MJSAT

Malaysian Journal of Science and Advanced Technology

journal homepage: https://mjsat.com.my/



An Ablation Study of Constraint-Handling Mechanisms in Differential Evolution for Multi-Depot Vehicle Routing with Time Windows

Farid Morsidi*[1]

[1]*Faculty of Computing and Meta-Technology, Computing Department, Universiti Pendidikan Sultan Idris, Tanjong Malim, Perak, Malaysia, 35900.*

KEYWORDS

Differential Evolution
Metaheuristics
Ablation Study
Large Neighborhood Search,
Multi-Depot Vehicle Routing Problem

ARTICLE HISTORY

Received 16 July 2026
Received in revised form 5 October 2026
Accepted 19 October 2026
Available online 2 November 2026

ABSTRACT

Classical Differential Evolution (DE) solves the Multi-Depot Vehicle Routing Problem with Time Windows (MDVRPTW) efficiently but struggles to maintain time-window feasibility. This study investigates whether DE can be systematically hybridized with three constraint-handling mechanisms: adaptive penalty control, Large Neighborhood Search (LNS), and Tabu memory, implemented as independently toggleable components in a Java/JavaFX solver, enabling a controlled ablation across four MDVRPTW instances from Vidal et al. (2013): pr11a, pr11b, pr12a, and pr12b, with 30 independent repetitions per configuration per instance. LNS was the dominant contributor to cost reduction, cutting mean route distance by 35–45% relative to baseline DE (Friedman $\chi^2 = 14.80$, $df = 4$, $p = 0.0051$). Tabu memory's individual contribution was not statistically significant, and contrary to the initial hypothesis, adaptive penalty substantially increased both route distance and time-window violations when combined with LNS and Tabu, raising mean fitness by 74–91% across instances while simultaneously reducing on-time service rate. No configuration achieved full feasibility on any instance. These results clarify which mechanisms meaningfully improve DE's behavior on MDVRPTW.

© 2026 The Authors. Published by Algoritma Sdn. Bhd.
This is an open access article under the CC BY-NC 4.0 license (https://creativecommons.org/licenses/by-nc/4.0/).

1. INTRODUCTION

The Multi-Depot Vehicle Routing Problem with Time Windows (MDVRPTW) is an extension of the classic vehicle routing problem [1], [2], [3]. In this case, vehicles operate from multiple locations to deliver goods to their specific clients. Each of these clients has their own service time-window. This problem is highly relevant in areas such as city delivery services and transit construction where efficiency and the need to comply with certain time limitations must be connected. The MDVRPTW is NP-hard which means that solving it in a reasonable time frame is not possible for most cases [4], [5].

Originally pioneered for continuous optimization, Differential Evolution (DE) has seen increasing applications in discrete and combinatorial problems with the help of permutation representations, considered simple, computationally inexpensive, and with excellent global exploration [6]. However, the classical DE algorithm does not have any inherent mechanism for dealing with the constraints of the optimization problem; thus, the optimization without customized components or dynamic constraint strategies will lead to finding infeasible low-cost solutions, which is one of the drawbacks when solving constrained problems like MDVRPTW. Several studies have been conducted to address the issue. For example, adaptive parameter adjustment learned from optimization experience to reduce the DE's dependency on manual tuning [7], the developments of DE modifications that focus on scheduling, like RI-DE utilizing re-initialization mutation and local search to improve routing solutions [6], and adaptive differential evolution (ADE) used to study exploration-exploitation trade-off in multi-compartment routing [8]. Apart from that, dynamic and adaptive penalty functions have also been suggested to lower the search bias that occurs due to applying the fixed mechanisms for constraint handling [9].

*Corresponding author:
E-mail address: Farid Morsidi < farid_m2mfan@yahoo.com.my >.
https://doi.org/10.56532/mjsat.v6i4.801
2785-8901/ © 2026 The Authors. Published by Algoritma Sdn. Bhd.
This is an open access article under the CC BY-NC 4.0 license (https://creativecommons.org/licenses/by-nc/4.0/).

Combining approaches from global metaheuristics and local search has been found to be useful in solving routing problems, because improving local results allows one to deal with violation of constraints while keeping the structure of the solution intact; this can be illustrated by the use of 2-opt and route relocation [10], variable neighborhood descent [11], as well as hybrid destroy-repair strategies that incorporate Large Neighborhood Search with construction heuristics for multi-compartment and fuel-delivery routing variations [12], [13]. The issue of keeping variation in the population to prevent premature convergence in this area has to do with different methods such as random solution increase, crowd distance tracking, as well as changing mutation rates. However, despite being numerous, all the approaches have a common shortcoming, which is the absence of DE hybrid methods to MDVRPTW that separate, instead of combining, effects of using each method of penalty control, local search, and memory-based diversification on the overall efficiency and applicability of the approach. The main aim of the research was to evaluate whether DE can minimize routing costs and also comply with time windows at the same time, without being fully tailored to address the constraints involved, and what particular mechanisms actually account for observed improvements, instead of simply providing one single combined hybrid approach and declaring it as a better option. It was expected that standard DE would reduce routing costs effectively but not be able to keep time windows in mind.

A Java-based modular and extensible MDVRPTW solver has been developed that includes a DE engine, a JavaFX visualizer, and a metrics-logging framework capable of real-time visualization of algorithm performance and automatic collection of statistics such as fitness, distance, penalty, and feasibility throughout generations. The solver was tested by employing MDVRPTW benchmark instances from Vidal et al. (2013) to build the empirical evidence upon an established basis [14]. The contributions of this paper are threefold: (i) the DE-based MDVRPTW solver that provides full extensibility including the tools for route visualization and metrics analysis; (ii) the ablation study demonstrating isolated impact of three constraint-handling methods: adaptive penalty control, Large Neighborhood Search, and Tabu memory on DE's feasibility and cost efficiency in MDVRPTW; and (iii) the comprehensive examination of unresolved issues in DE-based solvers for MDVRPTW. Section 2 reviews related work; Section 3 describes the methodology; Section 4 presents the ablation results and discussion; Section 5 concludes with limitations and future work.

2. RELATED WORK

The Multi-Depot Vehicle Routing Problem with Time Windows (MDVRPTW) extends the standard Vehicle Routing Problem (VRP) by requiring that customer demand be met within specified time windows using a fleet based at more than one depot. First formalized in the benchmark studies of Cordeau, it captures the complexity of modern transportation logistics and has become a standard problem in the combinatorial optimization literature. The MDVRPTW is NP-hard, since routing cost must be optimized simultaneously with time-window and vehicle-capacity constraints.

