutive customers in the same or different routes.

N_7 (*Swap**). Two customers are first selected from different routes. These customers are then removed from their original positions and inserted into the best positions in each other's route. It is crucial to emphasize that in our algorithm, this operator is executed exclusively when the related routes of the selected customers have overlapping segments.

It's worth noting that the operators N_3, N_5, and N_6 can be considered as special cases of the general *cross-exchange* operator [43], where the substring size is limited to 2.

E. Population updating

The population update strategy is a crucial procedure in maintaining the diversity of the two-individual population. In TIEA, the distance between two solutions S_a and S_b, denoted as $Dis(S_a, S_b)$, is defined as the number of non-common edges in the two solutions, i.e., $Dis(S_a, S_b) =$

Algorithm 3: Pseudo-code of *UpdatingPop*

Input: Population P, solution to be inserted S'
Output: The updated population P.
if $\mathcal{D}(S', P) > 0.1 \times \nu$ **then**
  if $Fit(S') > Fit(S_w)$ **then**
    Delete S_w from P; // S_w is the worst individual in P
    Insert S' into P;
return P

$|E_a| - |E_a \cap E_b|$. The distance between a solution S and the population P, denoted as $\mathcal{D}(S, P)$, is defined as the minimum distance between S and any solution in the population, as shown in Eq. (3).

$$\mathcal{D}(S, P) = min\{Dis(S, S_i) : S_i \in P \setminus S\} \tag{3}$$

Let S' represent the candidate solution for insertion into the population P. If its distance from the population, as defined in Eq. 3, exceeds ten percent of the total number of edges in a solution (denoted as ν in Algorithm 3), we consider inserting the individual S' into the population. Subsequently, we recalculate the fitness of all individuals in P. Let $P' = P \cup \{S'\}$. We define $f_{max} = max_{S_i \in P'}\{f(S_i)\}$, $f_{min} = min_{S_i \in P'}\{f(S_i)\}$, $\mathcal{D}_{max} = max_{S_i \in P'}\{\mathcal{D}(S_i, P')\}$, and $\mathcal{D}_{min} = min_{S_i \in P'}\{\mathcal{D}(S_i, P')\}$. The fitness of the individuals in P' is calculated using Equation (4).

$$Fit(S) = \delta \frac{f_{max} - f(S)}{f_{max} - f_{min}} + (1 - \delta)\left(\frac{\mathcal{D}(S, P') - \mathcal{D}_{min}}{\mathcal{D}_{max} - \mathcal{D}_{min}}\right) \tag{4}$$

This fitness function considers both the quality and diversity of the solutions, with the parameter δ set to be 0.55. If the inserted individual has a better fitness than the worst individual in P, the solution S' replaces the worst individual (see Algorithm 3).

To enhance population diversity in case of search stagnation, our proposed algorithm incorporates an adaptive memory M [44], which stores the high-quality solutions encountered during the AFIVND procedure. Each solution improved by AFIVND is added to the adaptive memory, which has a predefined size of 5000. Once the memory reaches its limit, the oldest solution is removed to make space for the newest one, and the solutions are organized based on their insertion order. If the best solution remains unchanged for I_r consecutive iterations, we use a random solution from the first half of the memory (i.e., the solutions inserted earlier) to replace the worst individual in the population.

F. Discussion

In this section, we provide a discussion about the novelties and particular features of the proposed TIEA algorithm.

First, most existing algorithms restrict their search to the feasible space. However, visiting intermediate infeasible solutions can be beneficial for transitioning between different feasible population, which may be difficult to achieve within the feasible space alone. Our proposed algorithm allows for the exploration of infeasible solutions using a dynamic penalty approach to control capacity constraint violations. While a few studies, such as [7, 11, 18, 21], have considered infeasible solutions, they typically use a fixed penalty parameter. In contrast, our TIEA algorithm dynamically adjusts the penalty parameter to strike a balance between exploring feasible and infeasible search spaces. This approach prevents the algorithm from focusing excessively on either feasible or infeasible spaces for too long, and allows the algorithm to explore the boundary between the two types of search spaces, where high-quality solutions may lie. The experiments presented in Section V-C also show the merit of this approach.

Second, previous works, such as those by [2, 9, 20], have employed an evolutionary framework. In their algorithms, a solution is represented as a permutation of all customers (a giant tour), and they apply the standard order crossover (OX) operator to generate new offspring permutations. To convert these visit sequences into a classic VRP solution, a split procedure is used to decompose the permutation into a set of closed tours. Unlike these approaches, our dEAX crossover directly manipulates the multiroutes of the parent solutions to transfer favorable features (i.e., shorter subtours) from the parent solutions to the offspring, eliminating the need to decode the giant tour into multiroutes. As shown in Section V-B, our dEAX crossover outperforms the giant tour approach using the order crossover and the split procedure both in terms of solution quality and computational efficiency.

Third, another distinctive feature of our work is that, unlike other population-based algorithms [2, 12, 16, 22] that typically use several tens of individuals, our TIEA algorithm is the first hybrid approach based on a very small population of only two individuals for the CCVRP and the MDCCVRP. As shown in Section V-A, the use of such a small population size proves to be highly effective when compared to approaches that maintain larger populations of individuals.

Finally, compared to the local search based algorithms [6, 8, 13, 14], only our algorithm incorporates the swap* move operator. This move operator allows the algorithm to explore solution spaces that may be difficult for other approaches to visit, while keeping the additional computational resources required relatively modest.

IV. Computational results

In this section, we present an extensive computational study of the proposed TIEA algorithm applied to benchmark instances for both the CCVRP and the MDCCVRP. The main purpose is to provide a comprehensive evaluation of the algorithm's performance and to assess its effectiveness in solving these challenging problems. The instances used in our experiments and the solutions obtained by our algorithm are available at https://github.com/YujiZou/CCVRP. The code of our algorithm will be made available on this website when the paper is published.

A. Benchmark instances

For the CCVRP, we adopt three widely used benchmark instance sets known as CMT, GWKC, and L. To provide

a comprehensive assessment, we also show in the online supplemental appendix our results on five small-sized instance sets, named A, B, E, M, and P. For the MDCCVRP, we use three benchmark instance sets: lr, p, and pr.

1) Benchmark instances for CCVRP:

- Set CMT: This set includes 7 instances with 50 to 199 customers. It was originally introduced in [2] and has been extensively tested in subsequent studies.
- Set GWKC: This set consists of 20 medium-sized instances with 200 to 483 customers. It was first introduced in [7] and has been widely tested in many studies.
- Set L: This set contains 12 large-sized instances with 560 to 1200 customers. It was first presented in [11] and has only been evaluated in the original paper and a subsequent research conducted by [12].

2) Benchmark instances for MDCCVRP:

- Set S_1: This set consists of 18 instances with 10 to 100 customers and was initially introduced by [5]. It should be mentioned that three instances, lr10, lr11, and lr12, were originally tested with 20 vehicles in [5], while in [6], the number of vehicles used was 25. For our study, we conduct tests on these three instances using both 20 and 25 vehicles. Consequently, the set actually comprises 21 instances.
- Set S_2: This set, introduced in [5], contains 24 instances with 48 to 360 customers. These instances have been modified from the original sets p and pr, which were introduced by [27] for the multi-depot vehicle routing problem (MDVRP), by setting the number of vehicles in each instance to 35.
- Set S_3: This set, originally proposed for the multi-depot vehicle routing problem in [27], comprises instance sets p and pr with the original number of vehicles.

B. Experimental protocol and reference algorithms

TABLE II
PARAMETER TUNING RESULTS

Parameter	Related section	Description	Considered value	Final value
m_p	III-C	mutation probability	{0,0.1,0.2,0.3}	0.3
m_l	III-C	mutation length	{1,2,3,4,5}	2
α	III-D2	granularity threshold	{10,15,20,25,30}	20
u_p	III-D3	frequency of updating penalty parameter	{2,4,6,8,10}	3
I_r	III-E	frequency of replacing population	{50,100,150,200,250}	150

The TIEA algorithm requires five main parameters: the mutation probability (p_m), the mutation length (p_l), the granularity threshold (α), the frequency of updating the penalty parameter (u_p), and the frequency of replacing the population (I_r). To determine reasonable values for the parameters, we used the automatic parameter tuning package Irace [45]. Through this tuning process, we obtained the parameter configuration presented in Table II. This configuration can be regarded as the default parameter setting of TIEA, and it was consistently used for all the experiments conducted in this study.

To evaluate our TIEA algorithm, we compare it against the following state-of-the-art algorithms.

- For the CCVRP, we adopt five reference methods: the adaptive large neighborhood search (ALNS) [7], the unified hybrid genetic search (UHGS) [9], the two-stage adaptive variable neighborhood search (AVNS) [11], the brain storm optimization (BSO) algorithm [12], and the skewed variable neighborhood search (SVNS) [14]. For the CMT dataset, UHGS, AVNS and BSO were able to achieve all the best-known results. For the GWKC dataset, UHGS, AVNS and BSO retain the majority of the best-known results, while ALNS retains one best-known result. For the L dataset, only AVNS and BSO were tested. We note that the source code for SVNS is unavailable, so we faithfully re-implemented it (the code will be available at https://github.com/YujiZou/CCVRP). Furthermore, in [11], two versions of results are presented: one with five runs and the other with ten runs. We use the version that performed the best with ten runs as our reference. In addition to these five reference algorithms, we also include a comparison with the best-known solution (BKS) from all existing algorithms.
- For the MDCCVRP, we adopt three best reference algorithms: the perturb-based local search (PLS) [6], the iterated local search [18] and the branch-cut-and-price (BCP) algorithm [19]. It should be noted that although [16] presented some record-breaking results, some of them were found to be even better than the proven optimal results in [17]. Therefore, we ignore [16] in our study.

The proposed TIEA algorithm was implemented in C++ and compiled using the g++ compiler with the -O3 optimization option. The main experimental environments for the compared algorithms are provided in Table III. For the three commonly studied CCVRP instance sets (CMT, GWKC, and L), our stopping condition was determined by the running time of the leading algorithm for each instance given in the literature, which are shown in the detailed results presented in Tables A.I – A.III of the online supplemental appendix. The TIEA algorithm was executed ten times per instance. For the small-sized CCVRP instance sets, the stopping condition was set to 5000 generations, and the algorithm was run 20 times per instance. For the MDCCVRP, the stopping condition was set to 1000 generations (crossovers). In addition, to further show the performance of our algorithm in the long run, we extended the running generations to 5000 generations in our experiments. The algorithm for the MDCCVRP was run 30 independent times like [6].

TABLE III
EXPERIMENTAL ENVIRONMENTS OF THE COMPARED ALGORITHM

Algorithm	Programming language	Processor	CPU(GHz)	RAM(GB)	Operation system
ALNS [7]	C++	Pentium Core 2	2.0	3	Windows
UHGS [9]	C++	Opteron 275	2.2	-	-
AVNS [11]	C++	Core i7	3.4	8	-
BSO [12]	C++	Pentium 4	2.4	4	-
SVNS [14]	C++	Xeon E5-2670	2.5	2	Linux
PLS [6]	C++	Core i5-4210M	2.6	4	-
BCP [19]	Julia/CPLEX 12.9	Core i7-3770	3.4	16	Ubuntu
ILS [18]	C++	Core i7-8700K	3.7	32	Ubuntu
TIEA (this work)	C++	Xeon E5-2670	2.5	2	Linux

C. Computational results and comparison

The TIEA algorithm is evaluated through a comparative analysis with the reference algorithms discussed in Section IV-A on the CCVRP instances in Section IV-A1 and on the MDCCVRP instances in Section IV-A2.

TABLE IV
SUMMARY OF THE COMPARATIVE RESULTS OF THE BEST AND AVERAGE OBJECTIVE VALUES BETWEEN TIEA AND REFERENCE ALGORITHMS ON THE THREE SETS OF 39 COMMONLY USED CCVRP INSTANCES.

