Population-Based Iterated Local Search Approach for Dynamic Vehicle Routing Problems

Nasser R. Sabar, Say Leng Goh, Ayad Turky, and Graham Kendall, *Senior Member, IEEE*

***Abstract*— This article addresses the dynamic vehicle routing problem (DVRP). DVRP is a challenging variation of the classic vehicle routing problem in which some customers are not known in advance. The objective is to incorporate new customers into the schedule as they become known while still attempting to minimize the cost of serving all customers without violating the problem constraints. This work proposes an effective population-based approach that integrates various algorithmic components to address DVRP. The approach combines a local search algorithm with various evolutionary operators (crossover and mutation) in an adaptive manner. To promote diversity, the proposed approach utilizes a population of solutions and uses a quality-and-diversity strategy to retain only promising solutions. The well-known 21 DVRP benchmark instances are utilized to test the performance of the proposed approach. An experimental comparison is carried out to assess the contribution of the integrated components. Results demonstrate that the integrated components significantly improve search performance. It is also shown that the proposed approach produces new best results for several instances when compared with the best methods reported in the literature.**

***Note to Practitioners*—This work deals with dynamic vehicle routing problems (DVRPs). In DVRP, only limited information is available at the start, and new information is revealed over time. It proposes an effective hybrid approach to tackle this problem. The proposed approach combines evolutionary operators and a local search approach in an adaptive manner to exploit the benefits of each algorithmic component. The proposed approach produces several new best-known solutions. The experimental results demonstrate that the proposed approach is very effective in dealing with DVRP and can help decision-makers in designing best-routing solutions.**

***Index Terms*— Dynamic optimization, evolutionary algorithm, metaheuristic, vehicle routing problem (VRP).**

Manuscript received 29 November 2020; revised 23 March 2021 and 17 June 2021; accepted 8 July 2021. Date of publication 13 August 2021; date of current version 13 October 2022. This article was recommended for publication by Associate Editor S. Rathinam and Editor J. Li upon evaluation of the reviewers' comments. *(Corresponding author: Nasser R. Sabar.)*

Nasser R. Sabar is with the Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3086, Australia (e-mail: n.sabar@latrobe.edu.au).

Say Leng Goh is with the Optimization Research Group (OPT), Faculty of Computing and Informatics, Universiti Malaysia Sabah Labuan, Labuan 88400, Malaysia (e-mail: gohsayleng@yahoo.com).

Ayad Turky is with the Intelligent Technology Innovation Lab (ITIL) Group, Discipline of Information Technology, College of Engineering and Science, Victoria University, Melbourne, VIC 3011, Australia (e-mail: ayad.turky@vu.edu.au).

Graham Kendall is with the School of Computer Science, University of Nottingham Malaysia, Semenyih 43500, Malaysia (e-mail: graham.kendall@nottingham.edu.my).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASE.2021.3097778.

Digital Object Identifier 10.1109/TASE.2021.3097778

I. Introduction

THE vehicle routing problem (VRP) is one of the many combinatorial optimization problems [1], [2]. It is known as the hardest variant of transportation problem that has been intensively studied in the scientific literature [3]. Many real-world applications, particularly in logistics and transportation, have a strong relationship with the VRP, which motivates researchers and practitioners, across many communities to investigate the problem [4], [5]. Problems, such as the traveling salesman problem [6], are obviously closely related to the VRP, but many other problems that may not, at first sight, be related also to have a strong relationship with the VRP. This includes the knapsack problem and bin packing [7].

In the classic VRP, the goal is to derive a group of routes to deliver goods to a group of customers, minimizing the cost (e.g., time, fuel, and distance) while adhering to the imposed constraints [8]. Many VRP models have been introduced over the years. Each model involves different constraints to represent real-world scenario [4], [8]–[10]. Both the classic VRP and developed variants have been classified as NP-hard problems [4], [8]. Exact methods have been utilized to find the optimal solution for VRP instances [8]. However, their computational times grow exponentially as the problem instance size increases. To tackle medium-/large-sized instances, researchers and practitioners have often called upon metaheuristic algorithms because they can often produce a good quality solution in modest computational times, but at the expense of the optimality [4], [11]. Examples of these metaheuristic algorithms are simulated annealing [12], tabu search [13], [14], variable neighborhood algorithms [15], [16], genetic algorithms [17], and ant colony systems [18].

Many real-world problems involve a dynamic aspect or must cope with the uncertain environment [19]. In some VRP applications, logistics and transportation managers must deal with dynamic situations. For example, a few customer orders may appear or cancel their orders during the distribution process [20]. Another critical aspect is the traffic condition that often changes dramatically during the working time. Thus, the dynamic vehicle routing problem (DVRP) was introduced to cope with the dynamic aspect of a real-world VRP variant [19]. In DVRP, at least one aspect of the problem may change over time. These changes may affect the problem constraint(s), the objective function(s), customer demands, or vehicle workload [19].

This work addresses the DVRP variant that involves dynamic customer requests. In this DVRP, not all customers are known in advance, and changes may occur at any time [20]. That is, after designing the routing plan and the vehicles have departed to deliver customer demands, new orders from customers might appear. When this is the case, a solution to the DVRP must now include the new customers in the routing plan while still attempting to minimize the overall travel cost. Hence, the goal of the optimization method is not only to minimize the traveling cost of the routing plan but to also accommodate these changes into the schedule. Despite its real-world application, the DVRP has received little attention from researchers [19]. The DVRP has been categorized as an NP-hard problem, with metaheuristic algorithms being successfully utilized [19].

Existing approaches use time slices (periods), a concept introduced in [20] to handle DVRP. This concept divides the day into periods of equal duration. Then, an optimization approach is periodically executed to solve a static problem corresponding to the current period. These approaches can be categorized as periodic reoptimization approaches. Examples of metaheuristic periodic reoptimization approaches that have been proposed for the DVRP are: greedy randomized adaptive search procedure [21], tabu search [22], ant colony optimization algorithms [21], [23], [24], genetic algorithms [22], [25], variable neighborhood search algorithms [26], memetic approach [27], monarch butterfly algorithm [28], brain storm optimization algorithm [29], and particle swarm optimization algorithms [30]–[32].

This work proposes an effective periodic reoptimization approach for DVRP. The proposed approach integrates a local search (LS) algorithm with evolutionary operators in an adaptive manner. Our contribution can be summarized as follows.

1) We develop an effective hybrid approach for DVRP that integrates an iterated local search (ILS) algorithm with several algorithmic components. First, we utilize the skewed variable neighborhood descent (SVND) as an LS algorithm within ILS to search for a local optimum solution. Second, to effectively explore a new area around the current local optimum, we propose an adaptive multioperator perturbation procedure, which relies on three different but complementary rules to evolve a new solution. Finally, we use the exponential Monte Carlo criterion (EMC) as an acceptance procedure for the ILS.
2) A traditional ILS and other LS algorithms rely only on the best solution to center the search. Hence, the area explored by previous good solutions is often forgotten. However, these solutions might be very useful to guide the search toward promising regions that have not explored yet, especially in a dynamic environment because the search landscape keeps changing. To address this phenomenon, we propose a population-based ILS that uses a set of solutions to handle the dynamic changes. The population contains a group of high-quality and diverse solutions. These solutions are utilized to construct a new starting solution for LS algorithm. To promote diversity, we use a quality-and-diversity updating strategy to update the solutions in the population.
3) We have tested the proposed approach on 21 DVRP benchmark instances [20]. These instances have been used by many researchers. The proposed approach compares favorably with the best methods reported in the literature. Our approach produces 17 new best-known results in reasonable computational times. It obtained better average results across all instances and is shown to be statistically better than other methods.