The research explored different meta-heuristic techniques to solve the problem of premature solution exploration from different data points which included Genetic Algorithms (GA) [15], Simulated Annealing (SA) [16], Tabu Search [17], [18], [19], and Ant Colony Optimization (ACO) [20], [21], [22]. The growing importance of MDVRPTW reflects the wider adoption of just-in-time and rapid-delivery logistics models, which place tighter demands on routing efficiency than earlier scheduling paradigms. These approaches typically apply custom operators and constraint-handling strategies to maintain feasibility under time-window and multi-depot constraints. Lagaros et al. [23] review constraint-handling techniques (CHTs) for metaheuristic optimization algorithms (MOAs) more broadly, introducing eight variants that use feasibility- and ε-constrained selection based on the number and severity of constraint violations. Bae and Moon [24] examine a related MDVRPTW variant for scheduling delivery and installation vehicles from multiple depots, while Kashyap et al. [25] propose a Roulette-Wheel Particle Swarm Optimization (RWPSO) method shown to be competitive with contemporary algorithms on the same class of problem.

Differential Evolution (DE), introduced by Storn and Price, is a population-based optimizer valued for its simple design, strong global search, and fast convergence [6]. Although DE was originally developed for continuous optimization, it has since been applied to discrete domains including VRP and MDVRP [6]. Classical DE has no built-in mechanism for constraint handling, however, which is a significant limitation for feasibility-critical problems like MDVRPTW: without customized operators or dynamic constraint strategies, DE tends to converge to low-cost but infeasible solutions. Sun et al. [7] address DE's sensitivity to manual tuning through adaptive parameter control learned from prior optimization experience. Moonsri et al. [6] apply an improved DE variant (RI-DE), using re-initialization mutation and local search, to a multi-depot poultry-distribution routing problem in Thailand.

Hybridizing global metaheuristics with local search has proven effective for routing problems, since local refinement can resolve constraint violations while preserving overall solution structure, for example through 2-opt and route relocation [10], variable neighborhood descent [11], and destroy-repair strategies combining Large Neighborhood Search with construction heuristics. Chowmali and Sukto apply this combination to multi-compartment, heterogeneous-fleet routing for fuel delivery in northeastern Thailand, proposing both a hybrid Fisher and Jaikumar / Adaptive Large Neighborhood Search algorithm (FJA-ALNS) [13] and, in related work, a two-phase heuristic for minimizing driving distance and vehicle count in the same delivery network [12]. Sankul et al. study the exploration-exploitation trade-off in DE applied to the same multi-compartment cold-chain routing context via an adaptive DE (ADE) algorithm [8], while de Melo et al. propose Dual Search Optimization (DSO), a co-evolutionary algorithm that adapts its own penalty function through the co-evolution of solutions and perturbation strategies [26]. Diversity preservation through random re-initialization, crowding-distance tracking, and adaptive mutation rates remains an active area for avoiding premature convergence in these hybrid settings.

Across this body of work, a common gap remains: few DE-based hybrids for MDVRPTW isolate, rather than bundle, the individual contributions of penalty control, local search, and memory-based diversification. This manuscript addresses that gap, extending classical DE to MDVRPTW with a feedback-

adjusted adaptive penalty mechanism and evaluating each constraint-handling component's contribution separately, determining which mechanisms meaningfully improve DE's behavior on this problem rather than presenting only a single bundled result.

3. METHODS

3.1 Problem Representation

MDVRPTW is modeled via a permutation-based representation [3]. The chromosome in this instance is a permutation of customer nodes which is transformed to obtain depot-assigned routes fulfilling travel time, service time, and time-window constraints. After the problem is defined, the corresponding data is loaded, the parameters of the DE are initialized, the population is evaluated and a generational evolution loop follows before the final solution/metrics output is done. The whole solver procedure corresponds to the flow outlined in Figure 1.

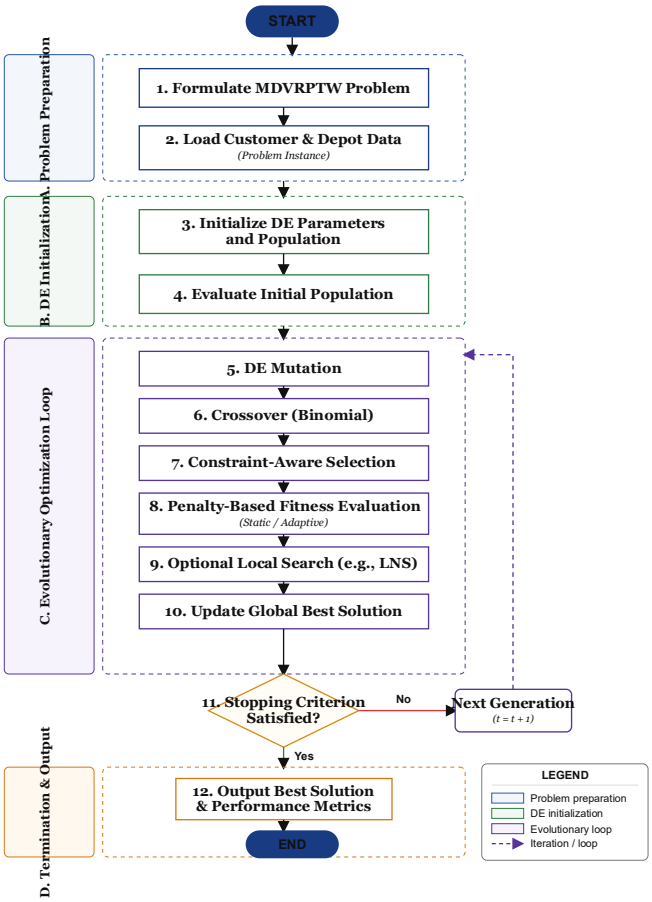


Fig. 1. The Organizational Flowchart for the Proposed Research Strategy

3.2 DE Configuration

The population size and number of generations for the solver that were kept constant all through the study were $NP = 50$, and 200 generations were conducted to get the result. Furthermore, the initial control parameters were $F = 0.8$ and $CR = 0.9$. Instead of keeping F and CR at the same values for the whole run, the solver self-adaptively adjusts the values of the control parameters during the run. This self-adaptive mechanism works in such a way that for each generation, the GA operators F and CR of each member of the population are slightly altered by Gaussian noise and then restricted to valid levels. The solver records F/CR values corresponding to the trial solutions that are successfully chosen (accepted) to be taken into the next generation. These recorded F/CR values are used to estimate the new population-wide F and CR by computing their mean at end of each generation. This change results in search focusing on the parameter settings that have proven to work best for the specific instance being tackled during the run, but as a result, these control parameters will no longer be compared easily to the initial ones towards the end of a run.

3.3 Genetic Operators (Mutation & Crossover)

Mutations affect three non-overlapping chromosomes (except for the target individual). A copy of the base chromosome is subject to some positional swaps given F. In each swap, one customer is moved to the position where they appear in either of the two donor chromosomes. Crossover is implemented via a simple order-crossover scheme (OX): the child gets a block of genes from the target chromosome that is copied from the parent, the rest comes from the mutant chromosome and skips those that were already put there, and as a result, the child remains a valid permutation. Decoding starts with depot assignment where each customer is allocated to the closest depot by Euclidean distance that is computed once and saved for each customer [3], [27]. The next stage is route construction were allocated to a depot customers are inserted in the chromosome order in capacity-feasible routes for that depot.