Instance	Pair algorithms	f_{best}				f_{avg}			
		#Wins	#Ties	#Losses	*p*-value	#Wins	#Ties	#Losses	*p*-value
CMT(7)	TIEA vs. BKS	0	7	0	-	-	-	-	-
	TIEA vs. AVNS	0	7	0	-	5	0	2	0.30
	TIEA vs. BSO	0	7	0	-	5	0	2	0.12
	TIEA vs. UHGS	0	7	0	-	1	1	5	0.015
	TIEA vs. ALNS	2	5	0	0.69	6	0	1	0.11
	TIEA vs. SVNS	2	5	0	0.69	7	0	0	0.16
GWKC(20)	TIEA_EX vs. BKS	5	13	2	0.97	-	-	-	-
	TIEA vs. BKS	4	9	7	0.05	-	-	-	-
	TIEA vs. AVNS	12	8	0	1.18e-3	18	0	2	1.34e-4
	TIEA vs. BSO	13	5	2	1.29e-2	16	0	4	5.32e-2
	TIEA vs. UHGS	7	8	5	0.50	3	0	17	3.42e-4
	TIEA vs. ALNS	20	0	0	1.91e-6	19	0	1	5.72e-6
	TIEA vs. SVNS	20	0	0	1.91e-6	20	0	0	1.91e-6
L(12)	TIEA_EX vs. BKS	8	3	1	1.28e-2	-	-	-	-
	TIEA vs. BKS	8	3	1	1.28e-2	-	-	-	-
	TIEA vs. AVNS	9	2	1	7.65e-3	12	0	0	4.89e-4
	TIEA vs. BSO	10	2	0	4.44e-3	11	0	1	1.46e-3

1) Comparative results on the CCVRP: Table IV shows a summary of the comparative results between the TIEA algorithm and the six references for the CCVRP instances, while the detailed results are provided in Tables A.I–A.III of the online supplemental appendix. In Table IV, the first column represents the instance set. Column f_{best} provides a summary of the best objective value achieved among 20 independent runs, while f_{avg} summarizes the average value. Additionally, the column labeled "#Wins" indicates the number of instances where our TIEA outperformed the reference algorithm, "#Ties" represents the number of instances with the same results, and "#Losses" indicates the number of instances where our TIEA achieved worse results compared to the reference algorithm. We also include the *p*-values from the Wilcoxon signed-rank test (with a significance level of 0.05) applied to the best and average values to verify the statistical significance of the observe performance differences between TIEA and each reference algorithm.

Based on the results, it is evident that TIEA outperforms the reference algorithms for most instances. Compared to one of the best reference algorithms UHGS [9], our TIEA algorithm also achieves better solutions, although the dominance trend is less pronounced, partly due to the longer running time of UHGS (see the appendix). Generally, TIEA improves on the best-known solutions for 12 instances. For the CMT instances, TIEA achieves all of the best-known results. For the media-sized GWKC instances, TIEA achieves 4 new best results and matches 9 best-known results. For the large-sized L instances, TIEA reports 8 new record-breaking results and matches 3 best-known results. Its result is slightly worse only in one case, with a negligible gap of 0.00019% to the best-known value.

To investigate the performance of our proposed algorithm with extended running time, we performed another experiment by extending the running time of TIEA to two hours for the medium-sized GWKC instances and three hours for the large-sized L instances. The summarized results are shown in Table IV (TIEA_EX) while the detailed results are provided in Table A.IV of the online supplemental appendix. Compared to the best-known results for these 32 GWKC and L instances, TIEA_EX improved the results for 13 instances and matched the results for other 16 instances. These results demonstrate the ability of the algorithm to find high-quality solutions.

We also evaluated the TIEA algorithm on the instance sets A, B, E, M, and P, which have been studied in previous CCVRP works. These instances are relatively small in size and are considered easier. As shown in Tables A.V–A.VIII of the online supplemental appendix, our algorithm is able to achieve the best-known solutions for all these instances, except for one case, surpassing all reference algorithms. These results provide additional evidence of the effectiveness and competitiveness of our algorithm in solving the CCVRP.

TABLE V
SUMMARY OF THE COMPARATIVE RESULTS ON THE BEST AND AVERAGE OBJECTIVE VALUES BETWEEN TIEA AND REFERENCE ALGORITHMS ON THE THREE MDCCVRP SETS OF 78 INSTANCES, WHERE TIEA-1000 AND TIEA-5000 REPORT THE RESULTS WITH A STOPPING CONDITION OF 1000 GENERATIONS AND 5000 GENERATIONS RESPECTIVELY.

Instance	Pair algorithms	f_{best}			f_{avg}		
		#Wins	#Ties	#Losses	#Wins	#Ties	#Losses
$S_1(21)$	TIEA-1000 vs. BKS	0	21	0	-	-	-
	TIEA-1000 vs. PLS	8	10	0	18	0	0
	TIEA-1000 vs. BCP	0	21	0	-	-	-
	TIEA-1000 vs. ILS	3	18	0	17	4	0
$S_2(24)$	TIEA-1000 vs. BKS	0	20	4	-	-	-
	TIEA-1000 vs. PLS	21	3	0	22	2	0
	TIEA-1000 vs. BCP	0	20	4	-	-	-
	TIEA-1000 vs. ILS	13	11	0	22	2	-
$S_3(33)$	TIEA-1000 vs. BKS	9	15	9	-	-	-
	TIEA-1000 vs. PLS	29	4	0	33	0	0
	TIEA-1000 vs. BCP	9	15	9	-	-	-
	TIEA-1000 vs. ILS	25	8	0	31	0	2
$S_1(21)$	TIEA-5000 vs. BKS	0	21	0	-	-	-
	TIEA-5000 vs. PLS	8	10	0	18	0	0
	TIEA-5000 vs. BCP	0	21	0	-	-	-
	TIEA-5000 vs. ILS	3	18	0	17	4	0
$S_2(24)$	TIEA-5000 vs. BKS	0	24	0	-	-	-
	TIEA-5000 vs. PLS	21	3	0	22	2	0
	TIEA-5000 vs. BCP	0	24	0	-	-	-
	TIEA-5000 vs. ILS	14	10	0	22	2	0
$S_3(33)$	TIEA-5000 vs. BKS	9	24	0	-	-	-
	TIEA-5000 vs. PLS	29	4	0	33	0	0
	TIEA-5000 vs. BCP	9	24	0	-	-	-
	TIEA-5000 vs. ILS	26	7	0	33	0	0

2) Results on the MDCCVRP: The comparative results for the 78 MDCCVRP instances are presented in Table V, while the detailed results are provided in Tables A.IX–A.XI of the online supplemental appendix. TIEA-1000 and TIEA-5000 refer to our algorithm with 1000 and 5000 generations respectively. The results indicate that our TIEA algorithm outperforms the reference algorithms by achieving 9 new best results and matching the remaining 69 best-known results.

Specifically, compared to the PLS and ILS heuristic algorithms, TIEA-1000 demonstrates a significant performance advantage. It achieves better results in 58 out of 75 instances and matches the remaining instances. Additionally, TIEA-1000 consistently outperforms PLS in terms of average results. A similar trend is observed when comparing with ILS, as TIEA outperforms ILS in 41 instances and matches the remaining results. By examining Tables A.IX–A.XI of the online supplemental appendix, it is evident that TIEA-1000 requires less computing time than PLS and requires significantly less time

than ILS to find equal or better solutions.

Compared to the exact BCP algorithm, TIEA consistently and quickly achieves all optimal solutions proven by BCP. For all instances in S_1, the best and average results of TIEA are identical, showing the consistency of our algorithm in achieving the optimal results over multiple runs. This trend is also observed for most of the instances in S_2. For the most difficult instances of S_3, this consistency is less obvious, but the gap between the best and average results remains very small.

Overall, the comparative results confirm the competitiveness of TIEA in solving the MDCCVRP problem. TIEA outperforms the PLS and ILS algorithms in terms of both improved results and computational efficiency. TIEA is able to find all known optimal solutions proven by BCP and to discover improved best solutions for 9 of the most difficult instances.

V. Assessment of algorithmic components

In this section, we conduct additional experiments to gain deeper insights into the individual influences of the main components of the TIEA algorithm. We focus on three key components: the small population, the dEAX crossover, and the adaptive feasible and infeasible search. To assess the impact of these components, we systematically replaced each component with other methods to create algorithm variants. For example, to demonstrate the superiority of the dEAX crossover, we created two algorithm variants, one replacing the dEAX crossover with the OX method using the giant tour encoding, and the other using the multi-start mechanism to replace the dEAX crossover. By comparing these modified algorithm variants with the original TIEA, we can isolate the effects of each component and analyze its influences on the overall algorithm's performance. The experiments were based on the 32 CCVRP difficult instances of sets GWKC and L. We set the experimental parameters of the TIEA variants to be consistent with TIEA, as presented in Table II. The stopping condition is also identical to that of TIEA, as detailed in Tables A.II and A.III in the supplementary materials. All compared variants were run on the same computer as TIEA.

A. Benefits of the small population

As outlined in Section III, our TIEA approach uses a small population consisting of only two individuals. To investigate the advantages of such a small population, we introduced four variants of TIEA: HEA_1 with only one individual, HEA_5 with five individuals, HEA_{20} with twenty individuals and HEA_{40} with forty individuals. In the variant HEA_1, where only one individual is present, the crossover is disabled, and a multi-start approach, combined with a random construction method (see Section III-A), is applied to continue the algorithm until the time budget is reached. For HEA_5, HEA_{20}, and HEA_{40}, as the number of individuals increases, the frequency I_r of renewing the population is also increased to ensure that the population is explored fully. Specifically, 200 replacement iterations are used for HEA_5, 1000 for HEA_{20}, and 2000 for HEA_{40}. In addition, since there are four different patterns (diamond, rectangle, flower and circle) in the GWKC and L instance sets related to the distribution of customer nodes, we have selected one representative instance from each of the four patterns to draw the convergence charts in Fig. 4.

From Table VI, it can be concluded that the use of a small population enhances the performance of the algorithm. TIEA consistently achieves better results on a large majority of the tested instances, demonstrating the effectiveness of the small population. Although TIEA may exhibit slightly worse performance in a few instances among the medium-sized cases, it generally produces better solutions for most instances. In the large-sized instances where algorithm effectiveness is critical, the advantage of the small population becomes more apparent, with only two instances experiencing a performance decline. The small p-values further confirms this conclusion. However, it should be noted that the average results obtained by TIEA are not as good as those of the variants with more individuals. This discrepancy may be due to the fact that a larger population ensures more population diversity, which can contribute to better stability of the algorithm.

The convergence charts in Fig. 4 vividly illustrate the remarkable effectiveness of the TIEA algorithm induced by the small population, as it quickly converges to better solutions. We can observe that TIEA's initial solution is much worse than the variants with more individuals due to the small population size. However, as time goes, TIEA converges much faster and better than the other variants, demonstrating the benefit of the small population. It's worth mentioning that we observed a similar trend on other instances during our experiments.

In summary, the computational analysis and convergence charts confirm the valuable contribution of a small population in enhancing the performance of TIEA.

