A Memetic Algorithm for Vehicle Routing with Simultaneous Pickup and Delivery and Time Windows

Zhenyu Lei and Jin-Kao Hao*

IEEE Transactions on Evolutionary Computation 29(5): 1924-1936, 2025

***Abstract*—The Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time Windows (VRPSPDTW) has a number of real-world applications, especially in reverse logistics. In this work, we propose an effective memetic algorithm that integrates a lightweight feasible and infeasible route descent search and a learning-based adaptive route-inheritance crossover to solve this complex problem. We evaluate the effectiveness of the proposed algorithm on the set of 65 popular benchmark instances as well as 20 real-world large-scale benchmark instances. We provide a comprehensive analysis to better understand the design and performance of the proposed algorithm.**

***Index Terms*—Vehicle routing; Simultaneous pickup and delivery with time windows; Combinatorial optimization; Heuristics; Memetic algorithm.**

I. Introduction

THE Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time Windows (VRPSPDTW) [1] is a member of the large family of Vehicle Routing Problems (VRPs) [2]. Specifically, VRPSPDTW is a variant of the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD) [3] that incorporates time window constraints. This is a computationally challenging problem because it can be trivially reduced to the NP-hard VRPSPD problem [1].

VRPSPDTW has a wide range of applications in real world scenarios, especially in the field of logistics. Many logistics companies now offer pickup services in addition to delivery services to improve transportation efficiency and reduce costs, a concept commonly referred to as reverse logistics [4].

VRPSPDTW can be described on a directed complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertices $\mathcal{V} = \{v_0, v_1, \ldots, v_\mathcal{N}\}$ consist of the depot node v_0 and the $\mathcal{N}$ customer nodes $(v_1, \ldots, v_\mathcal{N})$. The edges $\mathcal{E} = \{e_{ij} | v_i, v_j \in \mathcal{V}\}$ represent connections between the nodes. Each customer node $v_i \in \mathcal{V}$ $(i \neq 0)$ is associated with a delivery demand d_i and a pickup demand p_i. This implies that the vehicle must deliver d_i units of goods from the depot v_0 to v_i and pick up p_i units of goods from v_i to the depot v_0. Additionally, each node v_i has a time window $[e_i, l_i]$ (the earliest and latest time to start the service at node v_i) and a service time s_i (the time the vehicle spends at node v_i). The depot time window $[e_0, l_0]$ specifies the earliest time the vehicle can leave the depot and the latest time it must return to the depot, with the service time s_0 equal to 0. Furthermore, each edge $e_{ij} \in \mathcal{E}$ is associated with a travel distance c_{ij} and a travel time t_{ij}.

A fleet of M homogeneous vehicles with a capacity of $\mathcal{Q}$ is available to serve the given customers. VRPSPDTW aims to determine the best routes of the vehicles starting and ending at depot v_0 while satisfying the capacity and time window constraints. Thus, a solution S of VRPSPDTW is a set of closed routes $S = \{R_1, \ldots, R_M\}$, where each route R_i consists of a sequence of nodes $\{n_{i,0}, n_{i,1}, \ldots, n_{i,\mathcal{L}}, n_{i,\mathcal{L}+1}\}$ visited by the i-th vehicle. Here, $n_{i,j}$ indicates the j-th visiting node in route R_i, and $\mathcal{L}$ indicates the number of customers in route R_i. Notably, both the first and last nodes in route R_i are the depot v_0 (i.e., $n_{i,0} = v_0$ and $n_{i,\mathcal{L}+1} = v_0$). The load of each vehicle cannot exceed the capacity $\mathcal{Q}$, thus $q_{n_{i,j}} < \mathcal{Q}$ must be satisfied for all $n_{i,j} \in R_i$, where $q_{n_{i,j}}$ is the load of the i-th vehicle after visiting node $n_{i,j}$. Additionally, let $a_{n_{i,j}}$ be the arrival time at node $n_{i,j}$. Arriving before $e_{n_{i,j}}$ $(a_{n_{i,j}} < e_{n_{i,j}})$ results in a wait time $w_{n_{i,j}} = \max\{e_{n_{i,j}} - a_{n_{i,j}}, 0\}$. Conversely, arriving after $l_{n_{i,j}}$ $(a_{n_{i,j}} > l_{n_{i,j}})$ is considered infeasible.

Let μ_1 and μ_2 be the costs or weights assigned to the vehicles and the travel distance, respectively. A typical objective function of VRPSPDTW is then to minimize the total cost, which is the weighted sum of the number of vehicles and the travel distance, as shown in Equation (1). Another typical hierarchical objective function aims to minimize first the number of vehicles used (primary objective) and then the travel distance of the vehicles (secondary objective). This can be achieved by assigning a very high cost μ_1 to vehicles and a relatively low cost μ_2 to travel distances.

$$\begin{aligned} \text{Minimize} \quad & f(S) = \mu_1 \cdot M + \mu_2 \cdot D(S) \\ \text{Subject to} \quad & D(S) = \sum_{i=1}^{M} \sum_{j=0}^{\mathcal{L}} c_{n_{i,j} n_{i,j+1}} \\ & S = \{R_1, \ldots, R_M\} \\ & R_i = \{n_{i,0}, n_{i,1}, \ldots, n_{i,\mathcal{L}}, n_{i,\mathcal{L}+1}\}, \quad i = 1, \ldots, M \end{aligned} \tag{1}$$

Since the introduction of VRPSPDTW [1], many algorithms have been developed to solve this problem. Section II provides a comprehensive review of related studies. The inherent complexity of the problem is such that no single method consistently excels across all benchmark instances.

This work benefited from the computing facilities provided by the Centre de Calcul Intensif des Pays de la Loire (CCIPL). Support for the first author from the China Scholar Council (Grant No. 202206330014) is acknowledged. We thank Prof. Shi of [17] and Dr. Wu of [7] for their help in answering our questions about their works.

Z. Lei and J.K. Hao (Corresponding author) are with the Department of Computer Science, LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers Cedex 01, France (e-mails: zhenyu.lei@etud.univ-angers.fr, jin-kao.hao@univ-angers.fr).

Additionally, research on large instances and real-world applications remains limited. Meanwhile, a notable trend in recent algorithm design is the use of large neighborhood search techniques with destroy and repair operators [5, 6, 7]. While these strategies enhance search space exploration, they also increase algorithmic complexity.

To address these challenges, we propose a novel algorithm, MA-FIRD, which is based on the memetic algorithm framework. This framework has been proven effective for solving various VRPs, including VRPTW [8], split delivery VRP [9], and multi-trip VRP [10]. One recent work [6] has also showed the effectiveness of the memetic algorithm approach for VRPSPDTW. Following this framework, our proposed algorithm integrates several novel components and strategies. These include a lightweight feasible and infeasible route descent search where a dedicated penalty function is embedded to manage infeasible solutions, and a multi-parent route-inheritance crossover supported by reinforcement learning. Furthermore, we implement a max-min normalization-based fitness-distance population management strategy to maintain population diversity. The proposed algorithm is evaluated on the popular WC benchmark [11] and the large-scale real-world JD benchmark [6], demonstrating its superiority and practicality compared to existing algorithms.

The rest of this paper is organized as follows. Section II provides a literature review on VRPSPDTW. Section III presents the proposed algorithm. Section IV shows computational results on benchmark instances compared to state-of-the-art methods, highlighting its ability to solve real-world large-scale instances. Section V provides an in-depth analysis of the key components of the algorithm. Section VI concludes the paper and suggests possible avenues for future research.

II. Literature review

TABLE I
SUMMARY OF THE ALGORITHMS FOR VRPSPDTW IN THE LITERATURE.

Literature	Year	Method	Objective
Angelelli and Mansini [1]	2002	Branch-and-Price	travel cost
Liang et al. [12]	2009	GA	total cost with penalty
Lai and Cao [13]	2010	DE	travel cost
Wang and Chen [11]	2012	co-GA	hierarchical objective
Kassem and Chen [14]	2013	Insertion heuristic, SA	travel cost
Wang et al. [15]	2013	SA	hierarchical objective
Wang et al. [16]	2015	parallel SA	hierarchical objective
Hof and Schneider [5]	2019	ALNS, Path-relinking	hierarchical objective
Shi et al. [17]	2020	VNS, TS	hierarchical objective
Tang et al. [18]	2021	Co-evolution of parameterized search	weighted total cost
Liu et al. [6]	2021	MA	hierarchical objective
Wu and Gao [7]	2023	ACO	weighted total cost
Praxedes et al. [19]	2024	Branch-Cut-and-Price	travel cost

As a variant of the well-studied VRPSPD problem [3, 20], the first study of VRPSPDTW can be traced back to the work of Angelelli and Mansini [1]. Their pioneering study introduced a Branch-and-Price algorithm based on a set covering formulation to address the problem, with the objective of minimizing the total travel cost of the vehicles. Table I summarizes the most important solution approaches that have been carried out in the literature on VRPSPDTW since then.

Regarding the objective of minimizing the total travel cost only, Lai and Cao [13] presented an improved differential evolution algorithm (IDE), where they devised a novel decimal encoding method to represent solutions and introduced a penalty function to deal with illegal solutions. Kassem and Chen [14] presented the INST-SA heuristic, using an insertion heuristic for solution initialization, and simulated annealing and neighborhood search for solution improvement. Tang et al. [18] studied VRPSPDTW with the objective of minimizing the weighted total cost of the vehicle dispatching and the travel distance. They proposed a co-evolution of parameterized search (CEPS) to achieve generalizable parallel algorithm porfolio (PAP) based on some training instances. Liang et al. [12] studied a variant of VRPSPDTW with soft time window constraints, allowing delayed arrivals beyond the time windows with associated penalty costs.

Until the work of Wang and Chen [11], there was a lack of benchmark instances for VRPSPDTW, making it difficult to compare the solution algorithms for the problem. Wang and Chen filled the gap by introducing a set of 65 benchmark instances (denoted as WC) derived from the Solomon VRPTW benchmark [21]. They also proposed a co-evolution genetic algorithm (co-GA), which maintains two populations for diversification and intensification, to minimize the hierarchical objective of the number of vehicles first and the travel distance second. Following their study on the hierarchical objective, a succession of research efforts using metaheuristics [15, 16, 5, 17, 6] have contributed to advancing the state of the art in solving VRPSPDTW, particularly in the context of the WC benchmark.

Wang et al. [15] introduced a simulated annealing approach and later developed a parallel simulated annealing (p-SA) algorithm [16], utilizing multi-processor or multi-thread capabilities to enhance and accelerate the search process. Shi et al. [17] presented a two-stage algorithm VNS-BSTS. In the first stage, a Variable Neighborhood Search (VNS) was used to minimize the number of vehicles, featuring a novel learning-based evaluation function to assess each move. The second stage employed a Bi-Structure based Tabu Search (BSTS) to intensify the optimization. Recently, Wu and Gao [7] proposed an ant-colony optimization algorithm with destroy and repair strategies (ACO-DR) for VRPSPDTW, focusing on minimizing the weighted total cost of vehicle dispatching and travel distance.

Hof and Schneider [5] proposed an Adaptive Large Neighborhood Search with Path Relinking (ALNS-PR) algorithm for a class of VRPSPD, including VRPSPDTW. The algorithm involves 7 removal operators and 4 insertion operators, dynamically selected through an adaptive mechanism to iteratively destroy and repair the solution. Moreover, it incorporates a path-relinking component to explore promising search spaces between elite solutions. The algorithm temporarily accepts infeasible solutions by incorporating penalty terms into the objective function. Results showed the robustness of the ALNS-PR algorithm in solving this group of problems with competitive results on VRPSPDTW.

Liu et al. [6] proposed the Memetic Algorithm with Extended Neighborhoods (MATE). This innovative approach employs local search for small-step exploration and removal-reinsertion of large neighborhoods for large-step exploration, effectively navigating the search space. To generate promising solutions, MATE incorporates a RARI crossover com-

bined with regret insertion. Experiments on the WC instances showed the efficacy of the MATE algorithm by discovering 12 new best solutions. The authors also generated a new set of 20 large-scale benchmark instances, denoted as JD benchmark, derived from the real-world JD logistics network. Testing on the JD instances confirmed MATE's effectiveness in solving VRPSPDTW in real-world scenarios.

Finally, Praxedes et al. [19] introduced the exact Branch Cut and Price (BCP) algorithm to solve a broad class of VRPSPD, including VRPSPDTW. With a 12-hour time limit, BCP achieved 45 optimal solutions out of 65 WC instances for minimizing travel cost only.

Given their superior performance on popular benchmarks compared to other approaches, ALNS-PR [5] and MATE [6] are considered state-of-the-art methods for solving VRPSPDTW.

III. Memetic Algorithm with Feasible and Infeasible Route Descent Search for VRPSPDTW

The memetic algorithm framework [22, 23] combines the advantages of genetic algorithms and local search methods, providing an interesting way to balance search diversification and intensification. Leveraging this powerful framework, we design dedicated search operators and strategies to address the diverse features and constraints of VRPSPDTW. Specifically, we propose a lightweight feasible and infeasible route descent search, a learning-based adaptive route-inheritance crossover, and a max-min normalization-based fitness-distance population management strategy to enable efficient search.