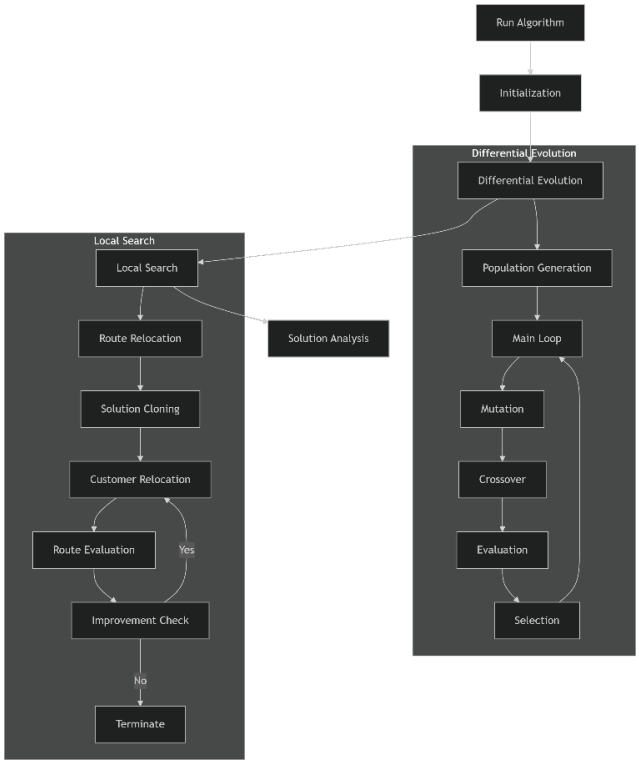


Fig. 2. The Proposed MDVRPTW Framework Combining Classic DE and Route Relocation with Adaptive Penalty Constraints

3.4 Constraint Handling Strategy

i. Adaptive Penalty System

Candidate solutions are compared using a penalized fitness (to be minimized),

$$f(x) = D(x) + w_g \cdot V(x) \quad (1)$$

where $D(x)$ is the total route distance of solution x, $V(x)$ is its total number of time-window violations, and w_g is the penalty weight at generation g. The weight is not raised at fixed intervals. Instead, it is adjusted every 10 generations using the feasibility ratio of the current population, defined as the fraction of individuals with zero time-window violations, which is compared against a target ratio of 0.5. If the ratio falls below the target, the weight is increased by a factor of 1.05, up to a maximum of 100000; if it rises above the target, the weight is decreased by a factor of 0.8, down to a minimum of 1000; otherwise, it is left unchanged.

The weight is therefore clamped to the interval [1000, 100000] throughout execution, and the initial weight w_0 is user-configurable (held at 1000 for all experiments reported here). The feedback-controlled nature of this approach adjusts penalty pressure in line with the population's actual feasibility state, rather than following a fixed schedule that disregards search progress: already-feasible populations experience relaxed constraint pressure and can explore more of the cost landscape, while struggling populations are pushed through increased penalty toward feasible regions.

ii. Solution Representation and Decoding Strategy

Each solution can be defined as permutation of customer indices and the decoding process occurs in two stages. At the first step of this process, customers are assigned to their nearest depot following the Euclidean distance (depot assignment step). At the second stage of decoding, the customers' route is built by adding a given customer to each depot's route in the order given in the chromosome (route construction stage). There is no check for any time window violations at this stage.

```
Algorithm 1: MDVRPTW_SOLVER
RUN_ALGORITHM();;
if no depots OR no customers then
    return ERROR;
end
Initialize: solutionRoutes = [], convergenceData = [];
runID ← current_timestamp(), currentPenalty ← initialPenalty;
population ← INITIALIZE_POPULATION(populationSize);
bestSolution ← NULL;
for generation = 0 to maxGenerations do
    if generation mod 10 = 0 then
        feasibleRatio ← FEASIBLE_RATIO(population);
        if feasibleRatio < 0.5 then
            currentPenalty ← min(currentPenalty × 1.05, 100000);
        else if feasibleRatio > 0.5 then
            currentPenalty ← max(currentPenalty × 0.8, 1000);
    end
    newPopulation ← [];
    foreach target ∈ population do
        Select distinct indices a, b, c from population (excluding target);
        mutant ← CREATE_DIFFERENTIAL_MUTANT(a, b, c, scalingFactor);
        trial ← PERMUTATION_CROSSOVER(target, mutant,
          crossoverRate);
        EVALUATE_WITH_PENALTY(trial, currentPenalty);
        if SHOULD_ACCEPT(trial, target) then
            newPopulation.add(trial);
            if trial.fitness < bestSolution.fitness then
                bestSolution ← trial;
            end
        end
        else
            newPopulation.add(target);
        end
    end
    population ← newPopulation;
    if generation mod 50 = 0 then
        INJECT_RANDOM_SOLUTION(population);
    end
    PLOT_CONVERGENCE(generation, bestSolution.fitness);
    if generation mod 5 = 0 then
        ANIMATE_SOLUTION(bestSolution, generation);
    end
    if enableLocalSearch then
        ROUTE_RELOCATION_LOCAL_SEARCH(bestSolution);
    end
end                                          2
solutionRoutes ← bestSolution.routes;
OUTPUT_METRICS(bestSolution, runID);
VISUALIZE_FINAL_SOLUTION();
return bestSolution;
```

Fig. 3. Main Algorithm Constituting MDVRPTW with Adaptive Penalty DE

```
Algorithm 2: CREATE_DIFFERENTIAL_MUTANT
Input: base, diff1, diff2, F
Output: mutant solution
mutant = base.chromosome.copy();
numSwaps = MAX(1, F × chromosomeLength);
for swapCount = 1 to numSwaps do
    idx1 = RANDOM_INDEX(chromosomeLength);
    customerId = diff1.chromosome[idx1];
    idx2 = FIND_POSITION(diff2.chromosome, customerId);
    if idx2 ≠ −1 then
        SWAP(mutant, idx1, idx2);
    end
end
return Solution(mutant);
```

Fig. 4. Mutation Phase Imposed in the MDVRPTW with Adaptive Penalty DE

Algorithm 3: PERMUTATION_CROSSOVER

Input: target, mutant, CR
Output: trial solution
trial = target.chromosome.copy();
modifications = 0;
for $j = 0$ **to** *chromosomeLength* -1 **do**
  if *RANDOM()* $<$ *CR* ***and*** *target[j]* $\neq$ *mutant[j]* **then**
    customerId = mutant[j];
    currentPos = FIND_POSITION(trial, customerId);
    SWAP(trial, j, currentPos);
    modifications++;
  end
end
if *modifications* $= 0$ **then**
  start = RANDOM_INDEX(chromosomeLength);
  length = RANDOM_INT(1, chromosomeLength − start);
  ROTATE_SEGMENT(trial, start, length);
end
return trial;

Algorithm 4: EVALUATE_WITH_PENALTY

Input: solution, penaltyWeight
routes ← DECODE_CHROMOSOME(solution.chromosome);
totalDistance = 0; totalViolations = 0;
foreach *route* ∈ *routes* **do**
  routeDistance ← CALCULATE_TOTAL_DISTANCE(route);
  routeViolations ← COUNT_VIOLATIONS(route);
  totalDistance += routeDistance;
  totalViolations += routeViolations;
end
solution.fitness = totalDistance + penaltyWeight × totalViolations;

Fig. 5. Crossover Strategy and Penalty Measures for the Adapted DE

A trial solution replaces its target if it is feasible while the target is not; it is rejected if the target is feasible while the trial is not; otherwise, the solution with the lower penalized fitness is retained (Fig. 6).