TABLE VI
Summary of the comparative results on the best and average objective values between TIEA and four variant algorithms: HEA_1 with only 1 individual, HEA_5 with 5 individuals, HEA_{20} with 20 individuals, and HEA_{40} with 40 individuals

Instance	Pair algorithms	f_{best}				f_{avg}			
		#Wins	#Ties	#Losses	p-value	#Wins	#Ties	#Losses	p-value
GWKC(20)	TIEA vs. HEA_1	19	1	0	1.91e-6	20	0	0	1.91e-6
	TIEA vs. HEA_5	6	12	2	0.18	3	0	17	1.34e-4
	TIEA vs. HEA_{20}	9	9	2	0.10	3	0	17	1.21e-2
	TIEA vs. HEA_{40}	11	8	1	1.11e-3	5	0	15	4.84e-2
L(12)	TIEA vs. HEA_1	12	0	0	4.88e-4	12	0	0	4.88e-4
	TIEA vs. HEA_5	7	3	2	0.04	5	0	7	0.42
	TIEA vs. HEA_{20}	9	2	1	2.44e-3	5	0	7	0.42
	TIEA vs. HEA_{40}	11	1	0	4.88e-4	6	0	6	0.30

B. Benefits of the dEAX crossover

The TIEA algorithm incorporates a dedicated dEAX crossover that allows the heritage of common features from parent solutions to offspring. We now explore the benefits of the dEAX crossover by introducing two variant algorithms: TIEA_OX and TIEA_NX. In TIEA_OX, we replaced the dEAX crossover with the order crossover (OX) that uses the giant tour encoding combined with the fast split decoding procedure introduced in [23]. This method has been used in [2] for solving the CCVRP. In our implementation, we adopted an improved version of the OX crossover, called adaptive order crossover (AOX), which prevents the introduction of excessive

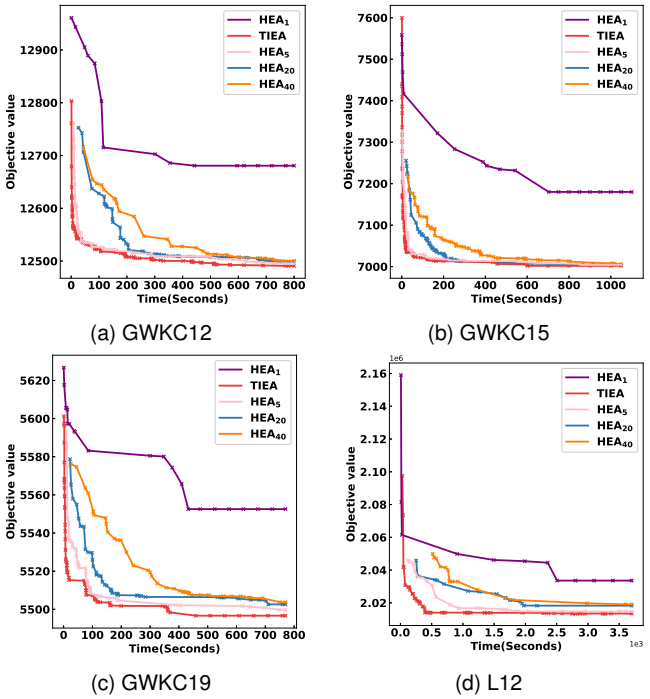


Fig. 4. Convergence charts (running profiles) of TIEA and four TIEA variants (denoted by HEA_1, HEA_5, HEA_{40}, HEA_{20}) on four representative CCVRP instances (GWKC12, GWKC15, GWKC19 and L12) where HEA_i ($i = 1, 5, 20, 40$) uses a population of i individuals.

TABLE VII
SUMMARY OF THE COMPARISON RESULTS BETWEEN TIEA AND THE VARIANTS TIEA_OX AND TIEA_NX, WHERE TIEA_OX USES THE ADAPTIVE ORDER CROSSOVER AND TIEA_NX DOES NOT USE ANY CROSSOVER.

Instance	Pair algorithms	f_{best}				f_{avg}			
		#Wins	#Ties	#Losses	*p*-value	#Wins	#Ties	#Losses	*p*-value
GWKC(20)	TIEA vs. TIEA_OX	12	7	1	0.01	17	0	3	1.21e-2
	TIEA vs. TIEA_NX	19	1	0	1.91e-6	20	0	0	1.91e-6
L(12)	TIEA vs. TIEA_OX	9	3	0	6.84e-3	9	0	3	6.40e-2
	TIEA vs. TIEA_NX	12	0	0	4.88e-4	12	0	0	4.88e-4

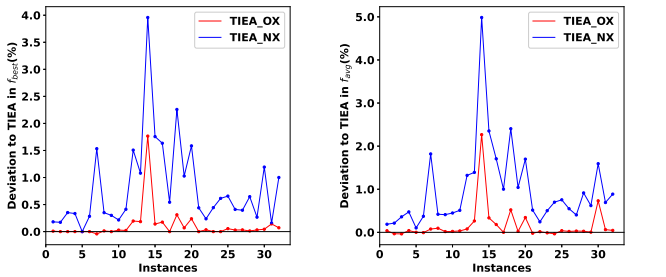


Fig. 5. Comparative results under 10 independent runs of TIEA with two variants on the CCVRP instance sets GWKS and L (32 instances), where TIEA_OX employs the order crossover with split procedure and TIEA_NX does not use a crossover.

long edges in the offspring and proves to be successful in solving the CVRP. Hence, TIEA_OX uses the AOX method together with the split decoding procedure to generate the offspring. In TIEA_NX, on the other hand, we simply removed the dEAX crossover. To ensure that the time budget is effectively used by TIEA_NX, we restart the algorithm repetitively until the time limit is reached. The TIEA_NX variant allows us to better understand the impact of the dEAX crossover itself, with the solutions being improved solely through the adaptive feasible and infeasible variable neighborhood descent. We ran the compared algorithms under the same experimental conditions as described in Section IV-B.

Table VII presents the comparative results between TIEA and the TIEA_OX and TIEA_NX variants, together with the corresponding *p*-values. Fig. 5 visualizes the deviation of the two variants from the reference results obtained by TIEA, in terms of the best and average objective values.

Table VII and Fig. 5 clearly show that the use of a crossover can enhance the performance of the algorithm, as observed for both TIEA and TIEA_OX. In terms of the best and average objective values, TIEA outperforms its competitors in all but one instance when compared to TIEA_NX. Furthermore, compared to TIEA_OX, TIEA consistently achieves better or equal solutions for all but one instance. In addition, TIEA performs better than TIEA_OX in terms of the average objective values in most instances. The small *p*-values further confirm the statistically significant differences between the compared data.

This experiment highlights the significant role of the dEAX crossover in the TIEA algorithm.

C. Benefits of the adaptive feasible and infeasible search

The local optimization procedure AFIVND adaptively explores both feasible and infeasible solutions. In this section, we examine the usefulness of this strategy by comparing it with two algorithmic variants: TIEA_FP and TIEA_NP. TIEA_FP explores infeasible solutions by applying a fixed penalty parameter, initially set as the average cost per demand unit for each route of the VND input solution after the repair procedure. On the other hand, TIEA_NP visits only feasible solutions. For this, we set the penalty value to an extremely large value to avoid any violation of the capacity constraint.

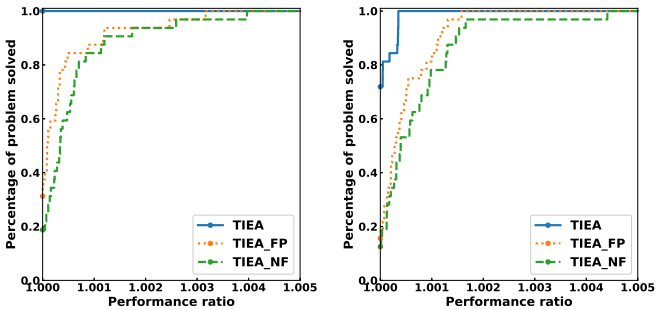
The comparative results of the three compared algorithms in solving the 39 CCVRP instances are shown in Table VIII. To further evaluate the compared algorithms, we used a benchmark tool called the Performance Profile [46], shown in Fig. 6. Given a set of algorithms O and a set of instances T, the performance ratio is defined as $\zeta_{o,t} = \frac{f_{o,t}}{min\{f_{o,t}\}}$, where $o \in O, t \in T$. This variable represents the performance of algorithm o on the instance t compared to the best performance among all the algorithm on instance t. The performance function of algorithm o is defined as $T_O(\upsilon) = \frac{|t \in T | \zeta_{o,t} < \upsilon|}{|T|}$, which calculates the fraction of instances that algorithm o can reach with at most υ many times the best algorithm.

From the information presented in Table VIII, it is evident that our TIEA algorithm consistently obtains better solutions than the two algorithm variants in all instances tested. In terms of the average objective values, TIEA shows a clear advantage over the other algorithms. Furthermore, allowing the algorithm to explore infeasible solutions and to move back and forth between different search regions proves beneficial

in finding high-quality solutions. Fig. 6 highlights that TIEA achieves the highest $T_O(1)$ values for both the best and average objective values, indicating its ability to produce the best results. Moreover, TIEA is the first to reach $T_O(v)$, implying its consistent ability to solve all instances.

TABLE VIII
SUMMARY OF THE COMPARATIVE RESULTS BETWEEN TIEA AND THE TWO VARIANTS TIEA_FP AND TIEA_NP, WHERE TIEA_FP ALLOWS TO VISIT INFEASIBLE SOLUTIONS WITH A FIXED PARAMETER TO PENALIZE THE CAPACITY VIOLATION AND TIEA_NP DOES NOT ALLOW TO VISIT INFEASIBLE SOLUTIONS.

Instance	Pair algorithms	f_{best}				f_{avg}			
		#Wins	#Ties	#Losses	p-value	#Wins	#Ties	#Losses	p-value
GWKC(20)	TIEA vs. TIEA_FP	12	8	0	1.40e-3	16	0	4	4.00e-2
	TIEA vs. TIEA_NP	10	2	0	2.14e-4	18	0	2	1.02e-3
L(12)	TIEA vs. TIEA_FP	16	4	0	3.35e-3	11	0	1	6.84e-3
	TIEA vs. TIEA_NP	10	2	0	1.46e-3	8	0	4	4.25e-2



(a) Best objective value (b) Average objective value

Fig. 6. The performance profiles of TIEA and the two variants TIEA_FP and TIEA_NP on 32 CCVRP instances from sets GWKC and L. The left figure corresponds to the best results, while the right figure shows the average results.

VI. CONCLUSION

The cumulative vehicle routing problem is a relevant model for various real-world scenarios. Many efforts have been made to develop solution methods for solving this NP-hard problem. In this study, we proposed an effective two-individual evolutionary algorithm (TIEA) to address this difficult problem, considering both the case with a single depot (CCVRP) and the case with multiple depots (MDCCVRP). Our TIEA algorithm features a small population of only two individuals, which helps to speed up the search process while preserving the potential of a memetic algorithm. We introduced a special crossover operator for the CCVRP and the MDCCVRP, inspired by the popular edge assembly crossover designed for the TSP, to generate high-quality offspring. We also designed an adaptive feasible and infeasible variable neighborhood descent to improve offspring solutions by dynamically exploring both feasible and infeasible search spaces.

To evaluate the performance of our algorithm, we conducted extensive experiments using three sets of 39 CCVRP instances and three sets of 78 MDCCVRP instances. The results demonstrate that our algorithm outperforms all the reference algorithms. Specifically, TIEA achieves 13 new best results and matches 23 best-known results for the CCVRP instances, while achieving 9 best results and matching the remaining 69 best-known results with significantly reduced computation time for the MDCCVRP. In addition, we conducted experiments on 85 small-sized instances, which confirmed the effectiveness and competitiveness of our algorithm by reaching 84 best-known results. Finally, we presented additional experiments to understand the contributions of key algorithmic components, including the two-individual strategy, the dEAX crossover, and the adaptive feasible and infeasible variable neighborhood descent.

Two directions can be considered for future work. First, the order in which the neighborhood structures are explored may influence the search progress, and a fixed order may not be appropriate for all situations. Therefore, it would be interesting to use learning techniques, such as reinforcement learning, to determine the order of application of the neighborhoods. Second, the modified dEAX crossover that proves to be effective for the MDCCVRP, could be tested to solve other routing problems involving multiple depots.

ACKNOWLEDGMENTS

We thank Prof. Salhi Said for answering our questions related to their work reported in [11].