The rest of this article is organized as follows. In Section II, we present the VRP and DVRP formulations. The proposed ILS is described in Section III. Section IV discusses the experiment and parameter settings, while the results and comparisons are reported in Section V. The conclusion of this work is given in Section VI.

II. Problem Description

This section first presents the description and formulation of the classical VRP and then DVRP.

A. Vehicle Routing Problem

The classical VRP can be modeled using a graph $G(V, E)$ [8]. In this model, V is a set of vertices and E is a group of edges. V represents the location of customers $V = \{v_0, v_1, \ldots, v_n\}$, while edges $E = \{(v_i, v_j) : v_i, v_j \in V,\ i < j\}$ indicate connection between locations v_i and v_j. In addition, each edge has a value indicating the travel time between v_i and v_j denoted as c_{ij}. A matrix $C = (c_{ij})$ can be defined to store the travel time between each pair of locations. In this model, v_0 is used as the depot and it has m vehicles of the same capacity, Q. All vehicles start their route from v_0, visit the allocated locations, and then go back to v_0. The set of vehicle routes is denoted as $R_1, \ldots, R_m$. All $v_1, \ldots, v_n$, except v_0, requests q_i goods to be picked or delivered as well as a servicing time δ_i. The goal is to generate $R_1, \ldots, R_m$ routes to serve all customer demands at minimal total travel distance while ensuring the problem constraints have not been violated [8], [33].

1) Each vehicle starts it journey from v_0, visits all the assigned customer locations, and returns to v_0.
2) The total demand allocated to the vehicle must not exceed its capacity.
3) All customers must be visited once only.
4) The travel time duration of each vehicle route must not be greater than the upper bound.

Each route R_i involves a set of locations and denoted as $R_i = \{v_{\pi(0)}, v_{\pi(1)}, \ldots, v_{\pi(n+1)}\}$, where $v_{\pi(j)} \in V$ $(j \in [0; n_i + 1])$, $v_{\pi(0)}$ and $v_{\pi(n_i+1)} = 0$ represent the depot and n_i is the number of locations in R_i. The cost (C) of R_i is calculated as follows [8]:

$$C(R_i) = \sum_{j=0}^{n_i} c_{\pi(j),\pi(j+1)}. \tag{1}$$

The total cost [or the objective function (f)] for all routes (a complete solution) is calculated as follows:

$$f = \sum_{i=1}^{m} C(R_i). \tag{2}$$

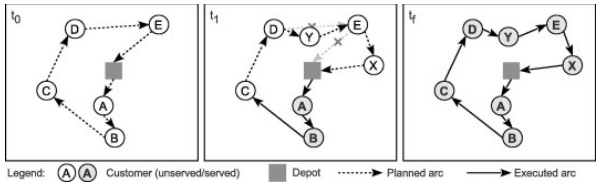

Fig. 1. Example of DVRP [19].

The calculation in (2) should satisfy the vehicle capacity constraint in the following equation:

$$\text{Demand}(R_i) = \sum_{j=1}^{n_i} q_\pi(j) \leq Q \tag{3}$$

where $q_\pi(j)$ is the demand of $\pi(j)$ and Q represents the maximum capacity of vehicle.

In addition to the capacity constraint, the prespecified traveling time bound (T) should not be violated

$$\text{Time}(R_i) = \sum_{j=0}^{n_i} c_{\pi(j),\pi(j+1)} + \sum_{j=1}^{n_i} \delta_{\pi(i)} \leq T. \tag{4}$$

B. Dynamic Vehicle Routing Problem

In this work, we deal with the DVRP model that was introduced in [20] and further refined in [21]. In this DVRP model, new orders are revealed over time [19], [34]. Hence, these new orders should be added to the current schedule by either adding them to current routes or creating a new one for them. Like the classical VRP, the goal of the optimization process is to incorporate new customers into the schedule as they become known while still attempting to minimize the cost of serving all customers without violating the problem constraints. An instance of DVRP scenario is given in Fig. 1 [19]. In this instance, we have known customers (A, B, C, D, and E) and customers that appear over the time. Initially, the vehicle departs the depot at time t_0 to serve A, B, C, D, and E. While the vehicle on the route, two new customers X and Y appear at time t_1. Thus, the vehicle must include them into the current routing plan. Consequently, at time t_f, the vehicle routing plan changed to visit customers A, B, C, D, Y, E, and X. This instance illustrates how the DVRP vehicle routing plan changes over time to accommodate new customers, which shows the real-time communication between the depot and vehicle driver [19].

The model we utilize in this work first divides the DVRP into a sequence of static VRPs and then solves the static VRPs using the proposed algorithm. Each working day D is split into a set of discrete periods n_{ts} of the same length D/n_{ts}. Each period represents one static VRP. The idea of using periods is to assign a fixed time for each static problem [21]. This model can be classified as a dynamic periodic reoptimization routing problem [19]. The model has two parameters: 1) a cutoff time, T_{co}, which is used to allow the system to postpone orders that have been received after T_{co} to the following day to limit the percentage of the advance requests over a day, and 2) an advanced commitment time (T_{ac}), which allows the system to communicate with the driver before leaving the most recently serviced customer to give the driver enough time to react before handling the new order. $T_{\text{ac}} = 0$ indicates that all requests are allocated at the last possible moment. The settings of the DVRP parameters are presented in Section IV-B1.

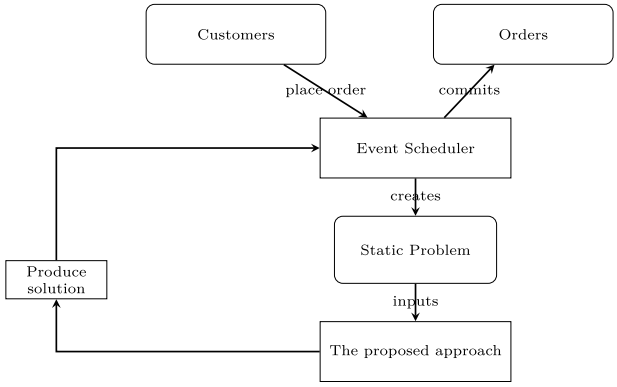

Fig. 2. Event scheduler of DVRP.

Fig. 2 shows the proposed DVRP solver, which has two parts (the event scheduler and the proposed approach). The event scheduler manages customer requests (place orders), assigns customers to vehicles (commits), and generates a static VRP problem as an input for the proposed approach. The role of the proposed approach is to solve the static VRP problem by generating the least cost routing plans. The event scheduler works as follows. First, it generates a static problem for the first period using the orders that were not serviced in the previous working day. It then calls the proposed approach to solve the generated static problem. Next, the generated solution (routing plan) by the proposed approach is committed to the vehicle to serve the current customers. If a new customer order appears during the current period, then it must be postponed to the end of the current period. After that, the event scheduler generates a static VRP problem for the next period, which consists of the orders that missed the service in the previous period. It should be noted that when the event scheduler commits orders to a vehicle, this vehicle is either serving customers or traveling to a location. The event scheduler will also update the state and the position of vehicles and customers after finishing the commitment phase.