Algorithm 5: SHOULD_ACCEPT

Input: trial, target
Output: True/False
if *trial.violations* $= 0$ ***and*** *target.violations* > 0 **then**
  return True;
end
if *target.violations* $= 0$ ***and*** *trial.violations* > 0 **then**
  return False;
end
return trial.fitness < target.fitness;

Fig. 6. Feasibility-Prioritized Acceptance Criterion for Trial Solutions

The following section (Figures 7-9) represents the illustration of Hybrid Adaptive Differential Evolution with Large Neighborhood Search and Tabu Mechanism (HADE-LNS-Tabu) imposed in this study.

Algorithm 1: Differential Evolution with LNS and Tabu Search

[1] **Input:** MDVRPTW instance via `solver`, population size NP, differential weight F, crossover rate CR, max generations G, initial penalty weight **Output:** Best feasible solution with minimized fitness DifferentialEvolution Initialize parameters and penalty weights Initialize tabu list and convergence tracking Initialize empty lists `successfulFs` and `successfulCRs`
Create initial population of NP random solutions: **for** *each individual* **do**
Generate random permutation Evaluate solution fitness $bestSolution \leftarrow$ best individual from population
for $g \leftarrow 0$ *to* $G - 1$ **do**
$g \bmod 10 = 0$ Adjust penalty weight based on feasibility ratio
Initialize `newPopulation` Clear `successfulFs` and `successfulCRs`
for *each target in population* **do**
$localF \leftarrow$ AdaptParameter$F$ $localCR \leftarrow$ AdaptParameterCR $mutant \leftarrow$ Mutatepopulation, target index, $localF$ $trial \leftarrow$ OrderCrossovertarget, mutant, $localCR$ Evaluate trial solution Perform LNS local search on trial Re-evaluate trial
if *ShouldAcceptItrial, target* **then**
Add $trial$ to `newPopulation` Record $localF$, $localCR$ **if** $trial$ *better than* $bestSolution$ *and not tabu* **then**
$bestSolution \leftarrow trial$ Add to `tabuList` **else**
Add $target$ to `newPopulation`
Update F and CR using `successfulFs` and `successfulCRs`
$population \leftarrow newPopulation$
if $g \bmod 30 = 0$ **then**
Perform LNS on $bestSolution$ **if** *improved and not tabu* **then**
Update $bestSolution$ Add to `tabuList`
if $g \bmod 5 = 0$ **then**
Animate $bestSolution$ for visualization
Log generation summary
Log total LNS improvement statistics Save metrics to CSV **return** $bestSolution$

Fig. 7. Proposed Hybrid DE Combining Large Neighborhood Search (LNS) and Tabu Search

Algorithm 2: Mutate

[1] Mutatepopulation, currentIndex, localF Randomly select three distinct indices $a, b, c \neq$ currentIndex $base \leftarrow$ population[a] $diff1 \leftarrow$ population[b], $diff2 \leftarrow$ population[c]
$mutant \leftarrow$ copy of $base.chromosome$ $swaps \leftarrow \max(1, \lfloor localF \times length \rfloor)$
for $i \leftarrow 1$ *to* *swaps* **do**
Randomly select index $idx1$ $customerId \leftarrow diff1.chromosome[idx1]$ $idx2 \leftarrow$ position of $customerId$ in $diff2.chromosome$ **if** $idx2 \neq -1$ ***and*** $idx1 \neq idx2$ **then**
Swap $mutant[idx1]$ and $mutant[idx2]$ **return** new Solution with $mutant$

Algorithm 3: OrderCrossover

[1] OrderCrossover$p1$, $p2$, $localCR$ **if** *random()* $> localCR$ **then**
return copy of $p1$ $child \leftarrow$ array of length n filled with -1 Randomly select $start$ and end such that $start < end$ **for** $i = start$ *to* end **do**
$child[i] \leftarrow p1[i]$ $cur \leftarrow (end + 1) \bmod n$ **for** $i = 0$ *to* $n - 1$ **do**
$idx \leftarrow (end + 1 + i) \bmod n$ **if** $p2[idx]$ *not in* $child$ **then**
$child[cur] \leftarrow p2[idx]$ $cur \leftarrow (cur + 1) \bmod n$ **return** $child$

Algorithm 4: LNSLocalSearch

[1] LNSLocalSearchsolution $before \leftarrow$ solution.fitness Randomly remove a fraction of customers from chromosome $remaining \leftarrow$ customers not removed $removed \leftarrow$ removed customers
while *removed is not empty* **do**
$cust \leftarrow$ randomly selected from removed $bestPos \leftarrow 0$, $bestFitness \leftarrow \infty$ **for** *each insertion position in* $remaining$ **do**
Insert $cust$ into $remaining$ at position i Evaluate temporary solution **if** *fitness* $<$ $bestFitness$ **then**
Update $bestFitness$, $bestPos$ Insert $cust$ at $bestPos$ in $remaining$ Update solution.chromosome and re-evaluate **if** *solution.fitness* $<$ *before* **then**
Log improvement

Fig. 8. Exploration and Exploitation Mechanisms Imposed in the Local Search Strategy

Algorithm 5: Helper Functions

[1] AdaptParameterbase Add Gaussian noise to *base* **return** value clipped to allowed bounds
ShouldAcceptttrial, target **if** *trial feasible and target not* **then**
return true target feasible and trial not **return** false **else**
return trial.fitness < target.fitness
AddToTabuListsolution Add chromosome string to `tabuList` If list exceeds size, remove oldest
IsTabusolution **return** true if chromosome string in `tabuList`

Fig. 9. Aspiration Criterion Implementing Tabu Listing

3.5 Local Search and Memory Mechanisms

The Large Neighborhood Search (LNS) [28] operator is applied to every trial solution in every generation when active, as well as once every 30 generations to refine the current best solution specifically. Every application begins by choosing a destroy size,

$$n_{remove} = min\big(25, max(3, \lfloor 0.4 \times n \rfloor)\big) \quad (2)$$

where n is the total customer count, that is, 40% of customers are targeted for removal, subject to a floor of 3 and a cap of 25 regardless of instance size. These n_{remove} customers are selected uniformly at random, without replacement, and are removed from the solution. They are then reinserted one at a time, in random order: for each customer, every candidate position within its (geometrically fixed) depot's current customer sequence is evaluated, and the position yielding the lowest combined cost: that depot's resulting route distance plus its time-window violation penalty, with all other depots' costs held at their current values is selected (greedy best-insertion; ties are broken in favour of the lowest-index position encountered). The destroy-repair procedure itself performs no independent accept/reject test, the reconstructed solution unconditionally replaces the previous one once all removed customers are reinserted [3]. Whether this LNS-modified solution subsequently propagates into the next generation is governed separately, by the DE-level acceptance rule (Section 3.4 and Fig. 6) and, when Tabu memory is enabled, the tabu-membership check described below.