Algorithm 1 Main framework of MA-FIRD

1: **Input**: Instance $\mathcal{I}$, Population size $\mathcal{N}_\mathcal{P}$, Maximum number of generations φ_{max}, Patience for stagnation generations ρ, Maximum running time τ.
2: **Output**: The best solution S^*.
3: $\varphi \leftarrow 0, \varphi_{st} \leftarrow 0$ /* Current generation and stagnation generation counter */
4: $\mathcal{I} \leftarrow Preprocessing(\mathcal{I})$ /* Section III-A */
5: $\mathcal{P} \leftarrow Initialization(\mathcal{I}, \mathcal{N}_\mathcal{P})$ /* Section III-B */
6: $S^* \leftarrow BestSolution(\mathcal{P})$ /* Record current best solution */
7: **while** $\varphi \leq \varphi_{max}$ **and** $\varphi_{st} \leq \rho$ **and** $time() \leq \tau$ **do**
8: $S' \leftarrow ARIX(\mathcal{P})$ /* Section III-D */
9: $S' \leftarrow FIRDSearch(S')$ /* Section III-C */
10: $\mathcal{P} \leftarrow UpdatePopulation(S', \mathcal{P})$ /* Section III-F */
11: $S_{best} \leftarrow BestSolution(\mathcal{P})$ /* The best solution in this generation */
12: **if** $f(S_{best}) < f(S^*)$ **then**
13: $S^* \leftarrow S_{best}$ /* Update the best solution */
14: $\varphi_{st} \leftarrow 0$
15: **else**
16: $\varphi_{st} \leftarrow \varphi_{st} + 1$ /* Stagnation counter is incremented */
17: **end if**
18: $\varphi = \varphi + 1$
19: **end while**
20: **return** S^*

Algorithm 1 outlines the framework of MA-FIRD. For a given problem instance $\mathcal{I}$, the algorithm starts with a preprocessing step to reduce the instance (line 4). Then, $\mathcal{N}_\mathcal{P}$ candidate solutions are generated to form the initial population (line 5), and the best solution S^* is recorded (line 6). The population is updated iteratively by generating new candidate solutions (lines 7-19). In each generation, the adaptive route-inheritance crossover creates a new solution from multiple parent solutions (line 8). This new solution S' is then improved by the feasible and infeasible route descent search (line 9). The improved solution S' is used to update the population according to the population management strategy (line 10). If a superior best solution is found, S^* is updated, and the stagnation counter is reset to 0 (lines 12-14). Otherwise, the stagnation generation counter φ_{st} is incremented (lines 15-16). The algorithm terminates and returns the best solution S^* found (line 20) when the termination condition is met (line 7), which includes reaching a maximum number of generations φ_{max}, surpassing a maximum patience threshold ρ for stagnation generations φ_{st}, or exceeding a predetermined maximum running time τ. Below, we provide a detailed description of each component of the algorithm.

A. Preprocessing

To improve computational efficiency, a preprocessing step is employed before executing the main algorithm to prune the search space. As outlined in Section I, VRPSPDTW can be defined on a directed complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertices $\mathcal{V} = \{v_0, v_1, \ldots, v_\mathcal{N}\}$ represent the depot and customer nodes, and the edges $\mathcal{E} = \{e_{ij} | v_i, v_j \in \mathcal{V}\}$ represent the connections between the nodes. The preprocessing procedure removes edges that violate the time window or vehicle capacity constraints of VRPSPDTW. For any pair of nodes v_i and v_j, the directed edge e_{ij} is eliminated if one of the following rules is true.

$$e_i + s_i + t_{ij} > l_j \tag{2}$$

$$d_i + d_j > \mathcal{Q} \tag{3}$$

$$p_i + d_j > \mathcal{Q} \tag{4}$$

$$p_i + p_j > \mathcal{Q} \tag{5}$$

Rule (2) allows the connection between v_i and v_j to be pruned if the arrival time at v_j after serving v_i is later than the allowed latest time l_j for v_j. Rules (3), (4), and (5) prohibit the connection between v_i and v_j if the capacity constraint is violated before, between, and after consecutive visits to v_i and v_j.

B. Initialization

The initialization stage generates a group of candidate solutions to form the initial population. A high-quality initial solution, characterized by its feasibility and a relatively lower number of vehicles, helps the algorithm in its subsequent search. We use the RCRS algorithm [4] to generate such initial solutions. RCRS is an insertion-based heuristic that starts with an empty solution and iteratively inserts customer nodes until all nodes are accommodated. Notably, RCRS only accepts feasible insertions, and a new route is introduced if no feasible insertion is possible with the existing routes. In addition to considering travel cost, RCRS incorporates two other metrics: Residual Capacity (RC), which represents the insertion's freedom in terms of capacity, and Radial Surcharge (RS), which is to prevent late and unfavorable insertions of remote customer nodes. Using these three metrics, the

RCRS algorithm efficiently constructs a relatively high-quality solution in a short time.

C. Feasible and infeasible route descent search

This section presents the feasible and infeasible route descent search (FIRD), which is one key component of the proposed algorithm.

1) General FIRD procedure: As shown in [24], allowing controlled exploration of infeasible solutions facilitates the transition between structurally different feasible solutions and thus improves the performance of local search algorithms. Indeed, this approach has proven successful in several difficult combinatorial optimization problems with complex constraints (see, e.g., [25, 26, 27]).

We adopt this feasible and infeasible search approach in the context of solving VRPSPDTW. This is basically achieved by introducing penalty terms for constraint violations in the evaluation function (Section III-C2). Combined with a set of local search operators (Section III-C3), the feasible and infeasible route descent search is able to tunnel through infeasible regions of the search space to obtain high-quality solutions that would otherwise be difficult to attain.

Algorithm 2 Feasible and infeasible route descent search

1: **Input**: The solution S for improvement.
2: **Output**: Improved solution S'.
3: $S' \leftarrow S$ /* Current best solution */
4: **if** S is infeasible **then**
5: $S \leftarrow FeasibleInfeasibleSearch(S, S')$ /* Try to make S feasible */
6: $S' \leftarrow S$
7: **end if**
8: **while** S is feasible **do**
9: $S \leftarrow RouteDescent(S)$ /* Remove a route of S */
10: **if** S is better than S' **then**
11: $S' \leftarrow S$
12: **else**
13: $S \leftarrow FeasibleInfeasibleSearch(S, S')$ /* Optimize S and try to find a better solution than S' */
14: **if** S is better than S' **then**
15: $S' \leftarrow S$
16: **end if**
17: **end if**
18: **end while**
19: **if** S' is feasible **then**
20: $S' \leftarrow FeasibleInfeasibleSearch(S', \emptyset)$ /* Intensive search from the best-found solution S' */
21: **end if**
22: **return** S'

As shown in Algorithm 2, the FIRD procedure starts with an input solution S, which can be either feasible or infeasible. If S is infeasible, the feasible and infeasible search is applied to ensure that the solution entering the route descent procedure is feasible (lines 4-7). The route descent procedure removes the shortest route from the solution S, and its nodes are reinserted into other routes using the infeasible regret insertion strategy presented in Section III-E (line 9), as illustrated in Figure 1. If the resulting solution of route descent is not feasible or not better than the current best solution S', the feasible and infeasible search is re-applied to try to find a better solution (line 13). The route descent procedure continues until it becomes impossible to find a feasible solution with the current number of vehicles (lines 8-18). Finally, an intensive search is performed on the best feasible solution S' (lines 19-21), and the final solution S' is returned (line 22).

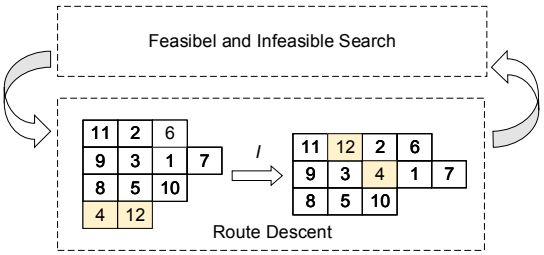


Fig. 1: Illustration of the route descent procedure. The solution has four routes. First, the shortest route 4-12 is removed. Then, the nodes 4 and 12 are inserted into other routes using the infeasible regret insertion strategy.

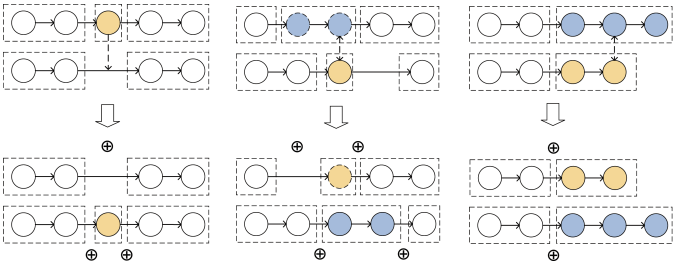
The feasible and infeasible search takes two inputs: the solution to be improved and an optional target solution. If a target solution (e.g., the best-found solution S' in lines 5 and 13) is provided, the search terminates prematurely if a better solution is found. If no target solution is provided, the search intensively improves the input solution (e.g., the best-found solution S' in line 20).

2) Evaluation function: The feasible and infeasible search dynamically accepts feasible and infeasible solutions to explore diverse areas of the search space. To achieve this, we define an evaluation function that incorporates penalty terms of time window and capacity constraint violations, as shown in Equation (6).

$$
\begin{aligned}
\text{Minimize} \quad & f_{eval}(S) = D(S) + \lambda_1 \cdot V_1(S) + \lambda_2 \cdot V_2(S) \\
\text{Subject to} \quad & D(S) = \sum_{i=1}^{M} \sum_{j=0}^{\mathcal{L}} c_{n_{i,j} n_{i,j+1}} \\
& V_1(S) = \frac{\sum c_k}{\sum t_k} \sum_{i \in R_r} \sum_{j=0}^{\mathcal{L}} \max(a_{i,j} - l_{i,j}, 0) \\
& V_2(S) = (2 \cdot \max c_k - \min c_k) \\
& \qquad \frac{\sum_{i \in R_r} \max(\max_{j=0}^{\mathcal{L}}(q_{n_{i,j}} - \mathcal{Q}), 0)}{\mathcal{Q}} \\
& S = \{R_1, \ldots, R_M\} \\
& R_i = \{n_{i,0}, n_{i,1}, \ldots, n_{i,\mathcal{L}}, n_{i,\mathcal{L}+1}\}, \quad i = 1, \ldots, M
\end{aligned}
\tag{6}
$$

$$
\lambda_i = \begin{cases} \lambda_i \cdot \kappa & \text{if } V_i = 0 \\ \frac{\lambda_i}{\kappa} & \text{if } V_i > 0 \end{cases} \quad i = 1, 2, \quad \kappa \in (0, 1) \tag{7}
$$

Since the number of vehicles (primary objective) is handled through the route descent strategy, the feasible and infeasible search only focuses on the travel distance objective $D(S)$. $V_1(S)$ and $V_2(S)$ represent the normalized violation values of time window and capacity constraints, respectively. The time-warp technique [28, 8] is adopted to address violation of time window constraints. During the search process, the coefficients λ_1 and λ_2 are dynamically adjusted at each move step based on the feasibility of the constraints. This adjustment is facilitated by the coefficient adjustment factor κ, as depicted in Equation (7). If the time window or capacity constraint is violated, the corresponding coefficient is increased to prioritize satisfying that constraint. If both constraints are violated, the coefficients



(a) Relocate (b) Swap (c) 2-opt*

Fig. 2: Illustration of operators and concatenation operation.

λ_1 and λ_2 are both increased. Conversely, if there is no constraint violation, the coefficients λ_1 and λ_2 are decreased to encourage exploration of infeasible areas.

3) Move operators and move evaluation: The feasible and infeasible search employs a set of local search move operators to explore the search space, including *Relocate*, *Swap*, and *2-opt**. *Relocate* moves a sequence of nodes to the same or another route. *Swap* exchanges two sequences of nodes in the same route or different routes. *2-opt** takes place between two routes. It first breaks one edge in two candidate routes, and then exchanges and reconnects the sequences of nodes. Here, the sequence length of nodes for *Relocate* and *Swap* operators ranges from 1 to 3. Considering their similar complexity of the employed operators, a random order is employed to execute these operators. The feasible and infeasible search stops under several conditions: when no improvement is detected after applying all operators in the sequence, or when the maximum iteration limit is reached, or when a better solution than the given target solution is found.