REFERENCES

[1] R. F. Hartl, G. Hasle, and G. K. Janssens, "Special issue on rich vehicle routing problems," *Cent. Europ. J. Oper. Res.*, vol. 14, no. 2, p. 103, 2006.
[2] S. U. Ngueveu, C. Prins, and R. W. Calvo, "An effective memetic algorithm for the cumulative capacitated vehicle routing problem," *Comput. Oper. Res.*, vol. 37, no. 11, pp. 1877–1885, 2010.
[3] J. Molina, A. López-Sánchez, A. G. Hernández-Díaz, and I. Martínez-Salazar, "A multi-start algorithm with intelligent neighborhood selection for solving multi-objective humanitarian vehicle routing problems," *J. Heuristics*, vol. 24, pp. 111–133, 2018.
[4] R. Bowerman, B. Hall, and P. Calamai, "A multi-objective optimization approach to urban school bus routing: Formulation and solution method," *Transp. Res. Pt. A-Policy Pract.*, vol. 29, no. 2, pp. 107–123, 1995.
[5] E. Lalla-Ruiz and S. Voß, "A popmusic approach for the multi-depot cumulative capacitated vehicle routing problem," *Optim. Lett.*, vol. 14, no. 3, pp. 671–691, 2020.
[6] X. Wang, T.-M. Choi, Z. Li, and S. Shao, "An effective local search algorithm for the multidepot cumulative capacitated vehicle routing problem," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 12, pp. 4948–4958, 2019.
[7] G. M. Ribeiro and G. Laporte, "An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem," *Comput. Oper. Res.*, vol. 39, no. 3, pp. 728–735, 2012.
[8] L. Ke and Z. Feng, "A two-phase metaheuristic for the cumulative capacitated vehicle routing problem," *Comput. Oper. Res.*, vol. 40, no. 2, pp. 633–638, 2013.
[9] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "A unified solution framework for multi-attribute vehicle routing problems," *Eur. J. Oper. Res.*, vol. 234, no. 3, pp. 658–673, 2014.
[10] J. Lysgaard and S. Wøhlk, "A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem," *Eur. J. Oper. Res.*, vol. 236, no. 3, pp. 800–810, 2014.
[11] J. F. Sze, S. Salhi, and N. Wassan, "The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: An effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search," *Transp. Res. Pt. B-Methodol.*, vol. 101, pp. 162–184, 2017.
[12] L. Ke, "A brain storm optimization approach for the cumulative capacitated vehicle routing problem," *Memet. Comput.*, vol. 10, pp. 411–421, 2018.
[13] S. Nucamendi-Guillén, F. Angel-Bello, I. Martínez-Salazar, and A. E. Cordero-Franco, "The cumulative capacitated vehicle routing problem:

New formulations and iterated greedy algorithms," *Expert Syst. Appl.*, vol. 113, pp. 315–327, 2018.
[14] N. Smiti, M. M. Dhiaf, B. Jarboui, and S. Hanafi, "Skewed general variable neighborhood search for the cumulative capacitated vehicle routing problem," *Int. Trans. Oper. Res.*, vol. 27, no. 1, pp. 651–664, 2020.
[15] N. A. Kyriakakis, M. Marinaki, and Y. Marinakis, "A hybrid ant colony optimization-variable neighborhood descent approach for the cumulative capacitated vehicle routing problem," *Comput. Oper. Res.*, vol. 134, p. 105397, 2021.
[16] M. Niu, R. Liu, and H. Wang, "A max-min ant system based on decomposition for the multi-depot cumulative capacitated vehicle routing problem," in *IEEE Congr. Evol. Comput., CEC - Proc.* Virtual, Krakow, Poland: IEEE, 2021, pp. 620–627.
[17] S. Nucamendi-Guillén, I. Martínez-Salazar, S. Khodaparasti, and M. E. Bruni, "New formulations and solution approaches for the latency location routing problem," *Comput. Oper. Res.*, vol. 143, p. 105767, 2022.
[18] A. Osorio-Mora, J. W. Escobar, and P. Toth, "An iterated local search algorithm for latency vehicle routing problems with multiple depots," *Comput. Oper. Res.*, vol. 158, p. 106293, 2023.
[19] C. M. Damião, J. M. P. Silva, and E. Uchoa, "A branch-cut-and-price algorithm for the cumulative capacitated vehicle routing problem," *4OR-Q. J. Oper. Res.*, vol. 21, pp. 47–71, 2023.
[20] R. Liu and Z. Jiang, "A hybrid large-neighborhood search algorithm for the cumulative capacitated vehicle routing problem with time-window constraints," *Appl. Soft. Comput.*, vol. 80, pp. 18–30, 2019.
[21] N. A. Kyriakakis, I. Sevastopoulos, M. Marinaki, and Y. Marinakis, "A hybrid tabu search–variable neighborhood descent algorithm for the cumulative capacitated vehicle routing problem with time windows in humanitarian applications," *Comput. Ind. Eng.*, vol. 164, p. 107868, 2022.
[22] N. A. Kyriakakis, M. Marinaki, N. Matsatsinis, and Y. Marinakis, "A cumulative unmanned aerial vehicle routing problem approach for humanitarian coverage path planning," *Eur. J. Oper. Res.*, vol. 300, no. 3, pp. 992–1004, 2022.
[23] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Comput. Oper. Res.*, vol. 31, no. 12, pp. 1985–2002, 2004.
[24] N. Christofides, A. Mingozzi, and P. Toth, "The vehicle routing problem," in *Combinatorial Optimization*, N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, Eds. John Wiley & Sons, 1979, pp. 315–338.
[25] B. L. Golden, E. A. Wasil, J. P. Kelly, and I.-M. Chao, "The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results," in *Fleet Management and Logistics*. Boston MA: Springer US, 1998, pp. 33–56.
[26] F. Li, B. Golden, and E. Wasil, "Very large-scale vehicle routing: new test problems, algorithms, and results," *Comput. Oper. Res.*, vol. 32, no. 5, pp. 1165–1179, 2005.
[27] J.-F. Cordeau, M. Gendreau, and G. Laporte, "A tabu search heuristic for periodic and multi-depot vehicle routing problems," *Networks*, vol. 30, no. 2, pp. 105–119, 1997.
[28] K. Corona-Gutiérrez, S. Nucamendi-Guillén, and E. Lalla-Ruiz, "Vehicle routing with cumulative objectives: A state of the art and analysis," *Comput. Ind. Eng.*, vol. 169, p. 108054, 2022.
[29] C. C. Ferrante Neri and P. Moscato, *Handbook of memetic algorithms*. Heidelberg, Germany:Springer, 2011.
[30] L. Moalic and A. Gondran, "Variations on memetic algorithms for graph coloring problems," *J. Heuristics*, vol. 24, pp. 1–24, 2018.
[31] J. Ding, Z. Lü, C.-M. Li, L. Shen, L. Xu, and F. Glover, "A two-individual based evolutionary algorithm for the flexible job shop scheduling problem," in *AAAI Conf. Artif. Intell.*, vol. 33, no. 01, Honolulu, HI, USA, 2019, pp. 2262–2271.
[32] Y. Zhou, G. Wang, J.-K. Hao, N. Geng, and Z. Jiang, "A fast tri-individual memetic search approach for the distance-based critical node problem," *Eur. J. Oper. Res.*, vol. 308, no. 2, pp. 540–554, 2023.
[33] J.-K. Hao, "Memetic algorithms in discrete optimization," in *Handbook of Memetic Algorithms*, F. Neri, C. Carlos, and M. Pablo, Eds. Heidelberg, Germany: Springer, 2012, vol. 379, ch. 6, pp. 73–94.
[34] Y. Nagata, "Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem," in *Proceedings of the 7th Internatinal Conferencen on Genetic Algorithms*, East Lansing, MI, USA, 1997, pp. 450–457.
[35] Y. Nagata and S. Kobayashi, "A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem," *INFORMS J. Comput.*, vol. 25, no. 2, pp. 346–363, 2013.
[36] Y. Nagata and O. Bräysy, "Edge assembly-based memetic algorithm for the capacitated vehicle routing problem," *Networks*, vol. 54, no. 4, pp. 205–215, 2009.
[37] Y. Nagata, O. Bräysy, and W. Dullaert, "A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows," *Comput. Oper. Res.*, vol. 37, no. 4, pp. 724–737, 2010.
[38] P. He and J.-K. Hao, "General edge assembly crossover-driven memetic search for split delivery vehicle routing," *Transp. Sci.*, vol. 57, no. 2, pp. 482–511, 2023.
[39] N. Mladenović and P. Hansen, "Variable neighborhood search," *Comput. Oper. Res.*, vol. 24, no. 11, pp. 1097–1100, 1997.
[40] K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," *Eur. J. Oper. Res.*, vol. 126, no. 1, pp. 106–130, 2000.
[41] P. Toth and D. Vigo, "The granular tabu search and its application to the vehicle-routing problem," *INFORMS J. Comput.*, vol. 15, no. 4, pp. 333–346, 2003.
[42] F. Glover and J.-K. Hao, "The case for strategic oscillation," *Ann. Oper. Res.*, vol. 183, pp. 163–173, 2011.
[43] É. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin, "A tabu search heuristic for the vehicle routing problem with soft time windows," *Transp. Sci.*, vol. 31, no. 2, pp. 170–186, 1997.
[44] C. D. Tarantilis, "Solving the vehicle routing problem with adaptive memory programming methodology," *Comput. Oper. Res.*, vol. 32, no. 9, pp. 2309–2327, 2005.
[45] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. Stützle, "The irace package: Iterated racing for automatic algorithm configuration," *Oper. Res. Perspect.*, vol. 3, pp. 43–58, 2016.
[46] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Math. Program.*, vol. 91, pp. 201–213, 2002.

Supplementary materials of paper "A two-individual evolutionary algorithm for cumulative capacitated vehicle routing with single and multiple depots"

Yuji Zou, Jin-Kao Hao*, Qinghua Wu

I. Detailed results for the CCVRP

This appendix provides the computational results of our TIEA algorithm and the reference algorithms on the CCVRP instances.

A. Results on the commonly used CMT, GWKC, and L sets

Tables A.I–A.III present detailed comparative results on these most commonly used instance sets under the experimental protocol given in Section IV-B in the main text. In these tables, the first column displays the name of the instance, while the second and third columns indicate the number of customers and the size of the vehicle fleet, respectively. The remaining columns provide computational information for each algorithm, including our TIEA and the reference algorithms. Columns f_{best} and f_{avg} present the best objective value and the average objective value over ten runs, respectively. Column T_{avg} displays the average running time for each reference algorithm and we use the shortest time as the cut-off time of our TIEA algorithm. The reported running times have been adjusted to account for differences in CPU frequency with respect to the 2.0GHz processor used to run the ALNS algorithm [8] (see Table III in Section IV-B of the main text for CPU information). Specifically, the reported times are multiplied by 1.25 for SVNS (2.5GHz CPU) [9], 1.1 for UHGS (2.2GHz CPU) [11], 1.7 for AVNS (3.4GHz CPU) [10], 1.2 for BSO (2.4GHz CPU) [2], and 1.25 for TIEA (2.5GHz CPU) (This work). Since the run time of an algorithm also depends on other factors (programming language, data structures, compiler...), the timing information is provided for indicative purposes only. The best results among the algorithms are highlighted in bold. The results show that our TIEA algorithm competes very favorably with the reference algorithms.

Table A.IV presents a comparative assessment on the GWKC and L sets between the best-known solutions (BKS) and our TIEA algorithm with longer running times (TIEA_EX) (two hours for the medium GWKC instances and three hours for the large L instances) under 20 independent runs. The dominant results are highlighted in bold.

B. Results on the small CCVRP instances

Table A.V presents detailed results on the 85 small-size instances from 5 instance sets, which were tested in several recent studies. As most reference algorithms, we run TIEA 20 times per instance with a stopping condition of 5000 generations per run. As our references, we use the following state-of-the-art algorithms: BCP (branch-and-cut-and-price) [5], IG-PRB (iterated greedy algorithm) [6], ACS-VND (hybrid ant colony system-variable neighborhood descent) [3], and HTS-VND (hybrid tabu search-variable neighborhood descent) [4].

In Table A.V. The first column indicates the name of the instance set, while the remaining columns provide the number of instances for which an algorithm reports the best result and the average running time for all instances in the set. The last row of the table displays the total result information for all five instance sets combined. The computational time has been adjusted based on the main frequency of the CPU. Specifically, to maintain consistency with the previously mentioned running times, we have used a base CPU frequency of 2.0GHz as well. Therefore, the reported running times have been multiplied by the following factors: 1.265 for BCP (2.53GHz CPU) [5], 1.7 for ACS-VND (3.4GHz CPU) [3], 1.2 for IG-PRB (2.4GHz CPU) [6], 1.75 for HTS-VND (3.5GHz CPU) [4], 1.25 for TIEA (2.5GHz CPU) (this work). Tables A.VI–A.VIII provide detailed result for each individual instance. These tables include the lower bound and upper bound obtained by BCP, as well as the best objective value and average value achieved by the heuristic algorithms. The best results among the algorithms are highlighted in bold in these tables.

These tables show that TIEA outperforms the reference algorithms. TIEA achieves the highest number of best-known results. There is only one instance where the best-known result is not obtained by TIEA, but the difference is extremely small (0.15%), and only one algorithm can achieve the best result for this instance. Furthermore, our TIEA algorithm shows better average results compared to the reference algorithms for most instances. Additionally, the total average running time of our algorithm is the shortest. These observations confirm the effectiveness and competitiveness of our TIEA algorithm for solving these five sets of CCVRP instances.