Tabu memory is a fixed size list of up to 15 entries, each of which is the complete chromosome (customer-order permutation) of a solution previously found to be the incumbent best. Testing for membership is done by an exact match only; a candidate solution is tabu if and only if its full chromosome is identical to an entry already present in the list, with no consideration given to distance or similarity. If the list reaches maximum size, then the oldest entry is removed (strict first-in-first-out order; not related to the fitness value). A candidate solution can update the incumbent best only when it improves upon the current best fitness and is not present in the tabu list [10], [18], without any aspiration criteria for exceptionally fit tabu-listed candidates.

4. RESULTS AND FINDINGS

4.1 Overall Performance Comparison

All computational experiments were performed on a system equipped with a 13th Generation Intel Core i7-13700HX processor (2.10 GHz base frequency, turbo boost up to 5.0 GHz) and 16 GB of RAM, running a 64-bit Windows operating system. The algorithms were implemented and executed in single-threaded mode. All reported wall-clock times correspond to the average over 30 independent runs. Table 1 reports total route distance for each of the five configurations: baseline DE, DE+LNS, DE+Tabu, DE+LNS+Tabu, and the full DE+LNS+Tabu+Adaptive hybrid, across four MDVRPTW instances from Vidal et al. (2013): pr11a, pr11b, pr12a, and pr12b.

Table 1. Mean Route Distance ± SD (Best-Worst) Across 30 Repetitions

Instance	Config	Mean ± SD	Min	Max
pr11a	DE	23477.5 ± 579.9	22705.0	25427.9
	DE+LNS	14991.5 ± 307.5	14378.6	15568.5
	DE+Tabu	23441.7 ± 497.9	22689.9	24339.3
	DE+LNS+Tabu	14865.5 ± 470.2	13872.3	15591.2
	+Adaptive	17814.7 ± 757.5	15825.1	19747.4
pr11b	DE	23436.8 ± 568.8	21823.0	24550.0
	DE+LNS	12929.8 ± 500.9	12054.9	13963.3
	DE+Tabu	23470.2 ± 579.9	22014.7	24588.4
	DE+LNS+Tabu	13159.9 ± 374.5	12387.1	13763.7
	+Adaptive	15761.4 ± 679.4	14311.6	17192.6
pr12a	DE	31651.1 ± 669.5	30280.4	32905.2
	DE+LNS	20556.1 ± 479.5	19399.0	21311.8
	DE+Tabu	31645.9 ± 597.0	30459.1	33161.3
	DE+LNS+Tabu	20745.9 ± 565.7	19587.5	21702.8
	+Adaptive	25795.7 ± 911.0	23885.9	27569.0
pr12b	DE	31419.7 ± 679.0	30026.1	32805.8
	DE+LNS	18585.3 ± 434.7	17443.8	19245.3
	DE+Tabu	31327.6 ± 649.6	30463.7	32901.1
	DE+LNS+Tabu	18636.5 ± 515.4	17418.5	19564.1
	+Adaptive	23477.8 ± 1122.2	20817.4	25965.0

The configurations that involved the Large Neighborhood Search (DE+LNS and DE+LNS+Tabu) achieved the shortest mean distance in every instance without exception. Averaged across the four instances, mean distance decreased from baseline DE (27,496.3) to DE+LNS (16,765.7), a 39.0% reduction, while DE+LNS+Tabu (16,852.0) showed a negligible difference from LNS alone, averaging 0.5% higher rather than lower. DE+Tabu without LNS showed no meaningful improvement over baseline DE (27,471.4 vs. 27,496.3, under 0.1% apart). The full adaptive hybrid DE+LNS+Tabu+Adaptive reached a mean distance of 20,712.4, substantially worse than DE+LNS+Tabu, though still considerably better than either non-LNS configuration, since it retains LNS and Tabu's contribution even as the adaptive penalty component actively degrades it. Table 2 reports time-window violation counts for the same configurations and instances.

Table 2. Mean Time-Window Violations ± SD; Feasible Runs / 30

Instance	DE	DE+LNS	DE+Tabu	DE+LNS+ Tabu	+Adaptive
pr11a	271.0 ± 2.1 (0/30)	104.8 ± 2.9 (0/30)	270.9 ± 2.1 (0/30)	102.2 ± 3.0 (0/30)	155.0 ± 7.7 (0/30)
pr11b	231.8 ± 2.2 (0/30)	73.0 ± 3.1 (0/30)	232.6 ± 2.0 (0/30)	74.4 ± 3.2 (0/30)	117.7 ± 8.1 (0/30)
pr12a	368.8 ± 2.5 (0/30)	156.3 ± 3.9 (0/30)	368.4 ± 2.4 (0/30)	156.0 ± 3.6 (0/30)	243.2 ± 14.5 (0/30)
pr12b	308.8 ± 3.6 (0/30)	108.3 ± 4.0 (0/30)	309.2 ± 2.3 (0/30)	108.7 ± 4.8 (0/30)	182.1 ± 14.7 (0/30)

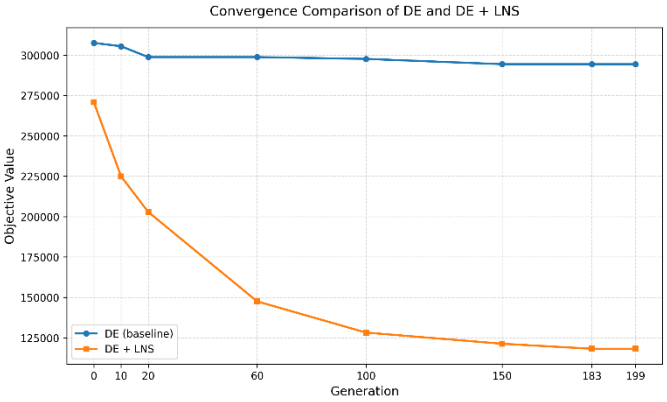


Fig. 10. Results from the total run between the implemented DE variants on selected problem instances