The search procedure adopts the best-improvement strategy to apply each local search operator. To speed up move evaluation and accommodate infeasible solutions, we introduce an infeasibility-allowed constant-time move evaluation method inspired by prior work [6, 8]. Since the used operators affect only one or two routes, the move evaluation process can be performed in constant time by computing the variation (called move gain) of the evaluation function between the original solution and a new solution. We conceptualize moves as rearrangements of subsequences of routes, using a concatenation operation denoted as $\oplus$ to symbolize the merging of two route subsequences. Illustrative examples of various operators are depicted in Figure 2, while detailed concatenation operation calculations are provided in the online supplement. With this method, the computation of the $D(\cdot)$, $V_1(\cdot)$, and $V_2(\cdot)$ values in the evaluation function can be done in constant time, which greatly speeds up the evaluation process.

D. Adaptive route-inheritance crossover

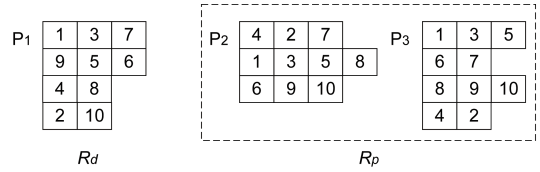
We propose an adaptive route-inheritance crossover (ARIX) for VRPSPDTW, which relies on inheriting routes from parent solutions and employs feasible and infeasible insertion operators to complete the construction of each new offspring solution. To decide the number of parent solutions and the insertion operators to apply, a learning-based adaptive mechanism is embedded to make the best possible decisions (see Section III-E).

Algorithm 3 Adaptive Route-Inheritance Crossover

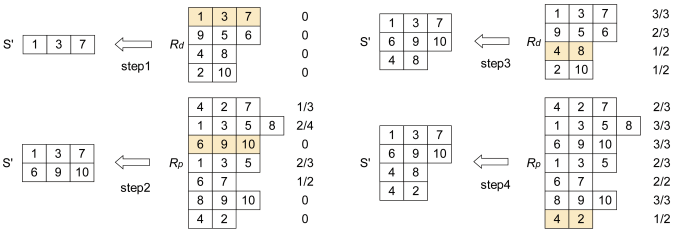
1: **Input**: Population $\mathcal{P}$.
2: **Output**: Offspring solution S'.
3: $S' \leftarrow \emptyset$ /* Initialize offspring solution */
4: Determine the number of parents N_p, the insertion operator I based on the adaptive strategy /* Section III-E */
5: Randomly select N_p solutions from the population $\mathcal{P}$ as parents P
6: $P_d \leftarrow P[0]$ /* Dominant parent */
7: $\mathcal{R}_d \leftarrow \forall R \in P[0]$ /* Dominant routes */
8: $\mathcal{R}_p \leftarrow \forall R \in P[1:]$ /* Routes from other parents form the route pool */
9: Determine the retention ratio γ_r
10: $N_r \leftarrow \lfloor \gamma_r \times |P_d| \rfloor$ /* Number of retained routes from the dominant routes */
11: $N_i \leftarrow |P_d| - N_r$ /* Number of introduced routes from the route pool */
12: **while** route number of S' is smaller than the route number of P_d **do**
13: **if** $N_r > 0$ **then**
14: Select the route with the lowest conflict rate from $\mathcal{R}_d$ and add it to S'
15: $N_r \leftarrow N_r - 1$
16: **end if**
17: **if** $N_i > 0$ **then**
18: Select the route with the lowest conflict rate from $\mathcal{R}_p$ and add it to S'
19: $N_i \leftarrow N_i - 1$
20: **end if**
21: **end while**
22: Remove redundant nodes in S'
23: $U \leftarrow \{n \in \mathcal{V} | n \notin S'\}$ /* Unrouted nodes */
24: Insert the unrouted nodes U into S' using the insertion operator I
25: **return** S'

The ARIX crossover (see Algorithm 3) starts by determining the number of parents N_p (≥ 2) and the insertion operator I based on the adaptive strategy (line 4). Then, ARIX randomly selects N_p parent solutions from the population $\mathcal{P}$ (line 5), designating the first parent as the dominant parent P_d, and gathering its routes in $\mathcal{R}d$ (lines 6-7). Routes from the other parents are pooled in $\mathcal{R}p$ (line 8). The retention ratio γ_r is then determined (line 9), which is dynamically adjusted during the search process and influences the number of routes retained from P_d and the number of introduced routes N_i (lines 10-11). ARIX then iteratively constructs the offspring solution S' by selecting routes from $\mathcal{R}d$ and $\mathcal{R}p$ based on N_r and N_i (lines 12-21). Routes with the lowest conflict rate, representing the ratio of nodes in S' to the total nodes in the route, are prioritized for selection. This process continues until the number of routes in S' reaches that of P_d (line 12). Redundant nodes in S' are subsequently removed based on selection order (line 22), and unrouted nodes are inserted using insertion operator I to obtain the final offspring (line 24). If the insertion leads to infeasibility, FIRD is employed to establish the feasibility.

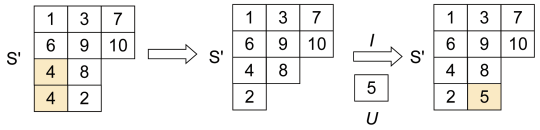
Five insertion operators are designed for the insertion of unrouted nodes. Feasible best insertion (FBI) and infeasible best insertion (IBI) prioritize the minimal cost increase, with FBI creating a new route if no feasible insertion is possible, while IBI accepts infeasible insertions without introducing new routes. Feasible regret insertion (FRI) and infeasible regret insertion (IRI) evaluate each insertion based on regret value, defined as the difference in travel cost between the best and second-best insertions. Random insertion (RI) inserts an unrouted customer randomly into a random route.



(a) Parent solutions and their routes.



(b) Inheritance from the dominant routes $\mathcal{R}_d$ and the route pool $\mathcal{R}_p$.



(c) Redundant nodes removal and unrouted nodes insertion.

Fig. 3: Illustration of the ARIX crossover.

Figure 3 illustrates the ARIX process with three parent solutions, P_1, P_2, and P_3 (Figure 3a), where P_1 is the dominant parent P_d, whose routes form the dominant routes $\mathcal{R}_d$. The routes from P_2 and P_3 form the route pool $\mathcal{R}_p$. Given that P_d has four routes and the retention ratio γ_r is set to 0.5 in this example, ARIX retains two routes from $\mathcal{R}_d$ and selects two routes from $\mathcal{R}_p$. Next (Figure 3b), the inheritance process begins. Each route is marked with its conflict rate. The offspring solution S' is constructed by alternately selecting routes with the lowest conflict rates from $\mathcal{R}_d$ and $\mathcal{R}_p$ until the number of routes in S' matches that of the dominant parent P_d. Finally (Figure 3c), the redundant node 4 is removed from S', and the unrouted node 5 is inserted using the selected insertion operator, resulting in the final offspring solution S'.

E. Learning-based adaptive strategy for the ARIX crossover

During the ARIX crossover, ARIX faces two main decision-making problems to decide the number of parents N_p used to generate offspring solutions and the insertion operator I used to insert the unrouted nodes in the last step of the crossover process. To help the ARIX crossover to make the best possible decisions for these choices, we employ an adaptive strategy based on multi-armed bandit (MAB) [29]. Specifically, we use the UCB1 algorithm from the Upper Confidence Bound (UCB) family of algorithms [30]. For a set of actions $a_i \in \mathcal{A}$, UCB1 calculates the upper confidence bound of the estimated reward for each action a_i at step t using Equation (8). $\bar{R}_i^d$ is the empirical mean reward of action a_i, N_i is the number of times action a_i has been selected so far. The action a_i with the maximum value of $UCB1(i,t)$ is selected at each step.

$$UCB1(i,t) = \bar{R}_i^d + \sqrt{\frac{2 \cdot \ln(t)}{N_i}} \tag{8}$$

In our context, we define the set of actions $\mathcal{A}$ as the set $\{2, ..., |\mathcal{P}|\}$ of all possible N_p parents, and the set $\{FBI, IBI, FRI, IRI, RI\}$ of all possible insertion operators I. The reward r of the action a is defined as the improvement of the objective function value of the offspring S' from the action a compared with the dominant parent P_d.

F. Population management

Following [31, 32], we use a fitness-distance based population management strategy, which relies on the extended fitness function ψ defined in Equation (9). This function is derived from the max-min normalization of the objective value and the distance within the population. The coefficient ξ adjusts the balance between the effects of the objective value and the distance, with higher fitness values indicating superior solution fitness. Furthermore, Equation (10) quantifies the distance of a given solution S from the population $\mathcal{P}$, while Equation (11) measures the dissimilarity between two solutions. $\mathcal{E}_{S_1}$ and $\mathcal{E}_{S_2}$ represent the sets of edges in solutions S_1 and S_2, respectively.

$$\psi(S) = \frac{\max f - f(S)}{\max f - \min f} + \xi \cdot \frac{D_{\mathcal{P}}(S) - \min D_{\mathcal{P}}}{\max D_{\mathcal{P}} - \min D_{\mathcal{P}}} \tag{9}$$

$$\text{Subject to} \quad \max f = \max_{S_i \in \mathcal{P}} f(S_i), \quad \min f = \min_{S_i \in \mathcal{P}} f(S_i)$$

$$\max D_{\mathcal{P}} = \max_{S_i \in \mathcal{P}} D_{\mathcal{P}}(S_i), \quad \min D_{\mathcal{P}} = \min_{S_i \in \mathcal{P}} D_{\mathcal{P}}(S_i)$$

$$D_{\mathcal{P}}(S) = \min_{S_i \in \mathcal{P}} D_S(S, S_i), \quad S_i \neq S \tag{10}$$

$$D_S(S_1, S_2) = 100\% \left(1 - \frac{|\mathcal{E}_{S_1} \cap \mathcal{E}_{S_2}|}{\max(|\mathcal{E}_{S_1}|, |\mathcal{E}_{S_2}|)}\right), \quad \mathcal{E}_{S_1}, \mathcal{E}_{S_2} \in \mathcal{E} \tag{11}$$

During the execution of the algorithm, the population is expanded for $\mathcal{N}_{\mathcal{P}}/2$ generations before being reduced to its original size $\mathcal{N}_{\mathcal{P}}$. This resizing involves eliminating solutions with the worst ψ values, thereby maintaining an adaptive balance in the population composition.

G. Discussion

The proposed algorithm has a number of novel features compared to existing studies on VRPSPDTW, particularly in terms of local optimization and crossover.

For local optimization, unlike the complex large neighborhood search techniques used in recent studies [5, 6, 7], we employ a lightweight feasible and infeasible search with simple local search operators. Key distinctions include the management of penalty terms, the infeasibility-allowed constant-time move evaluation, and the route descent strategy. Unlike penalty design in ALNS-PR [5], we normalize penalty terms and dynamically adjust coefficients to balance the exploration of feasible and infeasible solutions for two constraints. Compared to the evaluation method in MATE [6], which focuses only on feasible solutions, our method accommodates both feasible and infeasible solutions and facilitates the calculation of violation values. Additionally, unlike VNS-BSTS [17], which handles optimization objectives in two stages, our algorithm simultaneously optimizes the number of vehicles and the travel distance by integrating the route descent strategy into the feasible and infeasible search.

For crossover, although our ARIX crossover follows the similar idea of inheriting routes from parent solutions, ARIX differs from those in [11, 6] by using multiple parents and insertion operators that are adaptively determined by a learning mechanism to create more diversified and promising offspring solutions. In addition, ARIX strategically evaluates and selects the inherited routes based on the conflict rate instead of the random selection to improve the offspring quality.

In summary, we present a novel and effective approach to solve VRPSPDTW. Its strategies and operators can also be extended to other VRPs. The route descent strategy could be useful for VRP variants with a flexible number of vehicles. The management of penalty terms and the constant-time move evaluation can be applied to VRPs with complex constraints. The idea of our adaptive route-inheritance crossover, leveraging a learning-based strategy, could be extended to various VRPs due to its general nature.

IV. Computational results

We show computational experiments to evaluate the MA-FIRD algorithm and comparisons with the state-of-the-art algorithms based on two VRPSPDTW benchmarks.

A. Benchmark instances

The first benchmark consists of the popular 65 WC instances[1] from [11]. Among these instances, 9 are of small size with 10 to 50 customers, while the remaining 56 are of medium size with 100 customers. This benchmark is divided into three types: C instances with clustered customers, R instances with customer locations generated uniformly and randomly on a square, and RC instances with a combination of randomly placed and clustered customers. These instances are further classified to two categories: instances with narrow time windows and small vehicle capacity, and instances with large time windows and large vehicle capacity. Following [11, 15, 16, 5, 17, 6], we adopt the hierarchical objective for the WC benchmark with the number of vehicles and travel distance serving as the primary and secondary objectives, respectively. The instances of the WC benchmark have been extensively tested in the literature [11, 15, 16, 5, 17, 6, 7, 19]

The second benchmark consists of the 20 large JD instances[2] from [6], which was derived from real distribution system of JD Logistics company. This benchmark contains 4 groups of 5 instances with 200, 400, 600, 800, and 1000 customers. Unlike the WC benchmark designed for the hierarchical objective (first the number of vehicles and then the travel distance), the JD benchmark was initially designed to minimize the total cost, which is the weighted sum of the number of vehicles and the travel distance with predefined weights. This benchmark allows us to verify the usefulness of the proposed algorithm in real-world applications. Since this benchmark set is very recent, it has only been tested in [6].