III. Proposed Approach

Hybrid approaches combine several interacting modules (heuristics or algorithmic components) in a unified framework to efficiently solve computational search problems [35]–[39]. Combining several different components of various algorithms in a complementary manner can result in an effective solution methodology. This work proposes a population-based approach to effectively deal with DVRP. The proposed approach makes use of various algorithmic components in a complementary manner to attain high-quality solutions. More precisely, we integrate evolutionary operators and a population of solutions within ILS algorithm to devise an effective approach that can generate high-quality solutions for DVRP.

Fig. 3 shows the flowchart of the proposed approach. It consists of ILS components, population of solutions, and the population update procedure. The approach starts by setting the parameters (Section III-A) and initializing the population

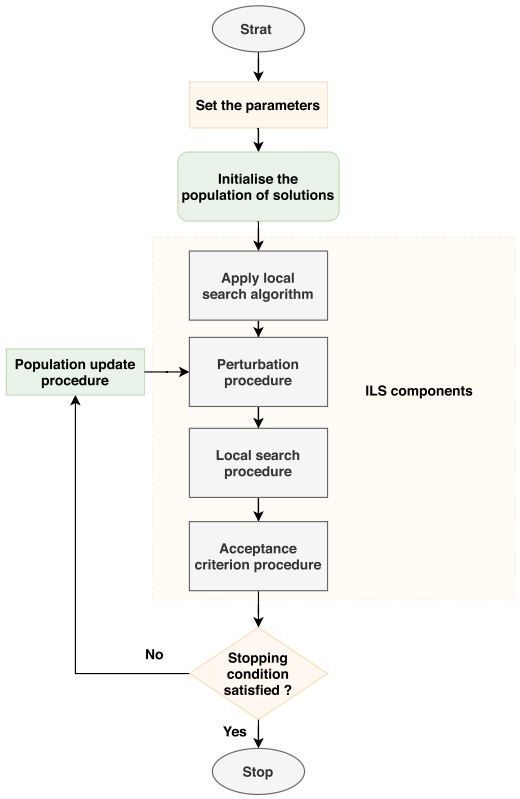

Fig. 3. Flowchart of the proposed approach.

of solutions (Section III-B). Then, it successively calls ILS components to improve the given solution and the update procedure to update the population of solutions until the predefined stopping condition is reached.

The components of the proposed approach are presented in the following.

A. Set the Parameters

The parameters of the proposed approach are set in this step. These are the total number of iterations, population size, and the termination condition of the search. The utilized values are discussed in Section IV-B.

B. Initializing the Population of Solutions

To effectively solve DVRP, the optimization approach should be able to cope with the dynamic aspect of the problem and continuously seek for the best solution. To this end, using a population of solutions within the optimization approach appears highly appropriate for DVRP because solutions can be allocated in different regions of the search space to track the DVRP changes [40]. In addition, a solution for a new change can be created using existing ones. This work utilizes a population of solutions to search the solution space of the DVRP model. The population involves a set of diverse solutions. We use the random permutation procedure to generate the initial set of solutions that randomly add all customer orders from the previous working day into the current plan, as in [21]. By using this method, we can generate several solutions that are spread across the search space. The solutions

Algorithm 1 Basic ILS

1 Set $T - Iter$; $It = 0$;
2 $S_i \leftarrow GenerateInitialSolution()$;
3 $S_c \leftarrow Localsearch(S_i)$;
4 **while** $It <$ *T-Iter* **do**
5 $S_p \leftarrow$ Perturbation procedure (S_c) ;
6 $S_n \leftarrow$ Local search procedure (S_p) ;
7 $S_c \leftarrow$ Acceptance criterion procedure $(S_c, S_n, history)$;
8 $It = It + 1$;
9 **end**
10 Return the best solution;

in the population are then sorted based on their quality, which is represented by the objective function [see (2)].

C. ILS Components

ILS is a single-solution-based optimization algorithm, proposed in [41]. Like most LS algorithms, ILS starts with an initial solution and then carries out an LS, seeking a local optimum. It then modifies this solution to explore a new area in the neighborhood of the current one. Next, ILS carries out another LS. This is repeated until the termination condition is satisfied. ILS can be viewed as iterative calls of the following three procedures: 1) LS; 2) a perturbation to provide a new solution for the next LS round; and 3) the acceptance criterion to decide to accept or reject the resultant solution. The pseudocode of the basic ILS algorithm is shown in Algorithm 1 [41], [42].

First, ILS sets the total number of iterations $(T - Iter)$ and the current iteration indicator It (line 1). In line 2, ILS calls $GenerateInitialSolution()$ (Section III-B) to generate S_i as an initial solution. In line 3, S_i will be improved by the LS algorithm. The new one will be saved as S_c. Then, at each iteration (lines 4–9), it applies the perturbation procedure (line 5) to construct a new starting solution, S_p, via perturbing S_c. Next, it calls the LS procedure to improve S_p and saves the improved solution as S_n (line 6). In line 7, based on utilized the acceptance criterion, S_n is either accepted and replaced with S_c or rejected. Finally, in line 8, ILS updates It and checks the halting condition. If satisfied, ILS will stop and print the best-found solution. Otherwise, it will start a new iteration.

Though ILS has successfully addressed many optimization problems, its performance is highly affected by the three procedures: LS procedure, perturbation procedure, and the acceptance criterion procedure. This is because different problems often require an ILS that has been tuned to the specific instance. To address these issues, the proposed approach adaptively combines various evolutionary operators with the ILS. First we employ the SVND [43] within ILS to search for local optima solutions. SVND uses several neighborhood structures to explore the search space. The main idea behind SVND is that distinct neighborhood structures generate different search paths; thus, it can help the search to explore various areas in the search space. Second, to effectively explore a new location around the current local optimum, we propose an adaptive multioperator perturbation procedure that relies

on three different but complementary rules to generate a new starting solution. Among these rules, two of them use a crossover operator. We devise a quality-diverse crossover operator and multisolution crossover operator to search for a new area in the solution space. Finally, we utilize the EMC as an acceptance criterion. The proposed components for ILS are presented in the following.

1) LS Procedure: The basic ILS utilizes the steepest descent algorithm in the LS phase. Although being simple to implement, it has the disadvantage of quickly becoming trapped in a local optimum, due to its greedy acceptance criterion. This often generates uncompetitive results, especially for hard, constrained optimization problems such as the DVRP. In addition, and more importantly, the type of neighborhood structures employed within the LS has a critical effect on algorithmic performance. Even when the neighborhood structure has been carefully selected for the instance at hand, its performance might change during the search process and is unlikely to be well suited to all problem instances. This is because not all problem instances have the same characteristics and the search landscape might change during the search.

To address this issue, our proposed approach employs an SVND algorithm as an LS procedure [43] within ILS. SVND integrates the distance between solutions within the objective function in order to visit distant valleys in the search space. SVND has three features that motivate us to integrate it with the ILS. First, it uses various neighborhood structures to avoid the basin of attraction points. As different neighborhood structures evolve different search paths, SVND can change the employed neighborhood structure to escape from the current local optimum point. Second, it uses several neighborhood structures, which might help the search to explore different regions and cope with the dynamic changes. Third, it uses the distance measure to move the search into other valleys.