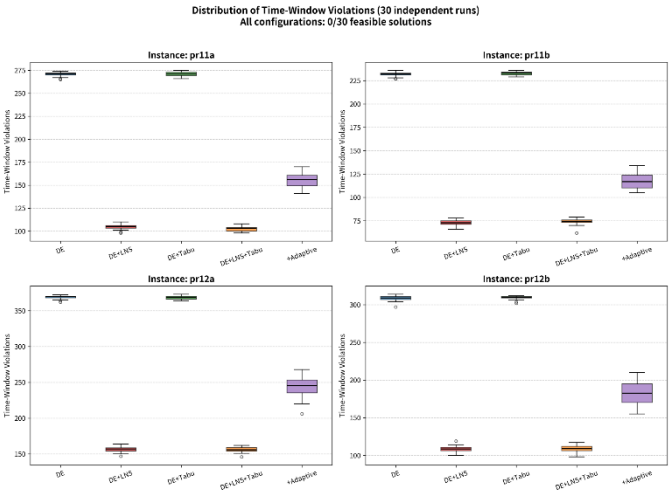


Fig. 11. Distribution of time-window violations across 30 independent runs per configuration on instances pr11a, pr11b, pr12a and pr12b.

Figure 11 shows the distribution of the number of time-window violations in 30 independent runs for each algorithm configuration on the four large-scale instances. Boxes show the interquartile range, horizontal lines indicate the median, whiskers extend to 1.5×IQR, and circles represent outliers. It can be seen that pure DE and DE+Tabu consistently produce extremely high numbers of violations, with very little variation between runs, while the configurations using LNS, with or without Tabu, show much lower median violation counts. The adaptive configuration shows intermediate performance with the greatest variance between runs. Also, it should be noted that none of the 150 runs per instance (30 runs for each of the five configurations) produced a feasible solution.

4.2 Statistical Comparison Across Ablation Variants

Table 3. Raw distance, penalized fitness, and on-time service rate (mean across 30 repetitions)

Instance	Config	Raw Distance	Penalized Fitness	% Customers On-Time
pr11a (n=360)	DE	23477.5	294477.5	24.7%
	DE+LNS	14991.5	119758.2	70.9%
	DE+Tabu	23441.7	294308.4	24.8%
	DE+LNS+Tabu	14865.5	117098.8	71.6%
	DE+LNS+Tabu+Adaptive	17814.7	204016.3	56.9%
pr11b (n=360)	DE	23436.8	255270.1	35.6%
	DE+LNS	12929.8	85929.8	79.7%
	DE+Tabu	23470.2	256036.9	35.4%
	DE+LNS+Tabu	13159.9	87559.9	79.3%
	DE+LNS+Tabu+Adaptive	15761.4	163555.3	67.3%
pr12a (n=480)	DE	31651.1	400484.4	23.2%
	DE+LNS	20556.1	176856.1	67.4%
	DE+Tabu	31645.9	400012.5	23.3%
	DE+LNS+Tabu	20745.9	176779.2	67.5%
	DE+LNS+Tabu+Adaptive	25795.7	311654.2	49.3%
pr12b (n=480)	DE	31419.7	340186.3	35.7%
	DE+LNS	18585.3	126918.6	77.4%
	DE+Tabu	31327.6	340560.9	35.6%
	DE+LNS+Tabu	18636.5	127336.5	77.4%
	DE+LNS+Tabu+Adaptive	23477.8	242717.3	62.1%

Across all four MDVRPTW instances, LNS-containing configurations achieved the lowest mean route distance in every case, reducing mean distance relative to baseline DE by 36.1% (pr11a), 44.8% (pr11b), 35.1% (pr12a), and 40.9% (pr12b). A Friedman test across all five configurations confirmed a statistically significant difference in ranking (χ^2=14.80, $df = 4$, $p = 0.0051$), with DE+LNS (mean rank 1.25) and DE+LNS+Tabu (1.75) ranking consistently best and DE (4.75) and DE+Tabu (4.25) consistently worst. Pairwise exact Wilcoxon tests are directionally unanimous: LNS-containing configurations beat baseline DE, and DE+LNS+Tabu beats the full adaptive-penalty hybrid, in all four instances, but with only four paired observations the smallest attainable two-sided p-value is 0.125, so these pairwise comparisons should be read as consistent rather than independently significant; the omnibus Friedman result is the appropriate basis for the overall significance claim. Time-window violations followed the same pattern: DE+LNS reduced mean violations by 61.3%, 68.5%, 57.6%, and 64.9% relative to baseline across the four instances, though no configuration achieved full feasibility on any instance. Adaptive penalty control substantially worsened both cost and feasibility relative to DE+LNS+Tabu in every instance; mean fitness increased by 74.2%, 86.8%, 76.3%, and 90.6% respectively, while on-time service rate simultaneously *dropped* (e.g., 77.4% → 62.1% for pr12b) confirming that the mechanism does not trade distance for feasibility but degrades both simultaneously.

4.3 Ablation Study

The findings from the ablation show three main effects. First, economically efficient route planning was achieved by LNS, cutting mean route distance by 35–45% relative to baseline DE across all four instances regardless of size (pr11a and pr11b at 360 customers; pr12a and pr12b at 480 customers). Second, Tabu memory has no meaningful contribution when

separated from LNS; its effect only becomes visible in combination with LNS, and even then, produces only a marginal difference relative to DE+LNS alone. Third, contrary to the initial hypothesis, the penalty mechanism brings negative consequences for solution quality when layered onto LNS and Tabu, worsening both route cost and feasibility rather than improving either. The likely explanation is that the adaptive mechanism continues escalating penalty pressure toward a feasibility target that remains unattainable within the evaluated computational budget, so the search is pushed away from good-cost regions without ever reaching the feasible region it is being pushed toward. This result goes against the initial hypothesis concerning the impact of hybrid systems.

4.4 Computational Discussion

This paper presents a study into mechanisms for maintaining the time-window feasibility of solution without compromising upon the global exploration capability of the core Differential Evolution algorithm. The results demonstrate that a significant feasibility gap exists even after employing penalty- and search-based mechanisms: (i) Large Neighborhood Search substantially reduces route cost, but none of the five configurations tested including the most comprehensive hybrid, managed to eliminate time-window violations entirely across the four benchmark instances. (ii) Unlike a purely cosmetic constraint-handling layer, violation levels here strongly respond to which mechanisms are active: LNS-containing configurations cut mean violations by 57–68% relative to baseline DE while Tabu alone yields no meaningful change and the adaptive penalty configuration falls between the two. This indicates that the constraint-handling mechanisms are having a genuine, differentiated effect upon feasibility, despite none closing the gap completely within the evaluated budget: a dedicated time-window repair operator was omitted as explicit future work (Section 5) as this solver did not include one. The larger instances tested, pr12a and pr12b (480 customers each), followed the same qualitative pattern as the smaller pr11a/pr11b instances (360 customers): LNS-containing configurations reduced distance by a comparable margin regardless of instance size, while absolute violation counts scaled with instance size (368–369 mean violations on pr12a/pr12b under baseline DE, versus 232–271 on pr11a/pr11b), indicating that the LNS effect is not an artifact of small problem size, and that the feasibility gap widens rather than narrows as the problem grows.