[1]https://oz.nthu.edu.tw/~d933810/test.htm

[2]https://github.com/senshineL/VRPenstein

B. Experimental protocol and parameters tuning

The MA-FIRD algorithm was coded in C++[3] and run on a computer with an AMD EPYC 7282 2.8 GHz processor and 4 GB RAM, running Linux.

To balance solution quality and computational efficiency, we empirically determined several parameters. The population size $\mathcal{N}_{\mathcal{P}}$ was set to 10, the maximum number of generations φ_{max} was set to 5000, and the patience for stagnation generations ρ was set to 500. Additionally, the algorithm relies on three critical parameters: the coefficient of the fitness function ξ, the coefficient adjustment factor κ, and the initial retention ratio γ_r^0. To find a suitable setting for these parameters, we conducted an exhaustive parameter sensitivity analysis, detailed in the online supplement. The resulting parameter values are summarized in Table II.

TABLE II
Parameters tuning results.

Parameter	Value range	Final value
ξ : coefficient of the fitness function	(0, 1)	0.8
κ : coefficient adjustment factor	(0, 1)	0.5
γ_r^0 : initial retention ratio	(0, 1)	0.9

To ensure a fair comparison, the maximum running time τ for the JD benchmark is set to 7200 seconds following [6]. Each instance was solved with 30 independent runs of the algorithm.

C. Reference algorithms

For the WC instances, we compare our results with the best-known solutions (BKS) ever reported in the literature and with the top performing algorithms ALNS-PR [5] and MATE [6]. These two algorithms outperformed the other algorithms, and together produced the current best-known solutions for the WC instances. For the JD instances, we focus on a comparative study with MATE [6], which tested these 20 large-scale instances. Since the code of MATE is open source, we reran it on our computer with the same experimental protocol as for our algorithm and the results are indicated by MATE*.

The experimental environments of the reference algorithms are as follows. ALNS-PR [5] was programmed in Java and performed on a Windows 10 Professional desktop computer with an Inter Core i5-6600 3.30 GHz processor and 16 GB RAM. MATE [6] was programmed in C++ and executed on a machine with an Intel Xeon E5-2699A V4 2.40 GHz and 128 GB RAM, running Centos 7.5. To account for different experimental environments, we apply a time conversion ratio γ to standardize the computing time in following experiments. Detailed information can be found in the online supplement.

D. Computational results and comparisons

The comparative results on the two benchmark sets are summarized in Table III, which provides a global view of the performance of the proposed algorithm across different

[3]The source code and our solutions for benchmark instances are available at https://github.com/leizy1008/MA-FIRD.

benchmark groups. These groups are identified as WC-S, WC-C, WC-R, WC-RC, and JD, corresponding to the small WC, clustered WC, random WC, clustered-random WC instances, and JD instances, respectively. The row *#Instances* gives the number of instances within each group. The table contains information about the instances where the proposed algorithm shows *better-same-worse* results compared to the BKS and the reference algorithms. The '−' symbol indicates the unavailability of the results. In addition, the Wilcoxon signed-rank test [33] with a confidence level of 0.05 was conducted to ascertain the statistical significance of the results, and the corresponding *p-values* are given in the table.

TABLE III
SUMMARY OF COMPARATIVE RESULTS ON BENCHMARK INSTANCES BETWEEN MA-FIRD AND THE REFERENCES.

Group	WC-S	WC-C	WC-R	WC-RC	JD	Total	*p-value*
#Instances	9	17	23	16	20	85	
MA-FIRD vs BKS	0-9-0	0-17-0	5-18-0	3-13-0	20-0-0	28-57-0	3.79E-06
MA-FIRD vs ALNS-PR [5]	-	3-14-0	11-12-0	10-6-0	-	24-41-0	1.82E-05
MA-FIRD vs MATE [6]	0-9-0	2-15-0	10-13-0	5-11-0	20-0-0	37-48-0	8.39E-08
MA-FIRD vs MATE*	0-9-0	2-15-0	11-12-0	6-10-0	20-0-0	39-46-0	9.56E-09

Table III shows that the MA-FIRD algorithm consistently provides comparable or superior results compared to the BKS and the reference algorithms. For the WC benchmark, MA-FIRD achieved 8 new best upper bounds and reached the BKS values for all remaining instances. For the JD benchmark, MA-FIRD established new best upper bounds for all 20 instances. The associated *p-values* ($<< 0.05$) indicates that the proposed algorithm statistically performs better than the BKS and reference algorithms.

1) Comparison on the small and medium WC instances: Tables IV and V show a detailed comparison of the proposed algorithm with the BKS and the reference algorithms on an instance-by-instance basis for the small and medium WC instances, where the hierarchical objective function aims to first minimize the number of vehicles and then the travel distance. The results of ALNS-PR and MATE are taken directly from [5] and [6] respectively, while MATE* shows the results we obtained by re-running the MATE code on our computer.

The *Instance* column indicates the instance names, and *BKS* shows the best-known solutions in the literature. The M and D columns display the number of vehicles and travel distance obtained by each algorithm. $M\pm$*std* and $D\pm$*std* respectively show the average values and standard deviations of the number of vehicles and the travel distance obtained by the corresponding algorithm. The *t* column shows the running time in seconds for each algorithm, while the $\gamma\cdot$ *t* column reports the converted time using the time conversion ratio γ.

For the 9 small WC instances, despite small differences in computation time, all methods solve these instances consistently and reliably within a short time. These instances do not represent a challenge for the compared algorithms.

For the 56 medium WC instances, a weight value of 2000 is assigned to the number of vehicles (M), and 1 to the travel distance (D) following [6]. The *Gap* column shows the relative gap between the solutions and the BKS, calculated using Equation (12). A negative gap indicates that the algorithm has found a better solution than the BKS, and these improved values are highlighted in bold.

$$\text{Gap} = 100\% \cdot \frac{(2000 \cdot M + D) - (2000 \cdot M_{\text{BKS}} + D_{\text{BKS}})}{2000 \cdot M_{\text{BKS}} + D_{\text{BKS}}} \tag{12}$$

Table V shows that the MA-FIRD algorithm outperforms the reference algorithms. It reaches the BKS values for 48 instances and improves the best-known results for the 8 remaining instances (indicated in bold). Notably, within these 8 instances, MA-FIRD obtains a new best upper bound for the instance *Rdp104* with a smaller *M* value (with 9 vehicles instead of 10 for the BKS). MA-FIRD performs consistently and stably with small, and even zero, standard deviations for the number of vehicles and the travel distance for most instances. For the primary objective *M* (the number of vehicles), MA-FIRD has a better mean value of 7.27 against 7.30 for ALNS-PR, 7.36 for MATE, and 7.45 for MATE*. In terms of the gap to the BKS, MA-FIRD holds the best gap of -0.19% (a negative gap, indicating an improved result) against 0.14% for ALNS-PR, 2.00% for MATE, and 2.79% for MATE*. In terms of the running time, all algorithms perform similarly, except for MATE*, which requires a longer running time.

2) Comparison on the large JD instances: Table VI shows the comparison results for the large JD instances between MATE, MATE*, and MA-FIRD. Recall that for the JD instances, the objective is to minimize the total cost, which is the weighted sum of the number of vehicles and the travel distance. The columns f_{Best} and f_{Avg} $\pm$ *std* show the best and average total cost of each algorithm, including standard deviations. Using MATE's results as a baseline, the columns Gap_{Best} and Gap_{Avg} show the relative gap for the best and average values obtained by MATE* and MA-FIRD over 30 independent runs. So a negative gap indicates an improved result. The number of vehicles M and the travel distance D are also reported for reference.

It is evident that the MA-FIRD algorithm consistently outperforms both MATE and MATE* with significant improvements in all instances in terms of the total cost, with a negative mean gap of -2.28% for average results. Even the average objective value is better than the best reference value (144677.53 against 179464.90). If we examine the number of vehicles and the travel distances separately, we observe that MA-FIRD dominates MATE and MATE* in both criteria. These results demonstrate the effectiveness of the proposed algorithm in solving large instances in real-world scenarios.

3) Time-to-target analysis: To further compare the computational efficiency between the proposed MA-FIRD algorithm and the MATE algorithm, we perform a time-to-target (TTT) analysis [34, 35] on the medium WC instances. This analysis involves solving each instance 100 times for both algorithms, measuring the probability distribution of the required time to achieve a predefined target objective value. In this analysis, the maximum running time per run is set to 300 seconds.

Figure 4 visually depicts the results for four representative instances *Cdp101*, *Rdp203*, *RCdp101*, and *RCdp204*. The results confirm that the proposed algorithm consistently achieves the given targets within a short running time, indicating its remarkable convergence speed and computational efficiency.

TABLE IV
COMPARATIVE RESULTS ON THE 9 SMALL WC BENCHMARK INSTANCES BETWEEN MATE, MATE* AND MA-FIRD.

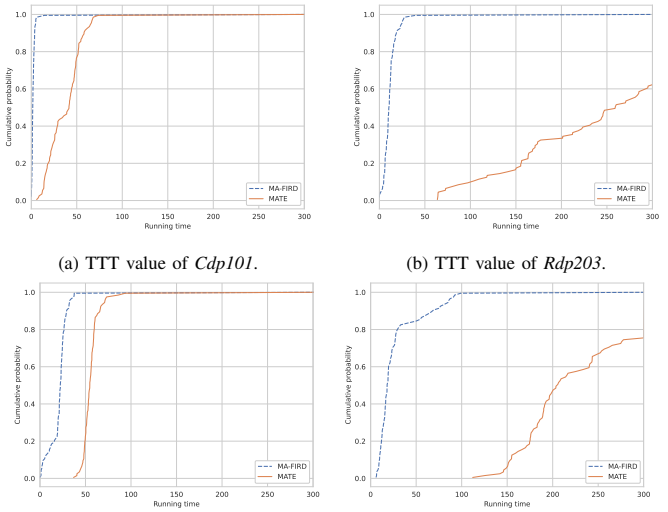
Instance	MATE (Avg)				MATE* (Avg)			MA-FIRD (Avg)		
	M ± std	D ± std	t	$\gamma \cdot t$	M ± std	D ± std	t	M ± std	D ± std	t
RCdp1001	3.00±0.00	348.98±0.00	1.00	1.24	3.00±0.00	348.98±0.00	0.06	3.00±0.00	348.98±0.00	0.16
RCdp1004	2.00±0.00	216.69±0.00	1.00	1.24	2.00±0.00	216.69±0.00	0.16	2.00±0.00	216.69±0.00	0.28
RCdp1007	2.00±0.00	310.81±0.00	1.00	1.24	2.00±0.00	310.81±0.00	0.19	2.00±0.00	310.81±0.00	0.26
RCdp2501	5.00±0.00	551.05±0.00	1.00	1.24	5.00±0.00	551.05±0.00	0.25	5.00±0.00	551.05±0.00	0.55
RCdp2504	4.00±0.00	473.46±0.00	1.00	1.24	4.00±0.00	473.46±0.00	1.28	4.00±0.00	473.46±0.00	1.42
RCdp2507	5.00±0.00	540.87±0.00	1.00	1.24	5.00±0.00	540.87±0.00	1.23	5.00±0.00	540.87±0.00	1.25
RCdp5001	9.00±0.00	994.18±0.00	1.00	1.24	9.00±0.00	994.18±0.00	1.38	9.00±0.00	994.18±0.00	2.56
RCdp5004	6.00±0.00	733.21±0.00	9.00	11.16	6.00±0.00	733.21±0.00	9.59	6.00±0.00	733.21±0.00	19.25
RCdp5007	7.00±0.00	809.72±0.00	9.00	11.16	7.00±0.00	809.72±0.00	9.00	7.00±0.00	809.72±0.00	6.51
Mean	4.78±0.00	553.22±0.00	2.78	3.44	4.78±0.00	553.22±0.00	2.57	4.78±0.00	553.22±0.00	3.58

TABLE V
COMPARATIVE RESULTS ON THE 56 MEDIUM WC BENCHMARK INSTANCES BETWEEN BKS, ALNS-PR, MATE, MATE* AND MA-FIRD.