SVND iteratively improves the given solution as follows. First, it randomly generates a sequence of neighborhood structures and then starts the search process with the first neighborhood in the sequence. When SVND finds a local optimum solution, it will try to escape from it by utilizing the following neighborhood structure in the generated sequence. SVND will stop if the current solution is a local optimum, and the last structure in the sequence has been called. Algorithm 2 gives the pseudocode of the SVND [43].

Initially, SVND sets, K_n, the number of neighborhood structures and the index of neighborhood structure (i) and constructs an initial starting solution (S_i) (lines 1–3). Next, at each SVND iteration (lines 4–12), it uses NS_i to generate a neighborhood solution S_1 (line 5). In SVND, the quality of the solution (S_1) is calculated as follows:

$$f(S_1) - \alpha * \text{dist}(s_1, s_2) \tag{5}$$

where α a is a parameter and $\text{dist}(s_1, s_2)$ is the distance between s_1 and s_2 as follows:

$$\text{dist}(s_1, s_2) = \frac{|E(s_1) \bigcup E(s_2)| - |E(s_1) \bigcap E(s_2)|}{|E(s_1) \bigcup E(s_2)|} \tag{6}$$

where E represent the number of edges. The distance function (dist) calculates dissimilarity between s_1 and s_2. $dist$ counts the number of customers assigned to different positions in s_1 and s_2. If the quality of S_n generated by NS_i is better than S_i ($f(S_n) - \alpha * \text{dist}(S_i, S_n) < f(S_i)$) (line 6), then replace S_i with S_n (line 7) and set $i = 1$ (line 8). If not, we increment i to run the next NS in the sequence (line 10). SVND will be terminated if the current local optimum of the K_n neighborhood structures cannot be improved any further.

Algorithm 2 SVND Algorithm

```
1  Let K_n be the total number of the neighborhood
   structures (NS);
2  Set i ← 1;
3  S_i ← GenerateInitialSolution();
4  while i < K_n do
5  |  S_n ← Generate a neighbor of S_i using NS_i;
6  |  if f(S_n) − α ∗ dist(S_i, S_n) < f(S_i) then
7  |  |  S_i = S_n;
8  |  |  i = 1;
9  |  else
10 |  |  i = i + 1;
11 |  end
12 end
13 Return the best solution;
```

This work uses various neighborhood structures within the SVND to effectively deal with DVRP instances. A neighborhood structure performs one change on a given solution to evolve a new solution. The operators that we used as neighborhood structures are as follows.

1) *NS1:* Randomly move one customer to a different feasible route.
2) *NS2:* Select two different customers at random and then change their routes.
3) *NS3:* Randomly pick one route and then reverse a segment of a tour between two chosen customers.
4) *NS4:* Select three customers at random and then exchange their routes.
5) *NS5:* Apply the 2-opt operator on all routes.
6) *NS6:* Pick two different routes at random and then exchange the first route segment with the first segment of the second route.
7) *NS7:* Pick two different routes at random and then exchange the first segment with the last segment.

2) Perturbation Procedure: When the LS procedure terminates (SVND in this work), the perturbation procedure takes this solution as an input and perturbs it to generate a new solution. The perturbation strength can either be weak or strong. Weak perturbation usually performs a small modification. This perturbation strength is preferred when the search should focus on the current area of the search, but it has the risk that the search may revisit previously visited solutions. Strong perturbation performs a larger modification. However, while providing a high degree of diversification, it has the disadvantage of making the algorithm behave the same as a random restart method. Neither weak nor strong perturbation is preferred over the other, and it is not known in advance which one is preferred for a given problem instance. Different perturbation strengths are required at different times in order

to generate a good starting solution. Consequently, there is a need for a perturbation procedure that can evolve a good starting solution based on the current search state and strike the necessary balance between weak and strong perturbation.

This work proposes a new adaptive perturbation procedure that utilizes three different rules (denoted as R1–R3) to provide a new starting solution based on the solutions stored in the population. R1 uses a crossover operator to mix high quality with high diversity solutions. The idea behind R1 is to generate a new starting solution that is not very far from the current best one. This can help the search to effectively explore all areas around the current search landscape before moving to a new area. R2 uses a multisolution crossover operator that combines the good features (high-quality set of routes) of all solutions stored in the population of solutions. The main idea behind R2 is to explore new and diverse locations in the search space by using all existing solutions to generate a highly diverse solution. R3 performs large neighborhood search using the ruin-recreate principle that tries to make a big jump in the search space. By using various rules, the search can effectively deal with huge and constrained search space, thus generating high-quality solutions and handling various instances with different characteristics.

In this work, when ILS calls the perturbation procedure to evolve a new starting solution, the perturbation procedure uses the solutions in the population to generate a new one by selecting one rule (R1, R2, or R3). We devise an adaptive strategy to pick the best performing rule using the ϵ-greedy strategy. Given the initial empirical performance means for all rules $(R_1(0), \ldots, R_3(0))$, $p_R(i)$, the probability of selecting Rth rule at iteration i is calculated as follows:

$$p_a(g+1) = \begin{cases} 1-\epsilon+\epsilon/R, & \text{if } \arg\max_{j=1,\ldots,3} R(g) \\ \epsilon/R, & \text{otherwise.} \end{cases} \tag{7}$$

Initially, we set a high value for ϵ and then decrease it during the search progress. This can help the selection to focus on exploration at the beginning of the search and then progressively move toward exploitation at a later state. The empirical mean of each rule R records the performance achieved by Rth rule when applied on the ith instance in the past. The proposed rules (R1–R3) are given in the following.

1) *R1:* Select two different solutions, one of high quality and one of high diversity, and then perform a crossover between them to construct a new solution. The new solution might be infeasible as there are either missed customers or duplicated customers. Missed customers are added to the best available routes that lead to minimal increase in the solution cost, whereas duplicated customers are removed from the routes with the highest costs.
2) *R2:* Generate a new starting solution by considering all solutions stored in population. First, identify the promising components for each solution. In DVRP, each solution comprises a set of vehicle routes and a promising solution refers to the best route in this solution. That is, the route that has the minimal traveling cost. Next, the extracted components (routes) from all solutions are merged to form a new solution. This might lead to the situation where there are either missed customers or duplicated customers. Missed customers are added to the best available routes that lead to minimal increase in the solution cost, whereas duplicated customers are removed from the routes with the highest costs.
3) *R3:* Randomly select one solution from the population and remove a subset of customers, *RC* and reassign them to different routes.

3) Acceptance Criterion Procedure: The acceptance criterion's role is to decide whether to accept or reject the generated solution. It compares the final solution's quality with the solution used as the input to the current LS. The solution with better quality is always accepted. To promote more diversity, and to make sure that the population contains high-quality and diverse solutions, worse solutions are also accepted to be included in the population according to the acceptance rule and the population update procedure. This work employs the EMC [44]. EMC uses the probability condition (Pro) to accept worse solutions as follows:

$$\text{Rand} < \exp(-\delta) \tag{8}$$

where Rand is a function that returns a random number between "0" and "1" and δ is the difference between the fitness of the current solution and the input solution. Pro will be decreased when δ is increased. In this work, the solution that has been accepted by EMC will be sent to the population update procedure (Section III-D) to decide whether to include it in the population according to the updating rule or not.