It is worth mentioning that the study does not intend for its DE-based ablation framework to compete on solution quality (in this benchmark) with state-of-the-art methods. The best-known solutions for these instances can be found in the PyVRP/Instances repository [29], and even the original Hybrid Genetic Search with Adaptive Diversity Control proposed together with this benchmark reports a 0.71% average gap to best-known [14] ; current solvers have comparable or better gaps under significantly larger computational budgets than the 200-generation limit used throughout this study. Therefore, a direct percentage-gap comparison was omitted for it to primarily reflect the deliberate budget and objective difference between a mechanism-isolation ablation study and a solution-quality-focused solver, rather than giving further insight into the constraint-handling results reported above.

5. CONCLUSION AND FUTURE WORK

This study investigates the possibility to improvise classical Differential Evolution competitive on the MDVRPTW through a series of targeted, individually adaptive modifications to its constraint-handling capabilities, rather than through a wholesale redesign of the optimization paradigm. An ablation study was carried out on four MDVRPTW instances from the set of Vidal (2013): pr11a, pr11b, pr12a, and pr12b, using DE, DE+LNS, DE+Tabu, DE+LNS+Tabu, and the full DE+LNS+Tabu+Adaptive configuration with 30 independent runs of each variant per instance.

The Large Neighborhood Search was found to be the dominating factor in terms of route cost reduction: mean distance was decreased by 36.1%, 44.8%, 35.1%, and 40.9% for the four instances compared to the baseline DE. A Friedman test across the five configurations revealed statistically significant differences in the rankings ($\chi^2 = 14.80$, $df = 4$, $p = 0.0051$), with LNS-containing configurations showing a clear advantage. The tabu memory usage, on the other hand, had little effect when considered in isolation, having produced results no better than the baseline DE by more than 0.5% in either direction. Notably, the adaptive penalty mechanism, the very factor that motivated this research's alternative hypothesis, had an adverse effect on both the route's distance and time-window violations when added to the DE+LNS+Tabu combination. This feature increased the mean fitness value by 74.2%, 86.8%, 76.3%, and 90.6% for the four instances while decreasing the on-time service rate (for instance, from 77.4% to 62.1% for the pr12b instance). Thus, the mechanism seems to offset the deterioration of the cost and feasibility of the solution rather than strike a balance between them. The pairwise comparisons were performed using the exact Wilcoxon tests, which were all positive in direction of the alternative hypothesis for every instance. However, with only four comparisons per instance, the minimum possible two-sided p-value is 0.125, and these results are reported as "tending towards significance" for the purpose of discussion, with the omnibus Friedman test being used as a primary basis for the conclusions. None of the configurations managed to find a feasible solution (with respect to time-windows) for any of the instances within the given budget. The findings discussed above are reported as the main contributions of this research, including the identification of the factors that failed to contribute to the optimization process as described above. The separate analysis of each method's impact is what allowed the surprising result of the adaptive penalty mechanism's detrimental effect to become evident, instead of only reporting the combined improvements of the hybrid approach.

Throughout the course of this study, several research directions emerged that appear to be promising avenues for endeavour. Firstly, it would be interesting to see how the repair algorithm would work if it were used directly on the solution, for instance, in the case of dynamic adjustment with time windows to avoid infeasibility as was the case for all scenarios in this research. Secondly, there is a need to consider other metaheuristic methods such as Genetic Algorithms, Ant Colony Optimization, and Adaptive Large Neighborhood Search to establish a comprehensive understanding of DE's ability to handle constraints. Thirdly, the improvement of the adaptive penalty function seemed like a logical direction to take, given the issues associated with its application in this study. Lastly, there is a need to expand the solver's capabilities to account for

multiple routing objectives and incorporate adaptive learning mechanisms to actualize a truly viable application in practice for multi-depot VRPs.

ACKNOWLEDGEMENT

Farid Morsidi was responsible for several aspects of work on this project including conceptualisation, methodology, validation, analysis, curation, investigation, management of resources, software, visualisation, and writing. The paper's source code would be made available at the author's Github repository (https://github.com/blackcontractor90) after the paper's publication.

REFERENCES

[1] D. Wang, J. Jiang, R. Ma, and G. Shen, "Research on Hybrid Real-Time Picking Routing Optimization Based on Multiple Picking Stations," *Math. Probl. Eng.*, vol. 2022, no. 1, pp. 1–15, Apr. 2022, doi: https://doi.org/10.1155/2022/5510749

[2] C.-M. Chen, S. Lv, J. Ning, and J. M.-T. Wu, "A Genetic Algorithm for the Waitable Time-Varying Multi-Depot Green Vehicle Routing Problem," *Symmetry (Basel).*, vol. 15, no. 1, p. 124, 2023, doi: https://doi.org/10.3390/sym15010124

[3] F. Morsidi, "An Intensification-Enhanced Adaptive Hybrid Memetic Algorithm for the Multi-Depot Vehicle Routing Problem with Time Windows," *Appl. Model. Simul.*, vol. 10, no. 1, pp. 50–63, 2026, url: https://arqiipubl.com/ojs/index.php/AMS_Journal/article/view/1127

[4] A. Shuib and P. M. Ibrahim, "a Mixed Integer Goal Programming (Migp) Model for Donated Blood Transportation Problem – a Preliminary Study," *Malaysian J. Comput.*, vol. 6, no. 2, p. 835, 2021, doi: https://doi.org/10.24191/mjoc.v6i2.10751

[5] N. S. Ibrahim, A. Shuib, and Z. A. Zaharudin, "Modified Hungarian Method for Lecturer-To- Course Assignment : a Multi-Objective Mathematical Programming Model for Optimizing Preferences and Competency (Pc Mo- Mhm)," *Malaysian J. Comput.*, vol. 10, no. 2, pp. 2308–2321, 2025, doi: https://doi.org/10.24191/mjoc.v10i2.5562

[6] K. Moonsri, K. Sethanan, and K. Worasan, "A Novel Enhanced Differential Evolution Algorithm for Outbound Logistics of the Poultry Industry in Thailand," *J. Open Innov. Technol. Mark. Complex.*, vol. 8, no. 1, p. 15, 2022, doi: https://doi.org/10.3390/joitmc8010015

[7] J. Sun, X. Liu, T. Back, and Z. Xu, "Learning Adaptive Differential Evolution Algorithm from Optimization Experiences by Policy Gradient," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 666–680, 2021, doi: https://doi.org/10.1109/TEVC.2021.3060811