Instance	BKS		ALNS-PR					MATE					MATE*				MA-FIRD (Best)				MA-FIRD (Avg)			
	M	D	M	D	t	$\gamma \cdot t$	Gap	M	D	t	$\gamma \cdot t$	Gap	M	D	t	Gap	M	D	t	Gap	M± std	D± std	t	Gap
Cdp101	11	976.04	11	976.04	19.30	23.93	0.00%	11	976.04	102.04	108.16	0.00%	11	976.04	10.08	0.00%	11	976.04	14.46	0.00%	11.00±0.00	977.36±0.53	18.77	0.01%
Cdp102	10	941.49	10	941.49	28.11	34.86	0.00%	10	941.49	78.18	82.87	0.00%	10	941.49	32.23	0.00%	10	941.49	51.50	0.00%	10.00±0.00	941.49±0.00	62.36	0.00%
Cdp103	10	892.98	10	892.98	48.03	59.56	0.00%	10	892.98	78.66	83.38	0.00%	10	892.98	100.54	0.00%	10	892.98	136.24	0.00%	10.00±0.00	893.75±1.11	175.70	0.00%
Cdp104	10	871.40	10	871.40	46.51	57.67	0.00%	10	871.40	79.41	84.17	0.00%	10	871.40	111.53	0.00%	10	871.40	226.19	0.00%	10.00±0.00	871.51±0.60	247.13	0.00%
Cdp105	10	1053.12	10	1053.12	15.97	19.80	0.00%	10	1074.51	67.36	71.40	0.10%	11	980.55	10.00	9.16%	10	1053.12	11.09	0.00%	10.00±0.00	1053.12±0.00	15.67	0.00%
Cdp106	10	963.45	10	967.71	17.36	21.53	0.02%	10	963.45	100.87	106.92	0.00%	10	963.45	18.65	0.00%	10	963.45	19.56	0.00%	10.00±0.00	963.45±0.00	25.72	0.00%
Cdp107	10	987.64	10	987.64	18.06	22.39	0.00%	10	988.60	88.37	93.67	0.00%	10	1072.35	18.49	0.40%	10	987.64	23.07	0.00%	10.00±0.00	988.09±1.41	33.73	0.00%
Cdp108	10	932.50	10	932.88	18.22	22.59	0.00%	10	932.49	76.52	81.11	0.00%	10	932.50	90.64	0.00%	10	932.50	36.99	0.00%	10.00±0.00	932.50±0.00	45.96	0.00%
Cdp109	10	909.27	10	910.95	38.45	47.68	0.01%	10	909.27	78.86	83.59	0.00%	10	909.27	104.62	0.00%	10	909.27	107.32	0.00%	10.00±0.00	909.27±0.00	131.31	0.00%
Cdp201	3	591.56	3	591.56	24.37	30.22	0.00%	3	591.56	90.21	95.62	0.00%	3	591.56	24.27	0.00%	3	591.56	3.98	0.00%	3.00±0.00	591.56±0.00	4.43	0.00%
Cdp202	3	591.56	3	591.56	46.09	57.15	0.00%	3	591.56	158.59	168.11	0.00%	3	591.56	36.36	0.00%	3	591.56	10.09	0.00%	3.00±0.00	591.56±0.00	10.90	0.00%
Cdp203	3	591.17	3	591.17	44.19	54.80	0.00%	3	591.17	62.22	65.95	0.00%	3	591.17	54.60	0.00%	3	591.17	26.50	0.00%	3.00±0.00	591.17±0.00	29.62	0.00%
Cdp204	3	590.60	3	590.6	51.76	64.18	0.00%	3	590.60	78.93	83.67	0.00%	3	590.60	100.62	0.00%	3	590.60	44.40	0.00%	3.00±0.00	590.62±0.10	48.30	0.00%
Cdp205	3	588.88	3	588.88	36.35	45.07	0.00%	3	588.88	134.77	142.86	0.00%	3	588.88	30.74	0.00%	3	588.88	4.42	0.00%	3.00±0.00	588.88±0.00	4.87	0.00%
Cdp206	3	588.49	3	588.49	36.39	45.12	0.00%	3	588.49	176.76	187.37	0.00%	3	588.49	37.96	0.00%	3	588.49	7.29	0.00%	3.00±0.00	588.49±0.00	7.64	0.00%
Cdp207	3	588.29	3	588.29	39.83	49.39	0.00%	3	588.29	196.92	208.74	0.00%	3	588.29	39.85	0.00%	3	588.29	11.51	0.00%	3.00±0.00	588.29±0.00	12.49	0.00%
Cdp208	3	588.32	3	588.32	34.47	42.74	0.00%	3	588.32	215.96	228.92	0.00%	3	588.32	46.42	0.00%	3	588.32	10.33	0.00%	3.00±0.00	588.32±0.00	11.66	0.00%
Rdp101	19	1650.80	19	1650.80	22.86	28.35	0.00%	19	1650.80	53.82	57.05	0.00%	19	1650.80	6.56	0.00%	19	1650.80	10.27	0.00%	19.00±0.00	1651.24±1.04	12.28	0.00%
Rdp102	17	1486.12	17	1486.12	20.89	25.90	0.00%	17	1486.12	49.49	52.46	0.00%	17	1486.12	20.62	0.00%	17	1486.12	22.18	0.00%	17.00±0.00	1488.95±5.14	35.19	0.01%
Rdp103	13	1294.64	13	1297.01	17.83	22.11	0.01%	13	1294.64	78.82	83.55	0.00%	13	1294.64	72.16	0.00%	13	1294.64	131.76	0.00%	13.77±0.43	1233.31±34.96	58.16	5.39%
Rdp104	10	984.81	10	984.81	28.40	35.22	0.00%	10	984.81	79.19	83.94	0.00%	10	984.81	116.38	0.00%	**9**	**1026.42**	73.55	**-9.33%**	**9.97±0.18**	**987.83±7.95**	75.87	**-0.30%**
Rdp105	14	1377.11	14	1377.11	17.58	21.80	0.00%	14	1377.11	105.43	111.76	0.00%	14	1377.11	12.08	0.00%	14	1377.11	10.68	0.00%	14.00±0.00	1377.11±0.00	13.46	0.00%
Rdp106	12	1252.03	12	1252.03	27.54	34.15	0.00%	12	1252.03	78.81	83.54	0.00%	12	1252.03	44.62	0.00%	12	1252.03	103.97	0.00%	12.00±0.00	1257.26±4.05	59.03	0.02%
Rdp107	10	1112.55	10	1121.86	18.79	23.30	0.04%	10	1124.90	78.99	83.73	0.06%	10	1126.94	180.03	0.07%	10	1112.55	107.73	0.00%	10.00±0.00	1121.51±6.38	95.27	0.04%
Rdp108	9	965.22	9	965.54	20.60	25.54	0.00%	9	965.22	79.71	84.49	0.00%	9	965.22	603.82	0.00%	9	965.22	112.70	0.00%	9.00±0.00	970.27±4.84	119.47	0.03%
Rdp109	11	1194.73	11	1194.73	16.08	19.94	0.00%	11	1203.97	76.69	81.29	0.04%	12	1156.84	31.43	8.46%	11	1194.73	36.80	0.00%	11.47±0.51	1183.27±27.94	41.06	3.97%
Rdp110	10	1121.46	10	1148.20	19.13	23.72	0.13%	10	1166.47	78.42	83.13	0.21%	11	1081.56	112.89	9.28%	10	1121.46	68.97	0.00%	10.23±0.43	1131.49±34.63	87.89	2.26%
Rdp111	10	1098.84	10	1098.84	22.15	27.47	0.00%	10	1098.84	79.07	83.81	0.00%	10	1098.84	129.21	0.00%	10	1098.84	101.44	0.00%	10.17±0.38	1096.63±19.15	96.34	1.57%
Rdp112	9	1010.42	9	1010.42	28.63	35.50	0.00%	10	953.63	78.32	83.02	10.22%	10	953.63	174.89	10.22%	**9**	**997.27**	101.98	**-0.07%**	9.93±0.25	961.87±11.65	38.03	9.56%
Rdp201	4	1252.37	4	1253.23	33.44	41.47	0.01%	4	1252.37	78.80	83.53	0.00%	4	1252.37	67.51	0.00%	4	1252.37	22.12	0.00%	4.00±0.00	1253.57±2.29	24.55	0.01%
Rdp202	3	1191.70	3	1191.70	46.71	57.92	0.00%	3	1223.69	77.46	82.11	0.44%	3	1244.54	165.95	0.73%	3	1191.70	39.56	0.00%	3.00±0.00	1193.36±2.53	54.76	0.02%
Rdp203	3	939.58	3	946.28	84.91	105.29	0.10%	3	939.58	73.94	78.38	0.00%	3	939.58	585.88	0.00%	**3**	**939.50**	88.73	**0.00%**	3.00±0.00	941.67±1.49	96.49	0.03%
Rdp204	2	833.09	2	833.09	111.52	138.28	0.00%	2	835.28	78.54	83.25	0.05%	2	860.16	520.65	0.56%	**2**	**825.52**	240.22	**-0.16%**	2.00±0.00	834.61±3.56	124.19	0.03%
Rdp205	3	994.43	3	994.43	80.35	99.63	0.00%	3	994.43	80.49	85.32	0.00%	3	994.43	289.25	0.00%	3	994.43	32.95	0.00%	3.00±0.00	997.59±6.63	48.02	0.05%
Rdp206	3	906.14	3	913.68	89.88	111.45	0.11%	3	906.14	78.23	82.92	0.00%	3	906.14	341.72	0.00%	3	906.14	106.05	0.00%	3.00±0.00	908.81±3.91	84.06	0.04%
Rdp207	2	890.61	2	890.61	82.51	102.31	0.00%	3	811.51	71.13	75.40	39.28%	3	811.51	579.05	39.28%	2	890.61	101.82	0.00%	2.03±0.18	896.76±18.39	134.66	1.49%
Rdp208	2	726.82	2	726.82	100.25	124.31	0.00%	2	726.82	81.23	86.10	0.00%	2	728.22	487.98	0.03%	2	726.82	118.93	0.00%	2.00±0.00	732.89±4.71	90.82	0.13%
Rdp209	3	909.16	3	909.16	86.59	107.37	0.00%	3	909.16	76.27	80.85	0.00%	3	909.16	1060.62	0.00%	3	909.16	102.72	0.00%	3.00±0.00	911.51±2.87	86.53	0.03%
Rdp210	3	939.37	3	939.37	82.84	102.72	0.00%	3	939.37	78.07	82.75	0.00%	3	939.37	449.32	0.00%	3	939.37	106.08	0.00%	3.00±0.00	945.45±5.99	86.71	0.09%
Rdp211	2	904.44	2	904.44	85.87	106.48	0.00%	3	767.82	68.64	72.76	37.99%	3	767.82	412.68	37.99%	**2**	**885.71**	55.53	**-0.38%**	2.03±0.18	905.64±29.15	79.89	1.38%
RCdp101	14	1708.21	14	1776.58	10.59	13.13	0.23%	14	1708.21	27.13	28.76	0.00%	14	1708.21	20.63	0.00%	14	1708.21	15.41	0.00%	14.83±0.38	1647.08±27.81	23.14	5.40%
RCdp102	12	1570.28	12	1583.62	19.19	23.80	0.05%	12	1570.28	78.92	83.66	0.00%	12	1589.91	71.53	0.08%	12	1570.28	114.33	0.00%	12.03±0.18	1575.69±20.05	61.87	0.28%
RCdp103	11	1282.53	11	1283.52	29.08	36.06	0.00%	11	1282.53	78.64	83.36	0.00%	11	1282.53	204.05	0.00%	11	1282.53	111.63	0.00%	11.00±0.00	1285.74±5.18	99.93	0.01%
RCdp104	10	1171.37	10	1171.65	22.57	27.99	0.00%	10	1171.37	79.88	84.67	0.00%	10	1171.37	186.61	0.00%	10	1171.37	100.26	0.00%	10.00±0.00	1171.56±0.51	146.17	0.00%
RCdp105	13	1632.29	14	1548.96	16.95	21.02	6.94%	13	1646.36	72.62	76.98	0.05%	14	1548.36	32.51	6.93%	13	1632.29	39.36	0.00%	13.77±0.43	1570.50±39.01	51.43	5.33%
RCdp106	12	1392.47	12	1392.47	20.47	25.38	0.00%	12	1392.47	78.62	83.34	0.00%	12	1392.47	41.38	0.00%	12	1392.47	37.52	0.00%	12.00±0.00	1393.42±3.32	46.37	0.00%
RCdp107	11	1252.79	11	1255.06	21.22	26.31	0.01%	11	1252.79	78.67	83.39	0.00%	11	1252.79	105.82	0.00%	11	1252.79	101.39	0.00%	11.00±0.00	1256.00±2.84	101.35	0.01%
RCdp108	10	1194.40	10	1198.36	19.52	24.20	0.02%	10	1208.58	79.18	83.93	0.07%	11	1151.71	101.75	9.24%	**10**	**1177.98**	104.70	**-0.08%**	**10.00±0.00**	**1186.63±12.00**	99.92	**-0.04%**
RCdp201	4	1406.94	4	1406.94	26.68	33.08	0.00%	4	1406.94	78.89	83.62	0.00%	4	1406.94	78.58	0.00%	4	1406.94	20.52	0.00%	4.00±0.00	1409.57±9.13	29.81	0.03%
RCdp202	3	1412.52	3	1414.55	39.94	49.53	0.03%	4	1161.29	79.10	83.85	23.59%	4	1161.29	73.45	23.59%	**3**	**1365.65**	107.51	**-0.63%**	**3.00±0.00**	**1372.33±14.13**	66.64	**-0.54%**
RCdp203	3	1050.64	3	1050.64	83.95	104.10	0.00%	3	1056.96	77.17	81.80	0.09%	3	1060.45	1152.32	0.14%	**3**	**1049.62**	60.83	**-0.01%**	3.00±0.00	1058.63±5.98	70.81	0.11%
RCdp204	3	798.46	3	798.46	94.85	117.61	0.00%	3	798.46	71.01	75.27	0.00%	3	798.46	334.63	0.00%	3	798.46	103.14	0.00%	3.00±0.00	799.33±2.76	85.43	0.01%
RCdp205	4	1297.65	4	1297.65	33.58	41.64	0.00%	4	1297.65	77.82	82.49	0.00%	4	1297.65	168.17	0.00%	4	1297.65	100.06	0.00%	4.00±0.00	1298.80±2.14	73.07	0.01%
RCdp206	3	1146.32	3	1146.32	59.36	73.61	0.00%	3	1146.32	78.54	83.25	0.00%	3	1146.32	392.20	0.00%	3	1146.32	37.67	0.00%	3.00±0.00	1152.35±5.86	46.23	0.08%
RCdp207	3	1061.14	3	1061.84	79.06	98.03	0.01%	3	1061.14	84.47	89.54	0.00%	3	1061.14	461.54	0.00%	3	1061.14	104.54	0.00%	3.00±0.00	1061.95±2.28	87.89	0.01%
RCdp208	3	828.14	3	828.14	72.78	90.25	0.00%	3	828.44	72.98	77.36	0.00%	3	829.00	506.23	0.01%	3	828.14	162.63	0.00%	3.00±0.00	830.39±3.36	112.32	0.03%
Mean	7.29	1044.45	7.30	1045.68	42.12	52.23	0.14%	7.36	1037.92	86.03	91.19	2.00%	7.45	1033.92	201.15	2.79%	7.27	1043.34	70.77	-0.19%	7.36±0.06	1041.11±7.13	66.63	0.65%