D. Population Update Procedure

To ensure that the current solutions are well-scattered over the search space, a new solution will be included in the population by considering its quality and diversity, that is, the population will contain both high-quality and diverse solutions. The quality (ψ) is calculated using (2), whereas the dissimilarity function represents the diversity (dist) measure [see (9)]. In the population update procedure, the diversity (dist) function is used as a matching procedure to find out how many customers in both solutions are not assigned to the same position and same route. If the value returned by dissimilarity function is zero, this means that the compared solutions are identical. This work treats these two measures [quality (ψ) and diversity (dist)] as two objectives according to the Pareto dominated principle. A solution a is Pareto dominated by a solution b if

$$\psi(b) < \psi(a) \text{ and } \sum_{i=1}^{\text{PS}} \text{dist}(b, c_i) \geq \sum_{i=1}^{\text{PS}} \text{dist}(a, d_i) \tag{9}$$

where c_i (resp. d_i) represents a solution in the population and PS is the total number of solutions. If the new solution's quality is better than the best one in the population, it will be added to the population. Then, for each solution in the population, calculate the number of solutions that this solution has dominated. The most dominant solution will be removed from the current population. If no solution is dominated, the population is updated according to first in, first out (FIFO).

TABLE I
DVRP BENCHMARK

Data set	Type	# Customers	Distribution
Christofides	c-series	50-199	Customers are either distributed in clusters around the serving area or uniformly distributed
Fisher	f-series	71-134	Most of the customers are distributed around the depot.
Taillard	tai-series	75-150	The distributions of customers are mixed (uniform and clusters)

TABLE II
PARAMETERS OF THE DVRP MODEL

#	Parameter	Value	Note
1-	D	750	The working day length
2-	n_{st}	25	The number of periods per day
3-	T_{co}	0.5	Cut-off time
4-	T_{ac}	0	The advanced commitment time

Using this idea ensures that the stored solutions are allocated in different locations in the search space while keeping both diverse and high-quality solutions.

IV. EXPERIMENTAL SETUP

In the following, we present the DVRP benchmark instances and the parameter settings.

A. DVRP Benchmark Instances

We use 21 DVRP instances introduced in [20] and further refined in [21] to test the performance of the proposed approach. These instances are derived from three well-known capacitated VRPs as follows: 13 instances from the Taillard benchmark, seven instances from the Christofides benchmark, and two instances from the Fisher benchmark (see [20], [21] for more details). The main differences between them are the size of each instance and customer distribution. The main characteristics of the utilized DVRP instances are presented in Table I.

B. Parameter Set-Up

This section involves two sections. The first one discusses the DVRP model parameters, whereas the second subsection presents the parameter settings of the proposed approach.

1) DVRP Parameter Settings: Table II presents the utilized parameter values of the DVRP model.

For the sake of comparability with the state-of-the-art methods, we have adopted the DVRP parameter values that were used in [21] and the compared methods. To study the effectiveness of these parameters on the algorithm's performance, various values for n_{st} and T_{co} have been tested, as shown in Tables III and IV, respectively. Both n_{st} and T_{co} have a direct impact on the DVRP model and performance of ILS. Eight different instances were selected to examine various parameter values by executing ILS for 31 times using different combinations of parameter values. Generally, the n_{st} parameter decides how often the event scheduler commits a new version of DVRP to the proposed approach. Consequently, having many n_{st} helps ILS to react faster to DVRP changes. However, due to the limited number of iterations, ILS might not have enough time to optimize the current DVRP instance effectively. This clearly justifies the results in Table III in which ILS performs the best when $n_{\text{st}} = 25$, which is consistent with state-of-the-art methods. T_{co} controls the degree of the problem dynamism. As can be seen from Table IV, when T_{co} increases, the quality of solution decreases. This is mainly because the problem becomes more difficult when there are too many changes that need to be accommodated within the current solving period.

TABLE III
RESULTS OF ILS USING DIFFERENT n_{st} VALUES

Instances	10	25	40	50
1	901.22	**870.22**	960.22	980.22
2	861.26	**830.26**	920.26	940.26
3	1103.93	**1072.93**	1162.93	1182.93
4	11827.08	**11796.08**	11886.08	11906.08
5	1727.26	**1696.26**	1786.26	1806.26
6	2097.26	**2066.26**	2156.26	2176.26
7	3353.75	**3322.75**	3412.75	3432.75
8	2907.46	**2876.46**	2966.46	2986.46

TABLE IV
RESULTS OF ILS USING DIFFERENT T_{co} VALUES

Instances	0.5	0.6	0.7	0.8
1	**870.22**	891.77	1030.35	1043.28
2	**830.26**	851.81	990.39	1003.32
3	**1072.93**	1094.48	1233.06	1245.99
4	**11796.08**	11817.63	11956.21	11969.14
5	**1696.26**	1717.81	1856.39	1869.32
6	**2066.26**	2087.81	2226.39	2239.32
7	**3322.75**	3344.3	3482.88	3495.81
8	**2876.46**	2898.01	3036.59	3049.52

TABLE V
PARAMETER SETTINGS

#	Parameter	Tested range	Suggested value
1-	MaxIter of ILS	100–2000	1000
2-	Population size, PS	2–50	10
3-	RC of R3	1%-50%	7% of the customers
4-	α	0-2	0.05

2) Proposed Approach Parameter Settings: The proposed approach has four tunable parameters. We have conducted preliminary tests to determine the value of these parameters, taking into consideration the quality of the generated solution as well as computational time. We have performed a preliminary investigation to set the parameter values. For this purpose, we randomly selected ten different instances to examine different parameter values. These are: c100b, c199, tai100d, f71, tai150b, tai75a, tai100c, f134, tai150a, and tai150c. The proposed approach has been executed 31 times using different parameter values combinations. To this end, we have utilized the Irace package (a parameter tuning tool) [45] to tune the parameters of the proposed approach. Table V summarizes the parameter values that were used in our computational experiments, along with the tested range for each parameter.

V. RESULTS AND COMPARISONS

The purpose of our simulations is twofold. 1) evaluate the effectiveness of the integrated components on the search performance (Section V-A) and 2) compare the computational results of the proposed approach with other methodologies reported in the scientific literature (Section V-B). Due to

TABLE VI
RESULTS OF ILS, NS1, NS2, NS3, NS4, NS5, NS6, AND NS7

Instances	ILS	NS1	NS2	NS3	NS4	NS5	NS6	NS7
	Average	Average	Average	Average	Average	Average	Average	Average
1	**870.22**	1213.39	1311.04	1061.32	1074.28	1124.72	10018.27	10203.05
2	**830.26**	1346.22	1365.75	1437.14	1511.03	1528.47	1571.23	1462.37
3	**1072.93**	1648.82	1564.54	1529.57	1546.92	1626.71	1538.46	1511.27
4	**11796.08**	16344.13	16542.23	16561.51	16263.35	16487.91	16136.7	16247.53
5	**1696.26**	1978.63	1912.46	1988.73	1991.58	2008.05	1875.37	1858.4
6	**2066.26**	2569.63	2512.94	2584.42	2536.1	2521.22	2545.83	2537.63
7	**3322.75**	3865.79	3871.74	3801.67	3865.44	3824.35	3840.98	3819.23
8	**2876.46**	3412.53	3417.71	3445.94	3409.83	3476.53	3510.36	3422.05

the stochastic nature of the proposed approach, we run each simulation for each instance for 31 independent runs using different random seeds.