II. Detailed results for the MDCCVRP

This appendix shows detailed comparative results of our proposed TIEA algorithm and the reference algorithms on the three MDCCVRP instance sets: lr, p, and pr with different vehicles (see Section IV-A of the main document). The results are presented in Tables A.IX–A.XI. In these tables, the first column displays the name of the instance, while the next three columns indicate the number of depots, the number of

TABLE A.I
COMPARATIVE RESULTS OF PROPOSED TIEA FOR CCVRP WITH THE REFERENCE ALGORITHMS ON THE 7 INSTANCES OF SET CMT.

Instances			SVNS [9]			ALNS [8]			UHGS [11]			AVNS [10]			BSO [2]			TIEA (this work)		
Name	n	R	f_{best}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}	f_{bst}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}
CMT1	50	5	**2230.35**	2260.03	16.17	**2230.35**	2235.27	30.29	**2230.35**	2230.35	29.04	**2230.35**	2242.47	20.21	**2230.35**	2242.09	17.64	**2230.35**	2247.12	17.63
CMT2	75	10	**2391.63**	2431.97	54.43	**2391.63**	2401.72	60.77	**2391.63**	2394.00	46.20	**2391.63**	2392.54	25.57	**2391.63**	2410.55	22.48	**2391.63**	2397.57	22.48
CMT3	100	8	**4045.42**	4103.36	192.50	**4045.42**	4063.98	172.45	**4045.42**	4045.42	61.38	**4045.42**	4068.71	69.75	**4045.42**	4061.90	60.96	**4045.42**	4046.81	60.94
CMT4	150	12	**4987.52**	5098.42	416.86	**4987.52**	4994.93	235.12	**4987.52**	4987.52	121.44	**4987.52**	4991.65	102.92	**4987.52**	5048.46	92.68	**4987.52**	4987.52	92.64
CMT5	199	17	5824.28	5920.87	416.86	5838.32	5857.76	277.37	**5806.02**	5809.94	265.32	**5806.02**	5828.57	152.57	**5806.02**	5816.06	135.36	**5806.02**	5816.45	135.32
CMT11	120	7	7317.22	7454.17	116.46	7315.87	7341.28	202.07	**7314.55**	7314.55	89.10	**7314.55**	7330.97	82.14	**7314.55**	7339.83	71.64	**7314.55**	7314.89	71.62
CMT12	100	10	**3558.92**	3581.48	65.78	**3558.92**	3566.06	152.74	**3558.92**	3558.93	42.24	**3558.92**	3560.18	58.16	**3558.92**	3558.92	53.72	**3558.92**	3558.92	55.70

TABLE A.II
COMPARATIVE RESULTS OF THE PROPOSED TIEA ALGORITHM FOR THE CCVRP WITH THE REFERENCE ALGORITHMS ON THE 20 INSTANCES OF SET GWKCT

Instances			SVNS [9]			ALNS [8]			UHGS [11]			AVNS [10]			BSO [2]			TIEA (this work)		
Name	n	R	f_{best}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}	f_{bst}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}
GWKC1	240	9	54862.24	55173.79	1165.32	54786.92	54853.76	1038.27	54739.85	54742.20	488.40	54739.80	54778.90	422.18	**54675.49**	54748.80	347.63	54739.85	54768.54	347.63
GWKC2	320	10	100998.87	101669.09	1524.52	100662.53	100934.34	1484.74	**100560.16**	100562.52	726.00	**100560.00**	100626.45	1202.94	**100560.16**	100619.90	848.55	**100560.16**	100609.76	848.55
GWKC3	400	10	171407.54	173032.55	2120.37	171613.59	172231.14	2061.75	**170923.53**	170964.42	1580.70	**170924.00**	171067.61	2103.21	**170923.55**	171093.44	1466.81	**170923.54**	171057.08	1466.81
GWKC4	480	10	263976.25	266339.67	2683.55	263433.03	265207.46	2626.89	**261993.33**	262044.19	1725.24	**261993.00**	262166.98	2517.67	262005.00	262126.13	1946.57	**261993.34**	262174.97	1946.57
GWKC5	200	5	114282.93	115552.48	1087.27	114494.66	114846.27	1200.67	**114163.63**	114163.63	458.70	**114163.64**	114264.27	428.30	**114163.64**	114163.64	349.82	**114163.63**	114169.26	349.82
GWKC6	280	7	140680.42	142416.93	769.674	140804.64	140929.10	1547.43	**140430.08**	140430.08	834.9	**140430.09**	140568.73	798.67	**140430.09**	140442.17	548.18	**140430.09**	140501.83	548.18
GWKC7	360	8	179739.77	182064.3	1574.42	180481.56	181610.82	1926.19	**178880.44**	178976.20	1759.56	179378.00	180124.94	1113.16	179479.34	180706.15	429.32	179129.79	179516.10	1113.16
GWKC8	440	10	195174.60	196216.38	2147.27	194988.74	195174.85	2330.34	**193659.14**	193683.21	1720.62	193689.00	193954.60	1814.34	193699.70	193815.40	1493.28	193686.84	193786.75	1493.28
GWKC9	255	14	4726.80	4745.07	438.00	4725.59	4728.05	864.89	4722.06	4724.01	460.02	4723.95	4730.46	356.01	**4721.39**	4724.14	310.34	**4721.39**	4723.97	310.34
GWKC10	323	16	6720.00	6744.71	985.01	6713.92	6717.76	1092.34	6713.26	6720.04	679.80	6712.53	6724.12	744.02	**6706.25**	6716.38	484.25	6712.53	6715.63	484.25
GWKC11	399	18	9229.99	9288.06	934.35	9214.07	9216.60	1356.99	9219.42	9222.92	964.26	**9210.45**	9230.59	992.44	9216.37	9225.26	709.16	**9210.32**	9216.97	932.37
GWKC12	483	19	12601.40	12754.60	1581.14	12526.17	12543.04	1540.32	12500.52	12516.98	1891.56	12506.40	12550.20	1044.77	12495.60	12554.02	829.51	**12490.93**	12530.45	829.51
GWKC13	252	26	3637.84	3662.73	653.12	3628.30	3638.50	632.72	**3627.45**	3632.63	472.56	3632.61	3654.13	210.63	3630.43	3637.91	195.68	**3627.45**	3633.75	328.08
GWKC14	320	29	5338.46	5450.88	446.47	5216.80	5257.95	682.19	**5187.56**	5206.53	1072.5	5205.75	5216.86	267.61	5209.86	5234.65	158.83	5196.47	5232.46	267.61
GWKC15	396	33	7055.50	7138.33	703.61	7010.41	7023.12	855.25	7005.47	7015.51	1234.20	7010.13	7031.42	1036.73	7014.92	7048.05	798.44	**7001.42**	7022.78	1036.73
GWKC16	480	37	9319.29	9395.76	802.94	9250.98	9268.30	1104.53	9239.10	9247.68	1800.48	9243.35	9269.51	1639.80	9244.53	9262.97	1218.37	**9238.74**	9250.90	1218.37
GWKC17	240	22	3077.59	3103.50	464.94	3065.46	3068.29	618.70	3060.14	3061.28	399.96	3063.02	3066.02	398.14	3060.50	3063.80	345.22	**3059.42**	3061.65	345.22
GWKC18	300	27	4410.46	4487.47	492.62	4221.14	4244.60	630.47	**4199.43**	4211.80	768.24	4216.01	4237.85	617.78	4223.73	4267.94	405.28	4206.01	4218.96	586.04
GWKC19	360	33	5571.80	5607.91	534.78	5523.38	5531.78	853.43	**5496.39**	5502.59	1125.96	5501.21	5534.90	767.94	5507.47	5513.75	630.62	**5496.66**	5507.88	767.94
GWKC20	420	38	7354.04	7479.58	552.33	7223.08	7240.86	881.92	**7184.19**	7188.59	1440.12	7207.88	7228.96	885.67	7203.81	7229.66	725.80	7189.15	7198.76	725.80

TABLE A.III
COMPARATIVE RESULTS OF THE PROPOSED TIEA ALGORITHM FOR THE CCVRP WITH THE REFERENCE ALGORITHMS ON THE 12 INSTANCES OF SET L

Instances			AVNS [10]			BSO [2]			TIEA (this work)		
Name	n	R	f_{best}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}
L1	560	10	374285.00	374865.40	2157.64	**373769.56**	374362.57	2055.30	**373769.53**	374116.30	2055.30
L2	600	14	218263.00	219265.10	2427.43	218313.77	218627.05	2535.72	**218255.11**	218348.33	2535.72
L3	640	10	**506252.00**	507468.06	2547.14	**506252.20**	506738.03	2390.88	**506252.21**	506794.72	2390.88
L4	720	10	**659440.00**	661788.10	2653.79	659441.23	660679.69	2718.30	659441.23	660134.29	2718.30
L5	760	17	251711.00	252270.80	3679.14	251646.09	252019.03	3600.00	**251582.73**	521733.64	3625.08
L6	800	10	834861.00	836689.50	3312.77	833388.51	834397.35	3285.42	**833336.74**	834073.14	3285.42
L7	840	19	263299.00	264877.60	3824.32	262890.15	263078.99	3600.00	**262881.75**	263026.39	3600.00
L8	880	10	1029060.00	1031879.01	4033.42	1028056.25	1031588.88	3600.00	**1027938.60**	1028366.48	3600.00
L9	960	10	1243250.00	1250196.50	4402.32	1243344.55	1244334.51	3600.00	**1243246.88**	1244002.57	4410.48
L10	1040	10	1479270.00	1481956.70	4568.07	1479775.49	1482572.72	3600.00	**1479261.66**	1480999.07	4658.34
L11	1120	10	**1735980.00**	1738516.50	5081.81	1736114.60	1739102.72	3600.00	**1735980.00**	1736531.14	5080.56
L12	1200	10	2013640.00	2016821.40	6008.65	2013640.86	2016745.78	3600.00	**2013447.82**	2015251.92	3600.00

TABLE A.IV
COMPARATIVE RESULTS OF THE PROPOSED TIEA ALGORITHM WITH LONGER RUNNING TIMES FOR CCVRP WITH THE BEST-KNOWN RESULT ON THE 32 INSTANCES OF SETS GWKC AND L

Instances			BKS	TIEA_EX
Name	n	R	f_{best}	f_{best}
GWKC1	240	9	**54675.49**	54739.85
GWKC2	320	10	**10560.16**	**100560.16**
GWKC3	400	10	**170923.55**	**170923.53**
GWKC4	180	10	**261993.00**	**261993.33**
GWKC5	200	5	**114163.64**	**114163.63**
GWKC6	280	7	**140430.09**	**140430.09**
GWKC7	360	8	**178880.44**	**178880.42**
GWKC8	440	10	**193659.14**	**193659.18**
GWKC9	255	14	**4721.39**	**4721.39**
GWKC10	323	16	**6706.25**	6712.53
GWKC11	399	18	**9210.45**	**9210.32**
GWKC12	483	19	12495.60	**12487.40**
GWKC13	252	26	**3627.45**	**3627.39**
GWKC14	320	29	**5187.56**	**5187.56**
GWKC15	396	33	7005.47	**6999.60**
GWKC16	480	37	9239.10	**9233.95**
GWKC17	240	22	3060.14	**3059.42**
GWKC18	300	27	**4199.43**	**4199.42**
GWKC19	360	33	**5496.39**	**5496.39**
GWKC20	420	38	7184.19	**7183.12**
L1	560	10	**373769.56**	**373769.51**
L2	600	14	218263.00	**218209.79**
L3	640	10	**506252.00**	**506252.150**
L4	720	10	**659440.00**	659441.19
L5	760	17	251646.09	**251460.65**
L6	800	10	8333388.51	**833336.66**
L7	840	19	262890.15	**262794.11**
L8	880	10	1028056.25	**1027938.58**
L9	960	10	1243250.00	**1243246.88**
L10	1040	10	1479270.00	**1479261.66**
L11	1120	10	**1735980.00**	**1735980.00**
L12	1200	10	2013640.00	**2013433.78**

customers, and the size of the vehicle fleet. The remaining columns provide computational information for each algorithm. Columns f_{best} and f_{avg} indicate the best objective value and the average objective value, respectively. Column T_{avg} shows the average running time for each algorithm. The reported running times have been adjusted to account for differences in CPU frequency. To maintain consistency with the running times in previous sections, a base CPU frequency of 2.0GHz was also used. The reported times are adjusted accordingly: multiplied by 1.3 for PLS (2.6GHz CPU) [12], 1.7 for BCP (3.4GHz CPU) [1], 1.25 for TIEA (2.5GHz CPU) (this work). Additionally, ILS [7] reported the total running time of the 30 independent runs in the original paper. We have transformed it into the average time per run and multiplied by 1.85 (3.7GHz). The CPU information can be seen in Table III of the main document. The BCP algorithm only reported the upper bound, the running time, and the number of used nodes. The best results for each instance among the algorithms are highlighted in bold.