TABLE VI
COMPARATIVE RESULTS ON THE 20 LARGE-SCALE JD BENCHMARK INSTANCES BETWEEN MATE, MATE* AND MA-FIRD.

Instance	MATE		MATE*								MA-FIRD							
	f_{Best}	f_{Avg}± std	M	D	f_{Best}	Gap_{Best}	M± std	D± std	f_{Avg}± std	Gap_{Avg}	M	D	f_{Best}	Gap_{Best}	M± std	D± std	f_{Avg}± std	Gap_{Avg}
F201	65106.00	66097.00±292.00	41	52364.90	64664.90	-0.68%	41.63±0.49	52647.61±213.67	65137.61±213.67	-1.45%	40	52083.40	**64083.40**	**-1.57%**	40.40±0.50	52420.04±374.59	**64540.04±374.59**	**-2.36%**
F202	65012.00	66038.00±422.00	43	51923.00	64823.00	-0.29%	43.00±0.00	52141.76±149.35	65041.76±149.35	-1.51%	42	51778.10	**64378.10**	**-0.98%**	42.20±0.48	52309.00±491.89	**64969.00±491.89**	**-1.62%**
F203	65980.00	67090.00±332.00	41	53293.30	65593.30	-0.59%	41.37±0.49	53636.95±196.48	66046.95±196.48	-1.55%	41	53247.40	**65547.40**	**-0.66%**	40.93±0.25	53762.45±375.53	**66042.45±375.53**	**-1.56%**
F204	64747.00	65851.00±326.00	42	51982.70	64582.70	-0.25%	42.13±0.35	52460.08±180.22	65100.08±180.22	-1.14%	41	51737.30	**64037.30**	**-1.10%**	41.30±0.47	52678.48±539.36	**65068.48±539.36**	**-1.19%**
F401	122319.00	123261.00±446.00	78	98157.70	121557.70	-0.62%	79.30±0.75	98636.59±256.89	122426.59±256.89	-0.68%	75	93376.00	**115876.00**	**-5.27%**	76.17±0.95	96172.30±1566.68	**119022.30±1566.68**	**-3.44%**
F402	126887.00	128091.00±410.00	83	101617.00	126517.00	-0.29%	82.47±0.90	102619.60±418.57	127359.60±418.57	-0.57%	79	98297.00	**121997.00**	**-3.85%**	79.57±0.77	100708.27±1260.18	**124578.27±1260.18**	**-2.74%**
F403	120130.00	122306.00±682.00	78	96729.80	120129.80	0.00%	79.30±0.60	97604.31±403.95	121394.31±403.95	-0.75%	74	93340.00	**115540.00**	**-3.82%**	75.80±0.85	95049.40±1398.61	**117789.40±1398.61**	**-3.69%**
F404	124517.00	125242.00±359.00	81	98897.50	123197.50	-1.06%	81.60±0.77	99879.39±452.51	124359.39±452.51	-0.70%	78	96051.00	**119451.00**	**-4.07%**	79.07±0.78	98081.70±1199.85	**121801.70±1199.85**	**-2.75%**
F601	182504.00	184119.00±608.00	116	147617.00	182417.00	-0.05%	115.92±0.89	148360.73±441.98	183137.65±441.98	-0.53%	109	141408.00	**174108.00**	**-4.60%**	110.97±1.10	144920.43±1783.13	**178210.43±1783.13**	**-3.21%**
F602	187236.00	188891.00±645.00	119	151059.00	186759.00	-0.25%	119.41±0.98	151888.00±591.75	187712.14±591.75	-0.62%	113	144417.00	**178317.00**	**-4.76%**	113.83±1.46	148246.53±2376.39	**182396.53±2376.39**	**-3.44%**
F603	186644.00	188050.00±621.00	118	149843.00	185243.00	-0.75%	118.63±1.01	151448.04±553.83	187036.93±553.83	-0.54%	110	144979.00	**177979.00**	**-4.64%**	113.63±1.25	148734.27±1812.42	**182824.27±1812.42**	**-2.78%**
F604	186289.00	188110.00±790.00	120	150449.00	186449.00	0.09%	119.67±1.12	151388.07±478.76	187288.07±478.76	-0.44%	114	144001.00	**178201.00**	**-4.34%**	115.00±1.55	147555.10±2227.07	**182055.10±2227.07**	**-3.22%**
F801	213661.00	214634.00±561.00	154	167407.00	213607.00	-0.03%	152.17±1.23	168743.57±610.66	214393.57±610.66	-0.11%	147	164225.00	**208325.00**	**-2.50%**	150.03±2.36	167414.03±2867.45	**212424.03±2867.45**	**-1.03%**
F802	212752.00	213276.00±292.00	150	166591.00	211591.00	-0.55%	151.00±1.05	167888.93±541.21	213188.93±541.21	-0.04%	145	162714.00	**206214.00**	**-3.07%**	148.57±1.76	165733.33±2238.98	**210303.33±2238.98**	**-1.39%**
F803	214126.00	214870.00±318.00	155	167790.00	214290.00	0.08%	153.60±1.40	168701.80±545.41	214781.80±545.41	-0.04%	150	165213.00	**210213.00**	**-1.83%**	152.93±2.23	168220.27±2641.75	**214100.27±2641.75**	**-0.36%**
F804	209431.00	210845.00±429.00	150	165302.00	210302.00	0.42%	148.93±1.46	165991.13±404.11	210671.13±404.11	-0.08%	145	160471.00	**203971.00**	**-2.61%**	147.57±2.31	164444.90±2467.06	**208714.90±2467.06**	**-1.01%**
F1001	312606.00	314914.00±1096.00	203	251707.00	312607.00	0.00%	203.41±1.74	253436.22±849.77	314458.44±849.77	-0.14%	195	244613.00	**303113.00**	**-3.04%**	197.27±2.32	248709.20±2547.42	**307889.20±2547.42**	**-2.23%**
F1002	309158.00	311718.00±1153.00	203	248731.00	309631.00	0.15%	202.41±1.50	250399.26±917.50	311121.48±917.50	-0.19%	193	241933.00	**299833.00**	**-3.02%**	195.73±2.83	245324.30±2909.40	**304044.30±2909.40**	**-2.46%**
F1003	311377.00	313989.00±981.00	204	249512.00	310712.00	-0.21%	203.87±1.58	251582.26±947.36	312743.13±947.36	-0.40%	195	242551.00	**301051.00**	**-3.32%**	196.90±2.72	246727.33±2729.98	**305797.33±2729.98**	**-2.61%**
F1004	308816.00	311415.00±943.00	204	248314.00	309514.00	0.23%	203.19±1.41	249761.69±840.35	310719.38±840.35	-0.22%	193	240655.00	**298555.00**	**-3.32%**	196.67±2.52	244625.53±2643.12	**303625.53±2643.12**	**-2.50%**
Mean	179464.90	180940.35±585.30	119.15	143464.40	179209.40	-0.23%	119.15±0.99	144460.80±499.72	180205.95±499.72	-0.59%	113.95	139354.51	**141764.89**	**-3.12%**	115.73±1.47	142091.84±1822.54	**144677.53±1822.54**	**-2.28%**



(a) TTT value of *Cdp101*. (b) TTT value of *Rdp203*.

(c) TTT value of *RCdp101*. (d) TTT value of *RCdp204*.

Fig. 4: Examples of time-to-target analysis between MA-FIRD and MATE. The x-axis represents the running time to reach the target value and the y-axis indicates the cumulative probability of reaching the given target value.

V. Performance Analysis

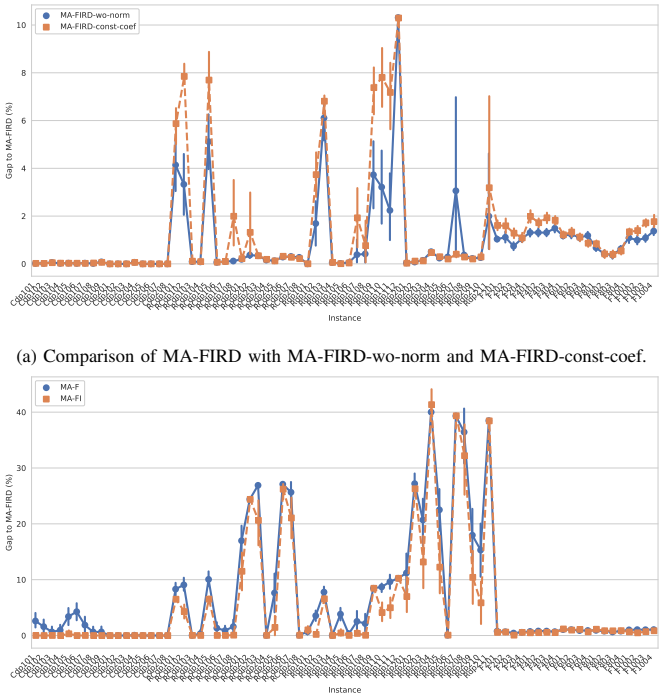
We now present experiments to study the influences of the algorithmic components based on the medium WC instances and the large JD instances.

A. Analysis on the feasible and infeasible route descent search

First, we investigate the impact of the penalty term management in the feasible and infeasible search, focusing on normalization and dynamic coefficient adjustment. We compare the MA-FIRD algorithm with two variants: MA-FIRD-wo-norm, which removes the penalty term normalization, and MA-FIRD-const-coef, which sets the penalty coefficients to a constant value ($\lambda_1 = \lambda_2 = 1.5$). Figure 5a shows the relative gap between the solutions obtained by these variants and those of MA-FIRD. The results indicate that normalization of penalty terms and dynamic coefficient adjustment significantly enhance performance as the variants perform much worse than the baseline MA-FIRD in most instances.

Next, we assess the role of the feasible and infeasible route descent search by comparing it with the variant MA-F, which eliminates the route descent strategy and accepts only feasible solutions, and the variant MA-FI, which also removes the route descent strategy but allows infeasible solutions. Figure 5b compares MA-F and MA-FI with the baseline MA-FIRD. MA-FI shows slightly better performance than MA-F, suggesting that accepting infeasible solutions can help escape feasible local optima and improve search capability. However, both variants perform significantly worse than MA-FIRD in almost all instances, highlighting the importance of the route descent strategy.

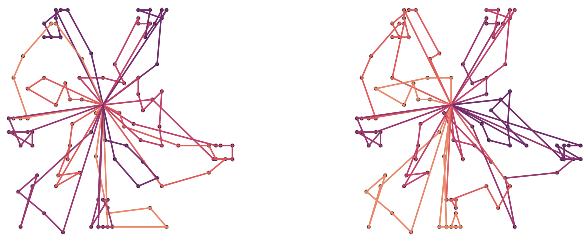
To better understand the benefits of the route descent strategy, Figures 6 and 7 illustrate high-quality solutions for adjacent numbers of vehicles in two representative instances. Different colors represent distinct routes. One notices that high-quality solutions for adjacent numbers of vehicles are



(a) Comparison of MA-FIRD with MA-FIRD-wo-norm and MA-FIRD-const-coef.