A. Effectiveness Evaluation

This section first investigates the impact of the employed neighborhood structures on the performance of ILS. We have tested the traditional ILS using all neighborhood structures and each neighborhood structure separately (denoted as NS1–NS7). The compared variants (ILS, NS1, NS2, NS3, NS4, NS5, NS6, and NS7) use the same random seeds, starting solution, termination condition, the number of runs and computer resources. In this experimental test, eight different instances are randomly chosen to test ILS, NS1, NS2, NS3, NS4, NS5, NS6, and NS7. In particular, we selected two instances from the Fisher benchmark, three instances from the Taillard benchmark, and three instances from the Christofides benchmark. These instances were chosen to avoid being biased to one benchmark as well as avoiding over generalizing. Each variant is executed 31 times over the eight DVRP instances. The results are presented in Table VI. The best-achieved results are highlighted in a bold font. From the table, we can see that ILS produced the best results for all the tested DVRP instances. This clearly justifies the benefit of using all neighborhood structures on the search performance compared to each neighborhood structure separately. From the table, we can see that no one neighborhood structure can be considered the best across all instances. Indeed, each neighborhood structure excels on certain instances only. This finding is also consistent with the no free lunch theorem, which states that no single method with a unique configuration can excel over all problem instances. To summarize, the findings from the results clearly demonstrate the benefit of using several neighborhood structures within ILS, so they can complement each other and excel on all the tested instances.

We now analyze the effect of our proposed extensions to ILS, that is, the LS, perturbation procedure, and the population of solutions. We compare the proposed approach with the following variants.

1) *ILS:* The proposed approach that uses both procedures, that is, the proposed SVND LS, perturbation procedure, and the population of solutions.
2) *ILS1:* The general ILS approach that uses a simple steepest descent algorithm and a simplified perturbation procedure. In this variant, a solution is generated by moving a few customers to distinct routes. The number of customers is randomly set between 5% and 10% of the total customers in a given instance. In this variant, the simple descent algorithm terminates after 100 nonimproving iterations.
3) *ILS2:* Same as ILS1 but use the proposed SVND LS.
4) *ILS3:* Same as ILS1 but utilize the population of solutions.
5) *ILS4:* Same as ILS1 but utilize the proposed perturbation procedure.

TABLE VII
RESULTS OF ILS COMPARED TO OTHER VARIANTS

Instances	ILS	ILS1	ILS2	ILS3	ILS4
	Average	Average	Average	Average	Average
1	**870.22**	1022.16	1002.23	9750.24	9680.17
2	**830.26**	1113.16	1017.23	996.37	1012.25
3	**1072.93**	1514.05	1311.27	1300.79	1302.54
4	**11796.08**	15021.04	13764.45	13875.73	13441.13
5	**1696.26**	1854.71	1789.31	1745.54	1772.36
6	**2066.26**	2436.81	2201.16	2292.26	2211.41
7	**3322.75**	3731.25	3507.03	3512.91	3576.12
8	**2876.46**	3201.75	3076.54	3052.25	3010.09

TABLE VIII
p-VALUE OF ILS VERSUS ILS1, ILS2, ILS3, AND ILS4

ILS vs.	ILS1	ILS2	ILS3	ILS4
Instances	$\mathcal{P}$-value	$\mathcal{P}$-value	$\mathcal{P}$-value	$\mathcal{P}$-value
1	+	+	+	+
2	+	+	+	+
3	+	+	+	+
4	+	+	+	+
5	+	+	+	+
6	+	+	+	+
7	+	+	+	+
8	+	+	+	+

All algorithms (ILS, ILS1, ILS2, ILS3, and ILS4) use the same random seeds, initial solution, stopping condition, the number of runs, and computer resources. Each variant is executed 31 times over the eight DVRP instances. The obtained results of all variants are tabulated in Table VII. ILS outperformed other variants on all instances. This comparison shows that the proposed components have positive impacts on search performance.

We further verify the performance of ILS using the Wilcoxon test with a 0.05 significance level. Table VIII shows the $\mathcal{P}$-values of ILS versus ILS1, ILS2, ILS3, and ILS4. In this table, we use "+"to indicate that ILS is statistically better than the competing variant ($\mathcal{P}$-value < 0.05), "-"if ILS was outperformed by the compared variant ($\mathcal{P}$-value > 0.05), and "$\sim$ "if both have the same performance ($\mathcal{P}$-value $= 0.05$).

Based on the $\mathcal{P}$-values shown in Table VIII, for all the DVRP instances, the results of ILS are statistically significant compared to the competing variants (ILS1, ILS2, ILS3, and ILS4). The better performance achieved by ILS can be due to the following features: 1) the ability of SVND to explore diverse areas of the search space by using several