TABLE A.V
SUMMARY OF THE RESULTS FOR THE CCVRP ON THE SMALL INSTANCES OF SETS A, B, E, M AND P, OBTAINED BY TIEA AND THE REFERENCE ALGORITHMS, INCLUDING THE NUMBER OF INSTANCES WHOSE BEST-KNOWN SOLUTIONS CAN BE REACHED BY AN ALGORITHM AND THE AVERAGE RUNNING TIME REQUIRED TO ATTAIN ITS BEST RESULTS.

Instances set	BCP [5]		IG-PRB [6]		ACS-VND [3]		HTS-VND [4]		TIEA (this work)	
	#BKS	T_{avg}	#BKS	T_{avg}	#BKS	T_{avg}	#BKS	T_{avg}	#BKS	T_{avg}
A(27)	26	507.97	**27**	16.27	**27**	5.59	**27**	9.33	**27**	8.25
B(22)	15	1797.29	19	17.06	21	6.10	**22**	17.17	**22**	7.76
E&M(14)	3	3257.38	12	111.06	12	24.94	9	27.81	**13**	10.04
P(22)	16	960.83	21	40.09	**22**	24.95	21	14.39	**22**	8.02
Total	60	6523.47	79	184.49	82	60.67	79	62.51	**84**	34.07

TABLE A.VI
COMPARATIVE RESULTS OF THE PROPOSED TIEA ALGORITHM FOR THE CCVRP WITH THE REFERENCE ALGORITHMS ON THE 27 SMALL INSTANCES OF SET A.

Instances				BCP [5]		IG-PRB [6]		ACS-VND [3]		HTS-VND [4]		TIEA (this work)	
Name	n	K	BKS	LB	UB	f_{best}	f_{avg}	f_{best}	f_{avg}	f_{best}	f_{avg}	f_{best}	f_{avg}
A-n32-k5	32	5	2192	2192	**2192**	**2192**	2192.00	**2192**	2192.00	**2192**	2192.00	**2192**	2192.00
A-n33-k5	33	5	1725	1725	**1725**	**1725**	1725.00	**1725**	1725.00	**1725**	1725.00	**1725**	1725.00
A-n33-k6	33	6	1612	1612	**1612**	**1612**	1612.00	**1612**	1612.00	**1612**	1612.00	**1612**	1612.00
A-n34-k5	34	5	2104	2104	**2104**	**2104**	2104.95	**2104**	2104.00	**2104**	2104.00	**2104**	2104.00
A-n36-k5	36	5	2279	2279	**2279**	**2279**	2279.00	**2279**	2279.36	**2279**	2279.00	**2279**	2279.00
A-n37-k5	37	5	1970	1970	**1970**	**1970**	1970.00	**1970**	1970.00	**1970**	1970.00	**1970**	1970.00
A-n37-k6	37	6	2241	2241	**2241**	**2241**	2241.00	**2241**	2243.20	**2241**	2241.40	**2241**	2241.00
A-n38-k5	38	5	2084	2084	**2084**	**2084**	2084.00	**2084**	2096.36	**2084**	2084.00	**2084**	2084.00
A-n39-k5	39	5	2312	2312	**2312**	**2312**	2312.00	**2312**	2321.96	**2312**	2312.00	**2312**	2312.00
A-n39-k6	39	6	2216	2216	**2216**	**2216**	2216.00	**2216**	2216.00	**2216**	2216.00	**2216**	2216.00
A-n44-k6	44	6	2563	2563	**2563**	**2563**	2563.00	**2563**	2563.00	**2563**	2563.00	**2563**	2563.00
A-n45-k6	45	6	2848	2848	**2848**	**2848**	2883.05	**2848**	2913.30	**2848**	2866.90	**2848**	2848.00
A-n45-k7	45	7	2831	2831	**2831**	**2831**	2831.00	**2831**	2831.00	**2831**	2837.00	**2831**	2831.00
A-n46-k7	46	7	2373	2373	**2373**	**2373**	2373.00	**2373**	2373.00	**2373**	2373.00	**2373**	2373.00
A-n48-k7	48	7	3101	3101	**3101**	**3101**	3101.00	**3101**	3101.00	**3101**	3101.00	**3101**	3101.00
A-n53-k7	53	7	3115	3115	**3115**	**3115**	3118.40	**3115**	3125.63	**3115**	3148.40	**3115**	3115.00
A-n54-k7	54	7	3357	3357	**3357**	**3357**	3364.45	**3357**	3384.73	**3357**	3377.30	**3357**	3357.00
A-n55-k9	55	9	2588	2588	**2588**	**2588**	2606.32	**2588**	2590.83	**2588**	2595.70	**2588**	2588.00
A-n60-k9	60	9	3446	3446	**3446**	**3446**	3447.00	**3446**	3448.36	**3446**	3452.30	**3446**	3446.00
A-n61-k9*	61	9	2868	2847	3652	**2868**	2905.15	**2868**	2837.80	**2868**	2898.90	**2868**	2869.16
A-n62-k8	62	8	3925	3925	**3925**	**3925**	3926.90	**3925**	3927.40	**3925**	3932.20	**3925**	3925.00
A-n63-k9	63	9	4630	4630	**4630**	**4630**	4633.8 0	**4630**	4642.76	**4630**	4645.20	**4630**	4630.00
A-n63-k10	63	10	3256	3256	**3256**	**3256**	3260.45	**3256**	3274.10	**3256**	3280.20	**3256**	3256.00
A-n64-k9	64	9	4135	4135	**4135**	**4135**	4138.00	**4135**	4145.86	**4135**	4148.80	**4135**	4135.00
A-n65-k9	65	9	3487	3487	**3487**	**3487**	3488.25	**3487**	3502.50	**3487**	3496.60	**3487**	3487.00
A-n69-k9	69	9	3528	3528	**3528**	**3528**	3538.55	**3528**	3531.56	**3528**	3534.20	**3528**	3528.00
A-n80-k10*	80	10	5929	5922	7174	**5929**	5936.80	**5929**	5946.26	**5929**	5943.80	**5929**	5930.42

TABLE A.VII
COMPARATIVE RESULTS OF THE PROPOSED TIEA ALGORITHM FOR THE CCVRP WITH THE REFERENCE ALGORITHMS ON THE 36 SMALL INSTANCES OF SET B, E AND M.

Instances				BCP [5]		IG-PRB [6]		ACS-VND [3]		HTS-VND [4]		TIEA (this work)	
Name	n	K	BKS	LB	UB	f_{best}	f_{avg}	f_{best}	f_{avg}	f_{best}	f_{avg}	f_{best}	f_{avg}
B-n31-k5	31	5	1830	1830	**1830**	**1830**	1830.00	**1830**	1832.13	**1830**	1830.20	**1830**	1830.00
B-n34-k5	34	5	2271	2271	**2271**	**2271**	2271.00	**2271**	2271.00	**2271**	2271.00	**2271**	2271.00
B-n35-k5	35	5	2846	2846	**2846**	**2846**	2846.00	**2846**	2846.00	**2846**	2846.00	**2846**	2846.00
B-n38-k6	38	6	2103	2103	**2103**	2164	2164.00	**2103**	2103.00	**2103**	2103.00	**2103**	2103.00
B-n39-k5*	39	5	1960	1949	1968	**1960**	1960.00	**1960**	1960.00	**1960**	1960.10	**1960**	1960.00
B-n41-k6	41	6	2329	2329	**2329**	**2329**	2329.00	**2329**	2329.00	**2329**	2329.00	**2329**	2329.00
B-n43-k6	43	6	2123	2123	**2123**	**2123**	2123.00	**2123**	2123.66	**2123**	2123.40	**2123**	2123.00
B-n44-k7	44	7	2295	2295	**2295**	**2295**	2296.60	**2295**	2295.16	**2295**	2295.50	**2295**	2295.00
B-n45-k5	45	5	2386	2386	**2386**	**2386**	2386.00	**2386**	2387.80	**2386**	2386.00	**2386**	2386.00
B-n45-k6*	45	6	2057	2017	-	**2057**	2067.70	**2057**	2072.76	**2057**	2062.30	**2057**	2057.00
B-n50-k7*	50	7	2261	2257	2293	**2261**	2261.00	**2261**	2261.00	**2261**	2261.00	**2261**	2261.00
B-n50-k8	50	8	2953	2953	**2953**	**2953**	2953.00	**2953**	2953.63	**2953**	2954.60	**2953**	2953.00
B-n51-k7	51	7	3133	3133	**3133**	**3133**	3140.90	**3133**	3137.96	**3133**	3134.30	**3133**	3133.00
B-n52-k7	52	7	2573	2573	**2573**	**2573**	2573.00	**2573**	2573.00	**2573**	2573.00	**2573**	2573.00
B-n56-k7	56	7	2358	2358	**2358**	**2358**	2358.00	**2358**	2358.60	**2358**	2358.00	**2358**	2358.00
B-n57-k7*	57	7	3883	3863	-	3885	3910.40	**3883**	3917.10	**3883**	3893.80	**3883**	3885.89
B-n57-k9	57	9	4500	4500	**4500**	**4500**	4500.00	**4500**	4500.02	**4500**	4502.30	**4500**	4500.00
B-n63-k10	63	10	4379	4379	**4379**	**4379**	4380.00	**4379**	4385.93	**4379**	4382.80	**4379**	4379.00
B-n64-k9*	64	9	2608	2597	3222	**2608**	2618.40	**2608**	2632.83	**2608**	2611.90	**2608**	2316.53
B-n66-k9*	66	9	4120	4109	4872	4132	4142.90	**4132**	4145.10	**4120**	4138.40	**4120**	4126.95
B-n67-k10*	67	10	2868	2860	2871	**2868**	2868.00	**2868**	2868.13	**2868**	2869.00	**2868**	2868.00
B-n68-k9	68	9	4058	4058	**4058**	**4058**	4058.00	**4058**	4058.07	**4058**	4058.00	**4058**	4058.00
E-n22-k4	22	4	845	845	**845**	**845**	845.00	**845**	845.00	**845**	845.00	**845**	845.00
E-n23-k3	23	3	1908	1908	**1908**	**1908**	1908.00	**1908**	1908.00	**1908**	1908.00	**1908**	1908.00
E-n30-k3*	30	3	1984	1984	1987	**1984**	1984.00	1987	1987.00	1987	1987.00	1987	1987.00
E-n33-k4	33	4	2852	2852	**2852**	**2852**	2852.00	**2852**	2852.00	**2852**	2852.00	**2852**	2852.00
E-n51-k5	51	5	2213	2213	**2213**	**2213**	2243.90	**2213**	2243.26	**2213**	2231.30	**2213**	2213.00
E-n76-k7*	76	7	2920	2905	3503	**2920**	2928.25	**2920**	2923.83	2926	2943.30	**2920**	2920.00
E-n76-k8*	76	8	2686	2672	3674	**2686**	2691.00	2686	2691.60	2687	2690.00	**2686**	2687.80
E-n76-k10*	76	10	2366	2357	3364	**2366**	2403.55	**2366**	2416.26	**2366**	2407.67	**2366**	2372.40
E-n76-k14*	76	14	2080	2075	-	2093	2105.55	**2080**	2096.26	**2080**	2093.90	**2080**	2080.00
E-n101-k8*	101	8	3954	3951	5166	**3954**	3984.55	**3954**	3975.06	**3954**	4019.80	**3954**	3954.00
E-n101-k14*	101	14	2955	2941	4022	**2955**	2963.25	**2955**	2972.26	2958	2972.40	**2955**	2955.40
M-n101-k10*	101	10	3565	3556	3566	**3565**	3570.85	**3565**	3565.03	**3565**	3569.30	**3565**	3565.00
M-n121-k7*	121	7	7223	7089	8539	7230	7273.85	7230	7318.70	**7223**	7257.80	**7223**	7238.68
M-n151-k12*	151	12	4917	4912	6314	**4917**	4991.50	**4917**	4941.16	**4917**	5010.10	**4917**	4922.11