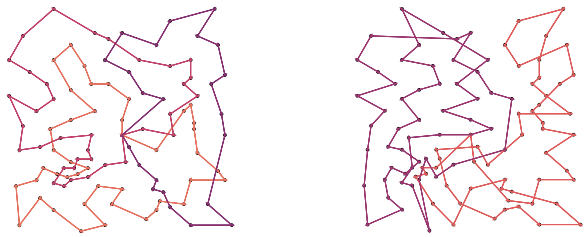
(b) Comparison of MA-FIRD with MA-F and MA-FI.

Fig. 5: Comparison of MA-FIRD with its variants. The x-axis denotes the instance names and the y-axis illustrates the relative gap between the solutions obtained by variants compared to those of MA-FIRD. A positive gap indicates that the variant performs worse than MA-FIRD, and vice versa.



(a) $M = 15$ and $f(S) = 1634.85$. (b) $M = 14$ and $f(S) = 1708.21$.

Fig. 6: Two solutions on the instance *RCdp101* with the distance of 32.17.



(a) $M = 3$ and $f(S) = 767.82$. (b) $M = 2$ and $f(S) = 885.71$.

Fig. 7: Two solutions on the instance *Rdp211* with the distance of 72.82.

not necessarily close and may show significant differences. For example, the distance between two solutions for *RCdp101* is 32.17, and for *Rdp211*, it is 72.82. This indicates significant dissimilarity between solutions with adjacent numbers of vehicles. This observation highlights that the distribution of solutions in the search space is not uniform, and high-quality solutions for different numbers of vehicles may not be proximate. The route descent strategy enables rapid traversal of the search space, helping to locate high-quality solutions for varying numbers of vehicles.

B. Analysis on the adaptive route-inheritance crossover

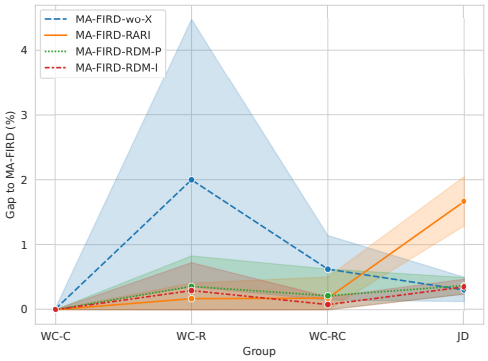
To study the impact of the ARIX crossover, we create the following variants of the MA-FIRD algorithm:

- MA-FIRD-wo-X: ARIX is removed, and only one solution is maintained in the population. At each generation, the correlated destroy and repair operator from [6] is used to perturb the solution, which is then improved by the FIRD search to generate a new solution.
- MA-FIRD-RARI: ARIX is replaced by the RARI crossover from [6].
- MA-FIRD-RDM-P: The number of parents is randomly chosen from $\{2, ..., |\mathcal{P}|\}$.
- MA-FIRD-RDM-I: The insertion operator is randomly selected from the set $\{FBI, IBI, FRI, IRI, RI\}$ at each generation.

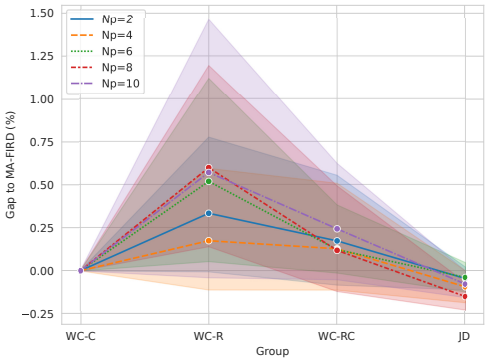
Figure 8a provides a visual comparison of the variants against MA-FIRD. The instances are grouped into WC-C, WC-R, WC-RC, and JD. Note that the WC-C instances are less sensitive to configuration variations, while instances in the other groups are more sensitive. In general, the baseline MA-FIRD algorithm outperforms MA-FIRD-wo-X and MA-FIRD-RARI in most cases, which confirms the importance of ARIX in the algorithm. Furthermore, the results of MA-FIRD-RDM-P and MA-FIRD-RDM-I lag behind MA-FIRD, suggesting that the learning-based adaptive mechanism helps ARIX to select the appropriate number of parents and insertion operators for different instances.

To gain deeper insights into the impact of different ARIX configurations, we conducted comprehensive experiments analyzing the influence of the number of parents and the insertion operations. Figures 8b and 8c present the visualized results. Figure 8b shows the results for different numbers of parents ($N_p = 2, 4, 6, 8, 10$), while Figure 8c presents the results for different insertion operators (FBI, IBI, FRI, IRI, and RI).

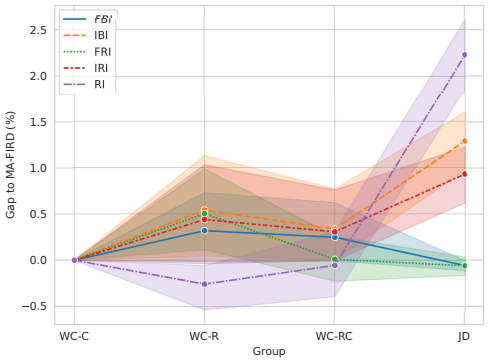
Overall, instances in the WC-C group still demonstrate less sensitivity to variations in setups. However, performance across the other three groups varies with different configurations. Notably, no single configuration consistently performs well across all instances. For example, $N_p = 6, 8, 10$ perform well on JD but poorly on WC-R, while certain insertion operators (IBI, IRI, and RI) perform well on WC-R but poorly on JD. This suggests the necessity of an adaptive mechanism to dynamically determine configurations and values. Additionally, these configurations generally do not perform better than the baseline MA-FIRD, indicating the efficacy of the learning-based adaptive mechanism.



(a) Comparison of the MA-FIRD and its variants.



(b) Results with number of parents $N_p = 2, 4, 6, 8, 10$.



(c) Results with insertion operations FBI, IBI, FRI, IRI, and RI.

Fig. 8: Comparison of the MA-FIRD algorithm and its variants. The x-axis denotes the instance groups and the y-axis illustrates the relative gap between the solutions obtained by variants compared to those of MA-FIRD. The mean relative gap for each group is indicated by a solid line, while the 0.95 confidence interval is represented by the corresponding shadow.

VI. Conclusion

We introduce an effective memetic algorithm to solve the challenging VRPSPDTW. Based on the memetic algorithm framework, our algorithm integrates several innovative strategies to effectively enhance the exploration of the solution space. These include a lightweight feasible and infeasible search mechanism featuring the route descent strategy, penalty term management, and infeasibility-allowed constant-time move evaluation. We also introduce a learning-based adaptive route-inheritance crossover to enhance generalization and robustness, and a fitness-distance population management strategy based on max-min normalization to maintain popula-

tion diversity.

Extensive experiments have been conducted to evaluate the algorithm's performance. The results show that the proposed algorithm consistently outperforms the reference algorithms with high computational efficiency. Remarkably, The algorithm attains the best solutions for all benchmark instances, reaching 57 best-known solutions (BKS) and achieving 8 new best upper bounds for the 65 popular benchmark instances. Furthermore, the algorithm achieves new best upper bounds for all 20 large real-world instances, demonstrating its effectiveness and practicality in real-world scenarios.

The flexibility and versatility of innovative strategies in the proposed algorithm lay the basis for future extensions to tackle other multi-attribute Vehicle Routing Problems. Future work aims to explore the adaptability and performance of the algorithm in a broader range of complex routing scenarios, thus contributing to the advancement of the field.

REFERENCES

[1] E. Angelelli and R. Mansini, "The vehicle routing problem with time windows and simultaneous pick-up and delivery," in *Quantitative Approaches to Distribution Logistics and Supply Chain Management*. Springer, 2002, pp. 249–267.

[2] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, "The vehicle routing problem: State of the art classification and review," *Computers & Industrial Engineering*, vol. 99, pp. 300–313, 2016.

[3] H. Min, "The multiple vehicle routing problem with simultaneous delivery and pick-up points," *Transportation Research Part A: General*, vol. 23, no. 5, pp. 377–386, 1989.

[4] J. Dethloff, "Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up: Fahrzeugeinsatzplanung und redistribution: Tourenplanung mit simultaner auslieferung und rückholung," *OR-Spektrum*, vol. 23, pp. 79–96, 2001.

[5] J. Hof and M. Schneider, "An adaptive large neighborhood search with path relinking for a class of vehicle-routing problems with simultaneous pickup and delivery," *Networks*, vol. 74, no. 3, pp. 207–250, 2019.

[6] S. Liu, K. Tang, and X. Yao, "Memetic search for vehicle routing with simultaneous pickup-delivery and time windows," *Swarm and Evolutionary Computation*, vol. 66, p. 100927, 2021.

[7] H. Wu and Y. Gao, "An ant colony optimization based on local search for the vehicle routing problem with simultaneous pickup–delivery and time window," *Applied Soft Computing*, vol. 139, p. 110203, 2023.

[8] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows," *Computers & Operations Research*, vol. 40, no. 1, pp. 475–489, 2013.

[9] P. He and J.-K. Hao, "General edge assembly crossover-driven memetic search for split delivery vehicle routing," *Transportation Science*, vol. 57, no. 2, pp. 482–511, 2023.

[10] D. Cattaruzza, N. Absi, D. Feillet, and T. Vidal, "A memetic algorithm for the multi trip vehicle routing problem," *European Journal of Operational Research*, vol. 236, no. 3, pp. 833–848, 2014.

[11] H.-F. Wang and Y.-Y. Chen, "A genetic algorithm for the simultaneous delivery and pickup problems with time window," *Computers & Industrial Engineering*, vol. 62, no. 1, pp. 84–95, 2012.

[12] C.-H. Liang, H. Zhou, and J. Zhao, "Vehicle routing problem with time windows and simultaneous pickups and deliveries," in *2009 16th International Conference on Industrial Engineering and Engineering Management*. IEEE, 2009, pp. 685–689.

[13] M. Lai and E. Cao, "An improved differential evolution algorithm for vehicle routing problem with simultaneous pickups and deliveries and time windows," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 2, pp. 188–195, 2010.

[14] S. Kassem and M. Chen, "Solving reverse logistics vehicle routing problems with time windows," *The International Journal of Advanced Manufacturing Technology*, vol. 68, pp. 57–68, 2013.

[15] C. Wang, F. Zhao, D. Mu, and J. W. Sutherland, "Simulated annealing for a vehicle routing problem with simultaneous pickup-delivery and time windows," in *Advances in Production Management Systems. Sustainable Production and Service Supply Chains: IFIP WG 5.7 International Conference, APMS 2013, State College, PA, USA, September 9-12, 2013, Proceedings, Part II*. Springer, 2013, pp. 170–177.

[16] C. Wang, D. Mu, F. Zhao, and J. W. Sutherland, "A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows," *Computers & Industrial Engineering*, vol. 83, pp. 111–122, 2015.

[17] Y. Shi, Y. Zhou, T. Boudouh, and O. Grunder, "A lexicographic-based two-stage algorithm for vehicle routing problem with simultaneous pickup–delivery and time window," *Engineering Applications of Artificial Intelligence*, vol. 95, p. 103901, 2020.

[18] K. Tang, S. Liu, P. Yang, and X. Yao, "Few-shots parallel algorithm portfolio construction via co-evolution," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 3, pp. 595–607, 2021.

[19] R. Praxedes, T. Bulhões, A. Subramanian, and E. Uchoa, "A unified exact approach for a broad class of vehicle routing problems with simultaneous pickup and delivery," *Computers & Operations Research*, vol. 162, p. 106467, 2024.

[20] Ç. Koç, G. Laporte, and İ. Tükenmez, "A review of vehicle routing with simultaneous pickup and delivery," *Computers & Operations Research*, vol. 122, p. 104987, 2020.

[21] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987.

[22] P. Moscato and C. Cotta, "A modern introduction to memetic algorithms," *Handbook of Metaheuristics*, pp. 141–183, 2010.

[23] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review," *Swarm and Evolutionary Computation*, vol. 2, pp. 1–14, 2012.

[24] F. Glover and J.-K. Hao, "The case for strategic oscillation," *Annals of Operations Research*, vol. 183(1), pp. 163–173, 2011.

[25] M. Li, J.-K. Hao, and Q. Wu, "Learning-driven feasible and infeasible tabu search for airport gate assignment," *European Journal of Operational Research*, vol. 302, no. 1, pp. 172–186, 2022.

[26] W. Sun, J.-K. Hao, X. Lai, and Q. Wu, "Adaptive feasible and infeasible tabu search for weighted vertex coloring," *Information Sciences*, vol. 466, pp. 203–219, 2018.

[27] Y. Zou, J.-K. Hao, and Q. Wu, "A two-individual evolutionary algorithm for cumulative capacitated vehicle routing with single and multiple depots," *IEEE Transactions on Evolutionary Computation*, 2024.