TABLE IX
BEST RESULTS OF ILS AND OTHER METHODS FOR THE 21 DVRP INSTANCES

Instances	ILS			ANT	GRASP	GA	TS	GA-HH	PSO	VNS	GA-2	PSO-2	ContDVRP	E-ACO	MBO	BSO	M-VRPDR	BKS
	Best	Δ (%)	*Rank*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*
c50	**523.11**	-0.29	1	631.3	696.92	570.89	603.57	597.74	1273.5	599.53	566.01	562.7	551.34	607.21	524.61	597.18	524.61	524.61
c75	**860.24**	-0.72	1	1009.38	1066.59	981.57	981.51	979.25	1646.36	981.64	944.46	874.08	886.42	924.71	866.45	938.79	866.45	835.26
c100b	**816.17**	-0.41	1	944.23	978.39	881.92	891.42	956.67	944.46	866.71	869.41	819.56	819.56	869.22	873.1	900.48	873.1	865.94
c100	**820.01**	-0.11	1	973.26	1080.33	961.1	997.15	975.17	988.72	1022.92	943.89	882.96	873.77	973.4	820.92	912.83	820.92	826.14
c120	**1051.43**	-0.5	1	1416.45	1546.5	1303.59	1331.8	1245.94	1276.88	1285.21	1288.66	1066.15	1056.7	1108.15	1199.64	1173.1	1199.64	1042.12
c150	**1081.92**	-0.17	1	1345.73	1468.36	1348.88	1318.22	1342.91	1371.08	1334.73	1273.5	1147.5	1097.27	1378.63	1083.79	1254.47	1083.79	1028.42
c199	**1361.37**	-0.95	1	1771.04	1774.33	1654.51	1750.09	1689.52	1640.4	1679.65	1646.36	1434.7	1374.47	1561.12	1376.26	1631.23	1376.26	1291.29
f71	260.13	0.16	2	311.18	359.16	301.79	280.23	288	279.52	304.32	288.3	270.35	270.2	**259.71**	275.51	270.28	275.51	237
f134	**11710.86**	-0.02	1	15135.51	15433.84	15528.81	15717.9	14801.55	15875	15680.05	14871.4	11773.74	11713.2	-	11817	14831.29	11817	1162
tai75a	**1618.61**	-2.38	1	1843.08	1911.48	1782.91	1778.52	1769.67	1816.07	1806.81	1744.78	1767.64	1691.95	1690.91	1658.13	1713.46	1658.13	1618.36
tai75b	1357.52	0.48	2	1535.43	1582.24	1464.56	1461.37	1450.44	1447.39	1480.7	1441.35	1366.8	1356.5	1509.56	1351.06	1426.93	**1351.06**	1344.62
tai75c	**1325.96**	-0.26	1	1574.98	1596.17	1440.54	1406.27	1685.15	1481.35	1621.03	1433.73	1427.76	1424.91	1329.42	1388.13	1392.96	1388.13	1291.01
tai75d	**1379.19**	-1.15	1	1472.35	1545.21	1399.83	1430.83	1432.87	1414.28	1446.5	1408.48	1404.75	1403.85	1409.14	1395.3	1503.79	1395.3	1365.42
tai100a	2123.9	0.05	2	2375.92	2427.07	2232.71	2208.85	2227.51	2249.84	2250.5	2181.31	2196.91	2147.07	2281.7	2122.92	2237.9	**2122.92**	2041.34
tai100b	1998.16	0.36	2	2283.97	2302.95	2147.7	2219.28	2183.35	2238.42	2169.1	2119.03	2060.46	2041.96	2255.83	1990.99	2068.12	**1990.99**	1939.9
tai100c	**1416.97**	-0.21	1	1562.3	1599.19	1541.28	1515.1	1656.92	1532.56	1490.58	1504.63	1476.24	1446.98	1442.45	1419.89	1479.48	1419.89	1406.2
tai100d	**1574.93**	-0.41	1	2008.13	1973.03	1834.6	1881.91	1834.4	1955.06	1969.94	1793.64	1676.1	1658.48	1581.36	1651.28	1855.26	1651.28	1580.46
tai150a	**3267.21**	-0.41	1	3644.78	3787.53	3328.85	3488.02	3346.08	3400.33	3479.44	3280.79	3476.84	3396.49	3307.63	3343.54	3391.39	3343.54	3055.23
tai150b	**2850.08**	-0.86	1	3166.88	3313.03	2933.4	3109.23	2874.83	3013.99	2934.86	2885.94	2978.3	2931.16	3128	2957.16	2899.56	2957.16	2727.03
tai150c	**2426.85**	-0.17	1	2811.48	3110.1	2612.68	2666.28	2583.04	2714.34	2674.29	2593.78	2523.75	2523.53	2583.36	2430.91	2596.67	2430.91	2358.66
tai150d	**2731.17**	-0.41	1	3058.87	3159.21	2950.61	2950.83	3084.52	3025.43	2954.64	2911.47	2958.75	2929.91	2808.99	2742.35	3073.54	2742.35	2645.39

neighborhood structures where each one navigates through the search space in a slightly different way; 2) ILS integrates a population of solutions with a perturbation procedure to generate a new starting solution; and 3) ILS employs an adaptive multioperator perturbation procedure that relies on three different but complementary rules to develop a new starting solution.

B. Comparisons With the State-of-the-Art Methodologies

This section compares ILS against the best methodologies reported in the scientific literature. We compare against the best and average results. The algorithms we compare against are as follows.

1) *ANT:* Ant colony optimization algorithm [21].
2) *GRASP:* Greedy randomized adaptive search procedure [21].
3) *GA:* Genetic algorithms [22].
4) *TS:* Tabu search algorithm [22].
5) *GA-HH:* Genetic algorithm-based hyperheuristic [46].
6) *PSO:* Particle swarm optimization algorithm [30].
7) *VNS:* Variable neighborhood search algorithm [26].
8) *PSO-2:* Particle swarm optimization algorithm [31].
9) *GA-2:* Genetic algorithm [25].
10) *M-VRPDR:* Memetic approach [27].
11) *ACO-CD:* Diversity-based ant colony optimization [23].
12) *ContDVRP:* Metaheuristic approach [32].
13) *E-ACO:* Enhanced ant colony optimization [24].
14) *MBO:* Monarch butterfly algorithm [28].
15) *BSO:* Brain storm optimization algorithm [29].

The best results of ILS compared to ANT, GRASP, GA, TS, GA-HH, PSO, VNS, GA-2, PSO-2, ContDVRP, E-ACO, MBO, BSO, and M-VRPDR are presented in Table IX. In the table, we present the best known solutions (BKSs) for static VRP.[1] We report the instance ranking and the percentage deviation $\Delta(\%)$, $\Delta(\%) = ((\text{best}_1 - \text{best}_2)/\text{best}_2)*100$, where best_1 is the best result of ILS and best_2 is the best result reported in the scientific literature. "−" indicates no results reported by the corresponding algorithm. The best-achieved results are highlighted in bold font. As shown in Table IX, ILS produced the new best results for 17 of the 21 instances. ILS obtained the second-best results for the other instances. One can remark that the percentage deviations $[\Delta(\%)]$ from the best results for these instances are relatively small (0.36% and 0.48%), which indicates that ILS is very competitive. If we consider the individual comparison between ILS and the other algorithms, ILS outperformed ANT, GRASP, GA, TS, GA-HH, PSO, VNS, GA-2 PSO-2, ContDVRP, E-ACO, MBO, BSO, and M-VRPDR 21, 21, 21, 21, 21, 21, 21, 21, 21, 20, 21, 21, and 18 out of 21 instances, respectively.

Table X compares the average results of ILS with GA-HH, GRASP, ANT, TS, GA, PSO, VNS, GA-2, PSO-2, ACO-CD, ContDVRP, E-ACO, and MVRPDR. "−" indicates no results reported by the corresponding algorithm. Please note that, in this comparison, we only included those who reported the average results. On all tested instances, ILS achieved better average results compared to GA-HH, GRASP, ANT, TS, GA, PSO, VNS, GA-2, PSO-2, ACO-CD, ContDVRP, E-ACO, and MVRPDR.

The performances of ILS, ANT, GRASP, GA, TS, GA-HH, PSO, VNS, GA-2, PSO-2, ContDVRP, MBO, BSO ACO-CD, and M-VRPDR are compared statistically using multiple Friedman statistical tests [47]. We first conducted Friedman and Iman–Davenport statistical test using a 0.05 significance level. Both Friedman and Iman–Davenport statistical test $\mathcal{P}$-values are less than the critical level of 0.05. Thus, Holm and Hochberg *post hoc* statistical tests are conducted to compare the difference between these methods [47]. Table XI summarizes the average ranking produced by the Friedman statistical test of ILS, ANT, GRASP, GA, TS, GA-HH, PSO, VNS, GA-2, PSO-2, ContDVRP, MBO, BSO ACO-CD, and M-VRPDR, which indicates that ILS ranked first followed by M-VRPDR, MBO, ContDVRP, PSO-2, GA-2, GA, ACO-CD, GA-HH, TS, VNS, PSO, ANT, and GRASP. Therefore, the Holm and Hochberg statistical tests will be performed using ILS as the controlling algorithm.