TABLE A.VIII
COMPARATIVE RESULTS OF THE PROPOSED TIEA ALGORITHM FOR THE CCVRP WITH THE REFERENCE ALGORITHMS ON THE 22 SMALL INSTANCES OF SET P

Instances				BCP [5]		IG-PRB [6]		ACS-VND [3]		HTS-VND [4]		TIEA (this work)	
Name	n	K	BKS	LB	UB	f_{best}	f_{avg}	f_{best}	f_{avg}	f_{best}	f_{avg}	f_{best}	f_{avg}
P-n16-k8	16	8	396	396	**396**	**396**	396.00	**396**	396.00	**396**	396.00	**396**	396.00
P-n19-k2	19	2	849	849	**849**	**849**	849.00	**849**	849.00	**849**	849.60	**849**	849.00
P-n20-k2	20	2	924	924	**924**	**924**	924.00	**924**	924.00	**924**	925.30	**924**	924.00
P-n21-k2	21	2	928	928	**928**	**928**	928.00	**928**	928.00	**928**	929.50	**928**	928.00
P-n22-k8	22	8	681	681	**681**	**681**	681.00	**681**	681.00	**681**	681.00	**991**	991.00
P-n23-k8	23	8	616	616	**616**	**616**	616.00	**616**	626.40	**616**	619.50	**616**	616.00
P-n40-k5	40	5	1541	1541	**1541**	**1541**	1542.60	**1541**	1541.76	**1541**	1541.80	**1541**	1541.00
P-n45-k5	45	5	1894	1894	**1894**	**1894**	1910.95	**1894**	1894.00	**1894**	1899.00	**1894**	1984.00
P-n50-k7	50	7	1554	1554	**1554**	**1554**	1554.00	**1554**	1554.00	**1554**	1563.30	**1554**	1554.00
P-n50-k8*	50	8	1533	1528	-	1533	1552.95	**1533**	1541.30	**1533**	1548.50	**1533**	1534.63
P-n50-k10	50	10	1347	1347	**1347**	**1347**	1347.15	**1347**	1350.16	**1347**	1351.70	**1347**	1347.00
P-n51-k10	51	10	1487	1487	1487	1489	1496.25	**1487**	1495.90	**1487**	1495.90	**1487**	1488.58
P-n55-k7	55	7	1764	1764	**1764**	**1764**	1764.00	**1764**	1764.00	**1764**	1765.20	**1764**	1764.00
P-n55-k8*	55	8	1620	1768	1768	**1620**	1620.00	**1620**	1620.00	**1620**	1620.00	**1620**	1620.00
P-n55-k10	55	10	1463	1463	**1463**	**1463**	1463.00	**1463**	1463.00	**1463**	1470.70	**1463**	1463.00
P-n60-k10	60	10	1704	1704	**1704**	**1704**	1704.00	**1704**	1708.86	**1704**	1714.60	**1704**	1704.00
P-n60-k15	60	15	1509	1509	**1509**	**1509**	1510.10	**1509**	1513.30	**1509**	1513.40	**1509**	1509.32
P-n65-k10	65	10	1948	1948	**1948**	**1948**	1956.30	**1948**	1957.76	**1948**	1961.60	**1948**	1948.95
P-n70-k10	70	10	2121	2121	**2121**	**2121**	2147.80	**2121**	2165.56	**2121**	2137.20	**2121**	2121.00
P-n76-k4*	76	4	4610	4553	6191	**4610**	4645.85	**4610**	4658.26	**4610**	4644.30	**4610**	4635.42
P-n76-k5*	76	5	3795	3775	5463	**3795**	3643.15	**3795**	38589.66	**3795**	3849.50	**3795**	3801.63
P-n101-k4*	101	4	6943	6807	8707	**6943**	7010.55	**6943**	6974.06	6946	7031.70	**6943**	6961.89

REFERENCES

[1] C. M. Damião, J. M. P. Silva, and E. Uchoa. A branch-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. *4OR-Q. J. Oper. Res.*, 21:47–71, 2023.

[2] L. Ke. A brain storm optimization approach for the cumulative capacitated vehicle routing problem. *Memet. Comput.*, 10:411–421, 2018.

[3] N. A. Kyriakakis, M. Marinaki, and Y. Marinakis. A hybrid ant colony optimization-variable neighborhood descent approach for the cumulative capacitated vehicle routing problem. *Comput. Oper. Res.*, 134:105397, 2021.

[4] N. A. Kyriakakis, I. Sevastopoulos, M. Marinaki, and Y. Marinakis. A hybrid tabu search–variable neighborhood descent algorithm for the cumulative capacitated vehicle routing problem with time windows in humanitarian applications. *Comput. Ind. Eng.*, 164:107868, 2022.

[5] J. Lysgaard and S. Wøhlk. A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. *Eur. J. Oper. Res.*, 236(3):800–810, 2014.

[6] S. Nucamendi-Guillén, F. Angel-Bello, I. Martínez-Salazar, and A. E. Cordero-Franco. The cumulative capacitated vehicle routing problem: New formulations and iterated greedy algorithms. *Expert Syst. Appl.*, 113:315–327, 2018.

[7] A. Osorio-Mora, J. W. Escobar, and P. Toth. An iterated local search algorithm for latency vehicle routing problems with multiple depots. *Comput. Oper. Res.*, 158:106293, 2023.

[8] G. M. Ribeiro and G. Laporte. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Comput. Oper. Res.*, 39(3):728–735, 2012.

[9] N. Smiti, M. M. Dhiaf, B. Jarboui, and S. Hanafi. Skewed general variable neighborhood search for the cumulative capacitated vehicle routing problem. *Int. Trans. Oper. Res.*, 27(1):651–664, 2020.

[10] J. F. Sze, S. Salhi, and N. Wassan. The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: An effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search. *Transp. Res. Pt. B-Methodol.*, 101:162–184, 2017.

[11] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. A unified solution framework for multi-attribute vehicle routing problems. *Eur. J. Oper. Res.*, 234(3):658–673, 2014.

[12] X. Wang, T.-M. Choi, Z. Li, and S. Shao. An effective local search algorithm for the multidepot cumulative capacitated vehicle routing problem. 50(12):4948–4958, 2019.

TABLE A.IX
COMPARATIVE RESULTS OF THE PROPOSED TIEA ALGORITHM FOR THE MDCCVRP WITH THE REFERENCE ALGORITHMS ON THE 21 INSTANCES OF SET LR (I.E., S_1).

Instances				PLS [12]			BCP [1]			ILS [7]			TIEA-1000 (this work)			TIEA-5000 (this work)		
Name	$\|D\|$	$\|C\|$	K	f_{best}	f_{avg}	T_{avg}	UB	Time	#Nodes	f_{best}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}
lr01	4	10	5	**545.69**	554.97	0.20	**545.69**	0.05	1	**545.69**	546.13	0.80	**545.69**	545.69	0.09	**545.69**	545.69	0.45
lr02	4	10	5	**832.69**	847.80	0.18	**832.69**	0.05	1	**832.69**	832.69	2.34	**832.69**	832.69	0.09	**832.69**	832.69	0.42
lr03	4	10	5	**832.78**	841.47	0.18	**832.78**	0.10	1	**832.78**	832.78	1.30	**832.78**	832.78	0.09	**832.78**	832.78	0.45
lr04	4	25	10	**2082.28**	2099.58	0.82	**2082.28**	1.73	1	**2082.28**	2082.57	23.25	**2082.28**	2082.28	0.37	**2082.28**	2082.28	1.78
lr05	4	25	10	**1827.41**	1870.90	0.85	**1827.41**	1.72	1	**1827.41**	1837.30	20.66	**1827.41**	1827.41	0.41	**1827.41**	1827.41	1.91
lr06	4	25	10	**1786.95**	1808.86	0.85	**1786.95**	1.65	1	**1786.95**	1786.95	14.12	**1786.95**	1786.95	0.42	**1786.95**	1786.95	2.01
lr07	4	50	20	**5424.57**	5440.37	3.54	**5424.57**	32.13	1	**5424.57**	5424.57	85.22	**5424.57**	5424.57	0.94	**5424.57**	5424.57	4.29
lr08	4	50	20	**3737.38**	3759.67	3.24	**3737.38**	28.22	1	**3737.38**	3743.96	43.97	**3737.38**	3737.38	1.00	**3737.38**	3737.38	4.34
lr09	4	50	20	**3802.88**	3811.65	3.49	**3802.88**	25.50	1	**3802.88**	3808.52	51.68	**3802.88**	3802.90	1.02	**3802.88**	3802.88	4.88
lr10	6	50	20	-	-	-	**2969.83**	21.76	1	**2969.83**	2992.04	61.67	**2969.84**	2969.84	1.02	**2969.84**	2969.84	4.96
lr11	6	50	20	-	-	-	**3095.22**	33.83	1	**3095.22**	3120.25	33.61	**3095.22**	3095.49	1.02	**3095.22**	3095.22	5.01
lr12	6	50	20	-	-	-	**3171.75**	24.99	1	3174.98	3192.62	50.20	**3171.75**	3171.75	1.03	**3171.75**	3171.75	5.01
lr10	6	50	25	2868.39	2883.50	7.93	**2866.73**	20.23	1	2867.28	2874.88	62.22	**2866.73**	2866.73	1.12	**2866.73**	2866.73	5.14
lr11	6	50	25	2987.75	3008.88	8.28	**2978.78**	33.83	1	2979.46	2992.10	49.52	**2978.78**	2978.78	1.13	**2978.78**	2978.78	5.21
lr12	6	50	25	3095.52	3112.60	8.00	**3090.38**	24.99	1	**3090.38**	3095.64	55.25	**3090.38**	3090.38	1.13	**3090.38**	3090.38	5.23
lr13	4	100	25	8293.42	8331.27	7.94	**8288.43**	351.90	1	**8288.43**	8324.16	116.67	**8288.43**	8288.43	3.03	**8288.43**	8288.43	14.67
lr14	4	100	25	7273.03	7353.68	7.15	**7257.31**	358.70	1	**7257.31**	7272.96	118.71	**7257.31**	7257.31	3.04	**7257.31**	7257.31	14.25
lr15	4	100	25	**8626.13**	8644.64	9.45	**8626.13**	462.70	1	**8626.13**	8643.77	159.35	**8626.13**	8626.13	3.15	**8626.13**	8626.13	15.49
lr16	6	100	25	5306.50	5483.36	9.15	**5265.30**	261.80	1	**5265.3**	5281.53	86.77	**5265.31**	5265.31	3.34	**5265.31**	5265.31	18.45
lr17	6	100	25	6141.07	6265.57	9.82	**6107.32**	299.20	1	6107.32	6112.13	101.01	**6107.32**	6107.32	3.03	**6107.32**	6107.32	15.04
lr18	6	100	25	5804.19	5905.71	10.18	**5788.73**	317.90	1	**5788.73**	5811.51	96.32	**5788.73**	5788.73	3.17	**5788.73**	5788.73	15.03
Avg	-	-	-	3959.37	4001.36	5.07	3827.55	109.67	-	3827.76	3838.53	58.77	3827.55	3827.56	1.41	3827.55	3827.55	6.86

TABLE A.X
COMPARATIVE RESULTS OF THE PROPOSED TIEA ALGORITHM FOR THE MDCCVRP WITH THE REFERENCE ALGORITHMS ON THE 24 INSTANCES OF SET P AND SET PR WITH $|K|$=35 (I,E., S_2).