[8] S. Sankul, N. Supattananon, R. Akararungruangkul, and N. Wichapa, "An adaptive differential evolution algorithm to solve the multi-compartment vehicle routing problem: A case of cold chain transportation problem," *Int. J. Prod. Manag. Eng.*, vol. 12, no. 1, pp. 91–104, 2024, doi: https://doi.org/10.4995/ijpme.2024.19928

[9] H. Budiman, S. Li Wang, F. Morsidi, T. Foo Ng, and S. Chin Neoh, "Self-Adaptive Ensemble-based Differential Evolution with Enhanced Population Sizing," in *2020 2nd International Conference on Cybernetics and Intelligent System (ICORIS)*, IEEE, Oct. 2020, pp. 1–6. doi: https://doi.org/10.1109/ICORIS50180.2020.9320767

[10] F. Morsidi, "Distribution Path Segmentation Using Route Relocation and Savings Heuristics for Multi-Depot Vehicle Routing," *Malaysian J. Sci. Adv. Technol.*, vol. 3, no. 2, pp. 72–80, May 2023, doi: https://doi.org/10.56532/mjsat.v3i2.154

[11] J. F. Castaneda Londono, R. A. G. Rendon, and E. M. T. Ocampo, "Iterated local search multi-objective methodology for the green vehicle routing problem considering workload equity with a private fleet and a common carrier," *Int. J. Ind. Eng. Comput.*, vol. 12, no. 1, pp. 115–130, 2020, doi: https://doi.org/10.5267/j.ijiec.2020.8.001

[12] W. Chowmali and S. Sukto, "A novel two-phase approach for solving the multi-compartment vehicle routing problem with a heterogeneous fleet of vehicles: A case study on fuel delivery," *Decis. Sci. Lett.*, vol. 9, no. 1, pp. 77–90, 2020, doi: https://doi.org/10.5267/j.dsl.2019.7.003

[13] W. Chowmali and S. Sukto, "A hybrid fja-alns algorithm for solving the multi-compartment vehicle routing problem with a heterogeneous fleet of vehicles for the fuel delivery problem," *Decis. Sci. Lett.*, vol. 10, no. 4, pp. 497–510, 2021, doi: https://doi.org/10.5267/j.dsl.2021.6.001

[14] 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," *Comput. Oper. Res.*, vol. 40, no. 1, pp. 475–489, Jan. 2013, doi: https://doi.org/10.1016/j.cor.2012.07.018

[15] F. Morsidi, "Multi-depot dispatch deployment analysis on classifying preparedness phase for flood-prone coastal demography in Sarawak," *J. ICT Educ.*, vol. 9, no. 2, pp. 175–190, Dec. 2022, doi: https://doi.org/10.37134/jictie.vol9.2.13.2022

[16] M. Alweshah *et al.*, "Vehicle routing problems based on Harris Hawks optimization," *J. Big Data*, vol. 9, no. 1, 2022, doi: https://doi.org/10.1186/s40537-022-00593-4

[17] Y. Shi, L. Lv, F. Hu, and Q. Han, "A heuristic solution method for multi-depot vehicle routing-based waste collection problems," *Appl. Sci.*, vol. 10, no. 7, 2020, doi: https://doi.org/10.3390/app10072403

[18] J. Sidorova and S. C. Kaja, "A New Approach for Solving the Disruption in Vehicle Routing Problem during Delivery," *Международный Научный Журнал "Современные Информационные Технологии И Ит-Образование,"* vol. 16, no. 2, pp. 371–378, 2020, doi: https://doi.org/10.25559/sitito.16.202002.371-378

[19] Y. Christopher, S. Wahyuningsih, and D. Satyananda, "Study of variable neighborhood descent and tabu search algorithm in VRPSDP," *J. Phys. Conf. Ser.*, vol. 1872, no. 1, 2021, doi: https://doi.org/10.1088/1742-6596/1872/1/012002

[20] Y. Shen, M. Liu, J. Yang, Y. Shi, and M. Middendorf, "A hybrid swarm intelligence algorithm for vehicle routing problem with time windows," *IEEE Access*, vol. 8, pp. 93882–93893, 2020, doi: https://doi.org/10.1109/ACCESS.2020.2984660

[21] S. Wahyuningsih, D. Satyananda, and L. T. Oktoviana, "Implementation of the ACS-RVND algorithm on the VRP variant and its application to distribution optimization," *J. Phys. Conf. Ser.*, vol. 2157, no. 1, 2022, doi: https://doi.org/10.1088/1742-6596/2157/1/012025

[22] S. Zhu, H. Fu, and Y. Li, "Optimization research on vehicle routing for fresh agricultural products based on the investment of freshness-keeping cost in the distribution process," *Sustain.*, vol. 13, no. 14, 2021, doi: https://doi.org/10.3390/su13148110

[23] N. D. Lagaros, M. Kournoutos, N. A. Kallioras, and A. N. Nordas, "Constraint handling techniques for metaheuristics: a state-of-the-art review and new variants," *Optim. Eng.*, vol. 24, no. 4, pp. 2251–2298, 2023, doi: https://doi.org/10.1007/s11081-022-09782-9

[24] H. Bae and I. Moon, "Multi-depot vehicle routing problem with time windows considering delivery and installation vehicles," *Appl. Math. Model.*, vol. 40, no. 13–14, pp. 6536–6549, 2016, doi: https://doi.org/10.1016/j.apm.2016.01.059

[25] G. S. Kashyap, A. E. I. Brownlee, O. C. Phukan, K. Malik, and S. Wazir, "Roulette-Wheel Selection-Based PSO Algorithm for Solving the Vehicle Routing Problem with Time Windows," 2023, doi: https://doi.org/10.48550/arXiv.2306.02308

[26] V. V. de Melo, A. M. Nascimento, and G. Iacca, "A co-evolutionary algorithm with adaptive penalty function for constrained optimization," *Soft Comput.*, 2024, doi: https://doi.org/10.1007/s00500-024-09896-5

[27] S. Suwannarongsri, "Optimal Solving Multi-Depot Vehicle Routing Problem By Modern Metaheuristic Algorithm," *Int. J. Innov. Comput. Inf. Control*, vol. 19, no. 6, pp. 1753–1767, 2023, doi: https://doi.org/10.24507/ijicic.19.06.1753

[28] F. Morsidi, "Scalable Metaheuristic Optimization of Asymmetric and Clustered TSP Variants using Iterated Local Search," *J. Telecommun. Electron. Comput. Eng.*, vol. 18, no. 1, pp. 35–44, 2026, doi: https://doi.org/10.54554/jtec.2026.18.01.005

[29] N. A. Wouda, L. Lan, and W. Kool, "PyVRP: a high-performance VRP solver package," *INFORMS J. Comput.*, vol. 36, no. 4, pp. 943–955, Mar. 2024, doi: https://doi.org/10.1287/ijoc.2023.0055