[28] Y. Nagata, O. Bräysy, and W. Dullaert, "A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows," *Computers & Operations Research*, vol. 37, no. 4, pp. 724–737, 2010.

[29] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *European Conference on Machine Learning*. Springer, 2005, pp. 437–448.

[30] V. Kuleshov and D. Precup, "Algorithms for multi-armed bandit problems," *arXiv preprint arXiv:1402.6028*, 2014.

[31] K. Sörensen and M. Sevaux, "MA|PM: memetic algorithms with population management," *Computers & Operations Research*, vol. 33, no. 5, pp. 1214–1225, 2006.

[32] D. C. Porumbel, J.-K. Hao, and P. Kuntz, "An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring," *Computers & Operations Research*, vol. 37, no. 10, pp. 1822–1832, 2010.

[33] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in statistics: Methodology and distribution*. Springer, 1992, pp. 196–202.

[34] R. M. Aiex, M. G. Resende, and C. C. Ribeiro, "Ttt plots: a perl program to create time-to-target plots," *Optimization Letters*, vol. 1, pp. 355–366, 2007.

[35] C. C. Ribeiro, I. Rosseti, and R. Vallejos, "Exploiting run time distributions to compare sequential and parallel stochastic local search algorithms," *Journal of Global Optimization*, vol. 54, pp. 405–429, 2012.

Zhenyu Lei received the B.S. and M.S. degrees in Computer Science from the Ocean University of China (Qingdao, China), in 2019 and 2022 respectively. He also received the Engineering degree from the Polytech Nantes (Nantes, France), in 2021. He is currently pursuing the Ph.D. degree in Computer Science at the LERIA laboratory, University of Angers, France. His research interests include evolutionary computation, heuristics, routing problems and other combinatorial optimization problems.

Jin-Kao Hao received the B.S. degree in Computer Science from the National University of Defense Technology, China, in 1982; a M.S. degree from the National Institute of Applied Sciences, Lyon, France, in 1987; the Ph.D. degree in constraint programming from the University of Franche-Comté, France, in 1991. Since 1999, he holds a full Professor position with the Computer Science Department at the University of Angers, France. His research focuses on the design of effective algorithms and intelligent computational methods for solving large-scale combinatorial search problems. He has co-authored more than 320 peer-reviewed publications and co-edited 9 books in Springer's LNCS series. He has served on more than 200 Program Committees of International Conferences and is on the Editorial Board of 7 International Journals. Dr. Hao became a Distinguished Professor in 2010 and Senior Fellow of the Institut Universitaire de France in 2015.

Supplementary materials of paper "A Memetic Algorithm for Vehicle Routing with Simultaneous Pickup and Delivery and Time Windows"

Zhenyu Lei and Jin-Kao Hao*

I. Calculations of concatenation operation

We provide detailed calculations of the concatenation operation, denoted as $\oplus$, for constant-time move evaluation of the local search operators presented in Section III-C3.

Since the move of the local search operators can be represented as a rearrangement of subsequences of routes, the evaluation of the move can be computed efficiently by maintaining a set of metrics for each route, including travel distance (D), duration time (T_d), earliest and latest arrival times (T_e and T_l), wait time (T_w), violated time window value (T_v), load values before and after the sequence (L_{in} and L_{out}), and the maximum load (L_{max}).

For a subsequence containing a single node, denoted as $\sigma^0 = \{v_i\}$, these metrics can be defined:

$$D(\sigma^0) = 0 \tag{1}$$
$$T_d(\sigma^0) = s_i \tag{2}$$
$$T_e(\sigma^0) = e_i \tag{3}$$
$$T_l(\sigma^0) = l_i \tag{4}$$
$$T_w(\sigma^0) = 0 \tag{5}$$
$$T_v(\sigma^0) = 0 \tag{6}$$
$$L_{in}(\sigma^0) = d_i \tag{7}$$
$$L_{out}(\sigma^0) = p_i \tag{8}$$
$$L_{max}(\sigma^0) = \max(d_i, p_i) \tag{9}$$

For two subsequences of routes, $\sigma_1 = \{v_i, \ldots, v_j\}$ and $\sigma_2 = \{v_k, \ldots, v_l\}$, the corresponding metrics of the combined sequence of routes $\sigma_1 \oplus \sigma_2$ can be calculated by:

$$D(\sigma_1 \oplus \sigma_2) = D(\sigma_1) + c_{j,k} + D(\sigma_2) \tag{10}$$
$$\Delta_t = T_d(\sigma_1) + T_w(\sigma^1) + t_{j,k} - T_v(\sigma_1) \tag{11}$$
$$\Delta_w = \max(T_e(\sigma_2) - \Delta_t - T_l(\sigma_1), 0) \tag{12}$$
$$\Delta_v = \max(T_e(\sigma_1) + \Delta_t - T_l(\sigma_2), 0) \tag{13}$$
$$T_d(\sigma_1 \oplus \sigma_2) = T_d(\sigma_1) + t_{j,k} + T_d(\sigma_2) \tag{14}$$
$$T_e(\sigma_1 \oplus \sigma_2) = \max(T_e(\sigma_1), T_e(\sigma_2) - \Delta_t) - \Delta_w \tag{15}$$
$$T_l(\sigma_1 \oplus \sigma_2) = \min(T_l(\sigma_1), T_l(\sigma_2) - \Delta_t) + \Delta_v \tag{16}$$
$$T_w(\sigma_1 \oplus \sigma_2) = T_w(\sigma_1) + \Delta_w + T_w(\sigma_2) \tag{17}$$
$$T_v(\sigma_1 \oplus \sigma_2) = T_v(\sigma_1) + \Delta_v + T_v(\sigma_2) \tag{18}$$
$$L_{in}(\sigma_1 \oplus \sigma_2) = L_{in}(\sigma_1) + L_{in}(\sigma_2) \tag{19}$$
$$L_{out}(\sigma_1 \oplus \sigma_2) = L_{out}(\sigma_1) + L_{out}(\sigma_2) \tag{20}$$
$$L_{max}(\sigma_1 \oplus \sigma_2) = \max(L_{max}(\sigma_1) + L_{in}(\sigma_2), L_{out}(\sigma_1) + L_{max}(\sigma_2)) \tag{21}$$

The difference of $D(\cdot)$, $V_1(\cdot)$, and $V_2(\cdot)$ in evaluation function values between the original route R and modified route R' can be computed using the values of D, T_v, and L_{max} by Equation (22) in constant time. It's important to note that multiple matrices must be maintained for each route to store the values of these metrics. These matrices can be updated promptly whenever a move alters the corresponding route.

$$\begin{aligned}\Delta f_{eval}(R', R) = {} & D(R') - D(R) \\ & + \lambda_1 \frac{\sum c_k}{\sum t_k}(T_v(R') - T_v(R)) \\ & + \lambda_2 \frac{2 \cdot \max c_k - \min c_k}{\mathcal{Q}} \\ & (\max(L_{max}(R') - \mathcal{Q}, 0) - \max(L_{max}(R) - \mathcal{Q}, 0))\end{aligned} \tag{22}$$

II. Sensitivity analysis of the parameters

This section presents a sensitivity analysis of the parameters of the algorithm to evaluate their impact on performance and calibrate them for best results. The analysis focuses on three key parameters: the coefficient of the fitness function ξ, the coefficient adjustment factor κ, and the initial retention ratio γ_r^0. The parameters are varied within the ranges: $\xi \in \{0.1, 0.2, \ldots, 0.9\}$, $\kappa \in \{0.1, 0.2, \ldots, 0.9\}$, and $\gamma_r^0 \in \{0.1, 0.2, \ldots, 0.9\}$.

Using the sensitivity analysis tool SALib [1], we randomly sampled 8000 parameter combinations from the parameter space and evaluated them on a training set comprising 10 representative and challenging benchmark instances. The average gap value against the best-known solutions (BKS) on

The first author is supported by a scholarship from the China Scholar Council (Grant No. 202206330014). (*Corresponding author*: Jin-Kao Hao)

Z. Lei and J.K. Hao are with the Department of Computer Science, LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers Cedex 01, France (e-mails: zhenyu.lei@etud.univ-angers.fr, jin-kao.hao@univ-angers.fr).

the training set served as the performance metric, and Sobol's method [2] was used to evaluate parameter sensitivity. Detailed results are presented in Table A.I and Figure 1.

TABLE A.I
SENSITIVITY ANALYSIS RESULTS.

Parameter	Sensitivity Index	Confidence Interval
Total-order indices (ST)		
ξ	0.024685	0.002019
κ	0.974937	0.057852
γ_r^0	0.000060	0.000007
First-order indices (S1)		
ξ	0.025848	0.014347
κ	0.976788	0.071350
γ_r^0	0.000146	0.000637
Second-order indices (S2)		
(ξ, κ)	-0.002324	0.028654
(ξ, γ_r^0)	-0.001171	0.020575
(κ, γ_r^0)	-0.002530	0.076821

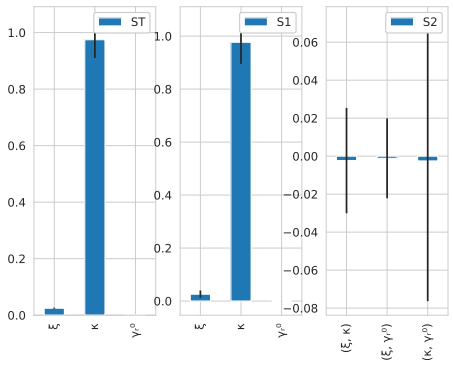


Fig. 1: Visualization of the sensitivity analysis results.

The analysis reveals that the coefficient adjustment factor κ significantly influences the algorithm's performance, with a high total-order (ST) value of 0.974937 and a first-order (S1) index of 0.976788, accompanied by low confidence intervals. Conversely, the coefficient of the fitness function ξ exhibits relatively low sensitivity, suggesting a minor impact on performance. Similarly, the initial retention ratio γ_r^0 has the lowest sensitivity index, implying minimal impact. The consistency between the total-order (ST) and first-order (S1) indices, along with negligible second-order indices (S2), indicates low interaction effects among the parameters.

Given the low interaction effects, we conducted a one-at-a-time sensitivity analysis to determine suitable values for each parameter. Each parameter was tested independently while keeping the others fixed at default values: $\xi = 0.5$, $\kappa = 0.5$, and $\gamma_r^0 = 0.5$. Each parameter setting was evaluated on the training set over 30 independent runs. Figure 2 illustrates the average gap values with different parameter settings. The best parameter settings were found to be $\xi = 0.8$, $\kappa = 0.5$, and $\gamma_r^0 = 0.9$. These values are used in the experiments.

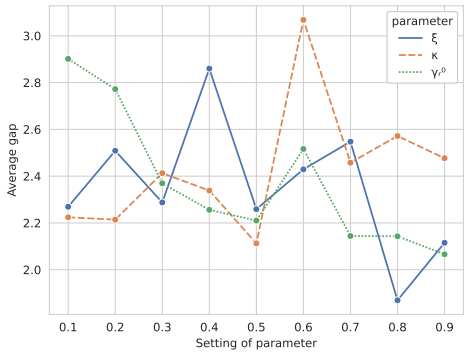


Fig. 2: Visualization of one-at-a-time sensitivity analysis.

III. CPU INFORMATION AND TIME CONVERSION RATIO

This section provides detailed CPU information and the calculated time conversion ratio γ for the proposed algorithm and the reference algorithms.

TABLE A.II
CPU INFORMATION AND TIME CONVERSION RATIO γ.

Algorithm	Processor	Base frequency	CPU mark	γ
MA-FIRD	AMD EPYC 7282	2.80 GHz	1829	1.00
ALNS-PR [3]	Intel Core i5-6600	3.30 GHz	2260	1.24
MATE [4]	Intel Xeon E5-2699A	2.40 GHz	1945	1.06

To account for different experimental environments in the proposed algorithm and reference algorithms, we apply a time conversion ratio γ to standardize the computation time. This ratio is defined as the ratio of CPU scores obtained from PassMark[1], a recognized CPU benchmarking platform. Table A.II presents the detailed information about the CPU used in our experiments and those used by the reference algorithms, and the calculated time conversion ratio γ.

REFERENCES

[1] J. Herman and W. Usher, "Salib: An open-source python library for sensitivity analysis," *Journal of Open Source Software*, vol. 2, no. 9, p. 97, 2017.

[2] I. M. Sobol, "Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates," *Mathematics and Computers in Simulation*, vol. 55, no. 1-3, pp. 271–280, 2001.

[3] J. Hof and M. Schneider, "An adaptive large neighborhood search with path relinking for a class of vehicle-routing problems with simultaneous pickup and delivery," *Networks*, vol. 74, no. 3, pp. 207–250, 2019.

[4] S. Liu, K. Tang, and X. Yao, "Memetic search for vehicle routing with simultaneous pickup-delivery and time windows," *Swarm and Evolutionary Computation*, vol. 66, p. 100927, 2021.

[1] https://www.passmark.com/