Instances				PLS [12]			BCP [1]			ILS [7]			TIEA-1000 (this work)			TIEA-5000 (this work)		
Name	$\|D\|$	$\|C\|$	K	f_{best}	f_{avg}	T_{avg}	UB	Time	#Nodes	f_{best}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}
p01	4	50	35	713.18	717.44	2.91	**712.49**	4.40	1	**712.50**	714.18	119.03	**712.50**	712.58	0.94	**712.50**	712.50	4.75
p02	4	50	35	713.59	716.78	2.89	**712.49**	8.50	1	**712.50**	713.65	115.66	**712.50**	712.57	0.94	**712.50**	712.50	4.68
p03	5	75	35	952.17	959.03	4.38	**950.25**	26.35	1	**950.25**	953.90	114.66	**950.25**	950.46	1.77	**950.25**	950.25	8.51
p04	2	100	35	1955.82	1959.14	6.79	**1955.31**	86.19	1	**1955.31**	1955.97	224.87	**1955.31**	1955.42	2.84	**1955.31**	1955.31	14.51
p05	2	100	35	1985.03	1988.46	6.66	**1982.33**	132.60	1	**1982.35**	1984.17	195.947	**1982.33**	1982.68	2.87	**1982.33**	1982.40	13.91
p06	3	100	35	1553.88	1563.22	5.94	**1552.13**	63.41	1	**1552.13**	1553.88	195.16	**1552.13**	1552.17	2.97	**1552.13**	1552.85	14.11
p07	4	100	35	1522.68	1528.43	6.89	**1520.46**	60.18	1	**1520.97**	1524.03	194.95	**1520.46**	1520.74	2.92	**1520.46**	1520.51	14.26
p08	2	249	35	15410.92	15458.51	26.60	**15372.60**	2125.00	1	**15372.60**	15400.51	490.62	**15372.60**	15386.35	16.04	**15372.60**	15374.68	70.85
p09	3	249	35	13136.24	13335.06	24.18	**13070.74**	1841.10	1	13071.60	13105.79	463.84	**13070.74**	13080.71	19.16	**13070.74**	13074.71	76.65
p10	4	249	35	12096.90	12432.72	24.23	**12052.56**	1666.00	1	12070.50	12155.06	467.28	**12052.56**	12092.55	26.33	**12052.56**	12086.63	119.70
p11	5	249	35	12033.48	12228.64	22.54	**11955.58**	2119.90	1	**11995.9**	12051.88	449.98	**11955.58**	11993.88	17.55	**11955.58**	11970.96	69.30
p12	2	80	35	**2897.06**	2897.06	2.63	**2897.06**	39.10	1	**2897.06**	2897.06	87.95	**2897.06**	2897.06	1.79	**2897.06**	2897.06	8.64
p15	4	160	35	**5794.11**	5794.11	6.76	**5794.11**	389.30	1	**5794.11**	5794.11	184.22	**5794.11**	5794.11	5.08	**5794.11**	5794.11	24.15
p18	6	240	35	11469.49	11546.91	16.03	**11433.91**	2029.80	1	11453.50	11476.90	309.81	11436.96	11451.50	13.15	**11433.91**	11441.38	56.67
pr01	4	48	35	1261.81	1264.74	2.67	**1261.53**	6.92	1	1262.43	1266.58	81.15	**1261.53**	1261.56	0.88	**1261.53**	1261.53	4.22
pr02	4	96	35	2572.84	2580.94	6.27	**2572.84**	85.51	1	**2572.84**	2574.88	168.93	**2572.84**	2572.92	2.61	**2572.84**	2572.84	12.53
pr03	4	144	35	4466.10	4511.37	9.78	**4462.50**	277.10	1	4464.91	4475.65	283.45	**4462.50**	4466.06	5.58	**4462.50**	4463.71	25.54
pr04	4	192	35	5813.87	5863.56	16.60	**5804.15**	919.70	1	5813.33	5825.20	475.25	**5804.15**	5812.07	9.59	**5804.15**	5807.01	46.20
pr05	4	240	35	7157.06	7225.66	26.43	**7120.22**	2145.40	1	7122.06	7146.94	707.98	7121.36	7130.92	15.75	**7120.22**	7123.25	71.40
pr06	4	288	35	8685.68	8874.53	35.48	**8603.85**	3927.00	1	8607.46	657.63	814.71	8604.68	8619.09	22.69	**8603.85**	8608.32	97.65
pr07	6	72	35	1727.25	1736.14	4.63	**1723.63**	29.75	1	1725.55	1727.74	133.35	**1723.63**	1724.08	1.69	**1723.63**	1723.68	1.96
pr08	6	144	35	4023.21	4046.68	10.66	**4004.11**	568.60	1	**4004.11**	4015.78	272.03	**4004.11**	4005.32	5.76	**4004.11**	4004.11	27.03
pr09	6	216	35	5937.19	6043.29	19.08	**5889.02**	1161.10	1	**5899.64**	5932.25	501.62	**5889.02**	5889.33	12.53	**5889.02**	5892.12	59.68
pr10	6	288	35	9166.57	9336.73	37.18	**9113.49**	3525.80	1	9135.86	9177.17	734.67	9120.00	9140.06	22.20	**9113.49**	9124.42	89.25
Avg	-	-	-	5543.59	5608.71	13.68	5521.56	968.28	-	5527.06	5545.04	324.46	5522.04	5529.34	8.90	5521.56	5525.29	39.01

TABLE A.XI
COMPARATIVE RESULTS OF THE PROPOSED TIEA ALGORITHM FOR THE MDCCVRP WITH THE REFERENCE ALGORITHMS ON THE 33 INSTANCES OF SET P AND PR WITH THE ORIGINAL NUMBER OF VEHICLES (I.E., S_3).

Instances				PLS [12]			BCP [1]			ILS [7]			TIEA-1000 (this work)			TIEA-5000 (this work)		
Name	$\|D\|$	$\|C\|$	K	f_{best}	f_{avg}	T_{avg}	UB	Time	#Nodes	f_{best}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}	f_{best}	f_{avg}	T_{avg}
p01	4	50	11	**1055.35**	1095.44	1.90	**1055.35**	4.79	1	**1055.35**	1071.27	27.84	**1055.35**	1058.59	1.56	**1055.35**	1057.30	8.25
p02	4	50	4	2055.40	2149.54	4.12	**2016.18**	101.66	1	**2016.18**	2047.63	37.76	**2016.18**	2023.17	2.71	**2016.18**	2019.85	10.52
p03	5	75	11	1758.70	1809.51	4.24	**1749.29**	58.48	1	1762.49	1799.85	64.52	**1749.29**	1755.04	3.07	**1749.29**	1752.62	14.84
p04	2	100	15	2618.88	2703.37	10.89	**2537.59**	1696.60	15	2552.67	2635.03	154.41	2538.54	2564.27	6.43	**2537.59**	2550.86	22.20
p05	2	100	8	3766.20	3824.49	12.92	**3749.92**	2128.40	3	3754.39	3790.63	118.91	**3749.92**	3760.48	5.60	**3749.92**	3755.14	26.89
p06	3	100	16	2160.93	2188.17	6.73	**2131.61**	181.90	1	2132.97	2164.93	148.14	**2131.62**	2140.54	5.26	**2131.62**	2135.02	23.19
p07	4	100	16	2142.11	2181.17	7.77	**2108.89**	268.60	3	2140.15	2183.26	141.14	**2108.89**	2127.95	5.16	**2108.89**	2123.30	22.89
p08	2	249	25	17393.46	17862.13	45.76	-	-	-	17272.00	17516.46	435.43	17075.28	17179.30	36.93	**17060.64**	17156.00	164.05
p09	3	249	26	15041.69	15448.98	35.01	-	-	-	14918.00	15021.94	414.34	**14774.18**	14846.98	30.82	**14774.18**	14828.52	146.22
p10	4	249	26	14265.77	14619.35	37.12	**14024.58**	31478.90	35	14089.20	14322.70	418.58	14037.15	14145.40	31.12	**14024.58**	14103.66	138.10
p11	5	249	26	14381.43	14552.90	36.86	-	-	-	14161.20	14404.70	402.28	14007.82	14123.58	32.11	**14005.41**	14085.59	142.33
p12	2	80	8	**5494.36**	5536.40	5.07	**5494.36**	142.97	1	**5494.36**	5495.85	69.67	**5494.36**	5495.96	3.33	**5494.36**	5494.36	17.65
p13	4	160	9	**4914.66**	4926.54	5.27	**4914.66**	108.80	1	**4914.66**	4914.83	59.82	**4914.66**	4914.66	2.91	**4914.66**	4914.66	14.13
p14	4	160	10	4510.12	4512.28	5.04	**4491.64**	101.15	1	**4491.64**	4492.13	55.15	**4491.64**	4493.63	2.62	**4491.64**	4491.64	13.08
p15	4	160	16	10662.27	10747.26	16.37	**10590.41**	1210.40	3	**10629.80**	10676.81	163.60	10629.82	10651.73	9.86	**10590.41**	10638.21	47.88
p16	4	160	17	10086.50	10122.13	13.92	**10008.28**	1416.10	17	10016.10	10070.56	157.15	10014.34	10047.28	9.21	**10008.28**	10051.90	42.40
p17	4	160	18	9538.69	9573.86	13.78	**9493.84**	710.60	1	9495.91	9518.63	152.05	**9493.84**	9511.04	8.29	**9493.84**	9507.26	40.57
p18	6	240	24	15912.27	16072.75	26.73	**15720.73**	21319.70	49	15847.80	15948.26	307.01	15741.73	15812.29	20.52	**15720.73**	15764.39	99.82
p19	4	160	25	15255.02	15370.98	24.64	**15105.26**	3622.70	7	15224.80	15301.79	292.88	15166.34	15201.09	16.45	**15105.26**	15169.30	79.02
p20	4	160	26	14709.23	14786.87	23.71	**14592.52**	3587.00	5	14635.70	14708.73	292.07	14619.53	14702.56	17.21	**14592.52**	14640.45	82.25
p21	4	160	34	25770.35	26102.29	51.51	-	-	-	25500.80	25892.53	677.23	24949.01	25188.11	70.50	**24941.00**	25165.28	310.23
p22	4	160	35	24451.01	24816.85	47.92	-	-	-	24330.70	24608.60	637.76	24138.89	24279.95	53.38	**24010.85**	24209.32	238.35
p23	4	160	36	23656.13	23925.16	42.48	-	-	-	23622.80	23784.41	594.36	23399.18	23539.45	42.34	**23372.41**	23492.21	197.22
pr01	4	48	4	**3748.11**	3773.27	4.38	**3748.11**	88.04	1	3768.69	3768.69	27.41	**3748.11**	3748.24	1.88	**3748.11**	3748.11	9.09
pr02	4	96	8	4973.36	5000.74	12.06	**4834.46**	720.80	1	**4834.46**	4854.38	60.64	**4834.46**	4849.04	4.35	**4834.46**	4844.04	21.30
pr03	4	144	11	8357.54	8470.56	24.38	**8353.05**	3177.30	1	8357.54	8424.14	120.60	**8353.05**	8378.36	9.19	**8353.05**	8370.30	43.26
pr04	4	192	14	9273.89	9585.47	51.73	**9071.44**	93632.60	121	9156.38	9324.65	273.28	9082.78	9219.44	19.43	**9071.45**	9181.97	90.36
pr05	4	240	19	10075.01	10283.54	68.72	-	-	-	9805.38	10025.37	462.67	9575.39	9779.45	42.01	**9469.51**	9707.03	179.11
pr06	4	288	23	11053.59	11234.57	74.10	-	-	-	10873.00	10996.31	550.99	10832.07	10900.33	35.00	**10799.16**	10864.82	166.76
pr07	6	72	6	4877.86	4906.62	24.38	**4760.65**	469.20	1	**4760.65**	4787.68	45.13	**4760.65**	4777.34	3.23	**4760.65**	4768.90	16.84
pr08	6	144	12	7141.88	7265.17	48.01	**6997.11**	3162.00	1	7049.50	7131.65	182.60	**6997.11**	7040.43	10.55	**6997.11**	7016.28	48.01
pr09	6	216	17	9235.29	9350.34	70.27	**9027.82**	48681.20	45	9147.07	9327.58	345.23	9069.84	9173.76	21.57	**9027.82**	9134.50	106.34
pr10	6	288	24	11643.69	11811.29	30.85	-	-	-	11335.10	11493.69	557.38	11207.50	11347.24	39.79	**11188.54**	11266.93	182.07
Avg	-	-	-	9757.00	9897.27	26.93	-	9086.25	-	9671.13	9772.90	256.00	9598.74	9991.90	18.31	9584.01	9636.66	83.79