Table XII gives the $\mathcal{P}$-values of the Holm and Hochberg statistical tests. As shown in Table XII, the results of ILS are statistically better than that of PSO-2, GA-2, GA, ACO-CD, GA-HH, TS, VNS, PSO, ANT, and GRASP (10 out of 13 methods) with a critical level of 0.05 ($\mathcal{P}$-value $<$ 0.05). If we use 0.010 critical level ($\mathcal{P}$-value $<$ 0.010), the results of ILS are statistically better than that of MBO and ContDVRP. If we consider the unadjusted $\mathcal{P}$-values, the results of ILS are statistically better than all the compared algorithms. The $\mathcal{P}$-value in Table XII shows that ILS is not statistically better than M-VRPDR; nevertheless, ILS obtained better best

[1] http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

TABLE X
AVERAGE RESULTS OF ILS AND OTHER METHODS FOR THE 21 DVRP INSTANCES

Instances	**ILS**	**ANT**	**GRASP**	**GA**	***TS***	**GA-HH**	**PSO**	**VNS**	**GA-2**	**PSO-2**	**ACO-CD**	**ContDVRP**	**E-ACO**	**M-VRPDR**
	Average	*Average*	*Average*	*Average*	*Average*	*Average*	*Average*	*Average*	*Average*	*Average*	*Average*	*Average*	*Average*	*Average*
c50	**544.12**	681.86	719.56	593.42	627.9	632.73	632.38	653.84	597.34	581.46	593.32	580.56	647.21	548.1
c75	**870.22**	1042.39	1079.16	1013.45	1013.82	1019.01	1031.76	1040	990.78	905.95	944.18	901.05	1045.44	885
c100	**830.26**	1066.16	1119.06	987.59	1047.6	1003.92	1051.5	1087.18	988.15	930.95	963.8	920.69	1044.96	864.03
c100b	**838.55**	1023.6	1022.12	900.94	932.14	1020.02	964.47	942.81	904.03	844.9	948.56	844.56	950.17	913.81
c120	**1072.93**	1525.15	1643.15	1390.58	1468.12	1372.45	1457.22	1469.24	1399.4	1085.46	1235.3	1116.08	1197.68	1295.31
c150	**1114.06**	1455.5	1501.35	1386.93	1401.06	1413.04	1470.95	1441.37	1359.25	1195.95	1268.16	1127.04	1472.4	1142.24
c199	**1397.82**	1844.82	1898.2	1758.51	1783.43	1746.98	1818.55	1769.95	1700.54	1503.94	1566.37	1418.71	1836.86	1466.3
f71	**271.27**	348.69	376.66	309.94	306.33	299.59	312.35	325.18	309.49	290.62	289.24	275.27	297.08	297.61
f134	**11796.08**	16083.56	16458.47	15986.84	16582.04	14921.24	16645.89	16522.18	15789.8	12038.02	15827.69	11810.04	-	12299.59
tai75a	**1696.26**	1945.2	2005.44	1856.66	1883.47	1859.17	1935.28	1954.25	1823.71	1825.87	1964.59	1782.79	1983.92	1705.76
tai75b	**1370.94**	1704.06	1758.88	1527.77	1587.72	1502.07	1484.73	1560.71	1546.18	1419.66	1527.33	1401.38	1647.78	1378.47
tai75c	**1398.67**	1653.58	1674.37	1501.91	1527.8	1779.14	1664.4	1746.07	1502.56	1487.39	1617.84	1486.78	1470.6	1427.74
tai75d	**1401.22**	1529	1588.73	1422.27	1453.56	1445.84	1493.47	1541.98	1434.56	1442.45	1522	1428.91	1661.73	1414.93
tai100a	**2161.06**	2428.38	2510.29	2295.61	2310.37	2309.98	2370.58	2462.5	2290.95	2261.66	2213.04	2226.48	2550.64	2168.87
tai100b	**2066.26**	2347.9	2512.27	2215.39	2330.52	2221.37	2385.54	2319.72	2212.58	2151.73	2391.59	2125.19	2500.72	2097.86
tai100c	**1455.08**	1655.91	1704.4	1622.66	1604.18	1756.2	1627.32	1557.81	1589.76	1512.13	1584.2	1485.47	1743.07	1460.72
tai100d	**1671.66**	2060.72	2087.55	1912.43	2026.76	2029.37	2123.9	2100.38	1916.03	1746.44	2275.15	1718.18	1843.82	1710.44
tai150a	**3322.75**	3840.18	3899.16	3501.83	3598.69	3487.85	3612.79	3680.35	3449.32	3777.98	3473.58	3526.1	3684.03	3354.09
tai150b	**2973.08**	3327.47	3485.79	3115.39	3215.32	3068.75	3232.11	3089.57	3073.58	3120.09	3347.3	3034.65	3439.38	2980.33
tai150c	**2495.36**	3016.14	3219.27	2743.55	2913.67	2731.05	2875.93	2928.77	2759.96	2678.16	2851.53	2642.82	2729.15	2558.99
tai150d	**2876.46**	3203.75	3298.76	3045.16	3111.43	3251.97	3347.6	3147.38	3010.34	3141.63	3191.28	3023.48	3186.08	2903.68

TABLE XI
FRIEDMAN TEST RESULTS

#	Algorithm	Ranking
1	ILS	1
2	M-VRPDR	2.8
3	MBO	3.9
4	ContVRP	4
5	PSO-2	5.4
6	GA-2	6.7
7	GA	7.2
8	ACO-CD	8.1
9	GA-HH	8.7
10	TS	9.4
11	VNS	10.9
12	PSO	11
13	ANT	12
14	GRASP	13.4

TABLE XII
HOLM AND HOCHBERG ADJUSTED p-VALUES

i	algorithm	unadjusted p	p_{Holm}	$p_{Hochberg}$
1	GRASP	0	0	0
2	ANT	0	0	0
3	PSO	0	0	0
4	VNS	0	0	0
5	TS	0	0	0
6	GA-HH	0	0	0
7	ACO-CD	0	0	0
8	GA	0.000001	0.000008	0.000008
9	GA-2	0.000008	0.00004	0.00004
10	PSO-2	0.000603	0.002411	0.002411
11	ContVRP	0.020137	0.06041	0.044403
12	MBO	0.022201	0.06041	0.044403
13	M-VRPDR	0.150282	0.150282	0.150282

results for 18 instances and better average results for 21 out of 21 tested instances compared to ContDVRP.

In Table XIII, we report the computational time (seconds) for ILS, ANT, GRASP, GA, TS, PSO-2, GA-HH, MBO, and ContDVRP. In this comparison, PSO-2 and ContDVRP employ parallel optimization processes. Note that PSO, GA-2, and VNS reported the normalized computational times only and therefore are not considered in this comparison. The results in Table XIII demonstrate that the computational time of ILS is relatively small compared to other methods. Please note that the computational resources comparison can only be indicative. This is mainly due to the fact that different researchers often use different types of computer resources. In addition, most of the researchers have not mentioned the used operating systems, the programming language, the programmer's skill level, and type of compilers.

TABLE XIII
COMPUTATIONAL TIME (SECONDS) OF ILS COMPARED TO OTHER ALGORITHMS

#	Algorithm	Time (s)
1	ILS	**72**
2	GRASP	1500
3	ANT	1500
4	TS	750
5	GA-HH	750
6	GA	750
7	PSO-2	75
8	MBO	750
9	ContDVRP	75

Even though ILS did not produce the best solutions for all DVRP instances, it was the best for 17 out of 21 tested instances, and when it was outperformed, it ranked second. Furthermore, the computational time of ILS is relatively small, which demonstrates that the proposed ILS is an efficient methodology for DVRP.

VI. CONCLUSION

This article proposed a population-based approach for the DVRP. The proposed approach combines evolutionary operators and a population of solutions with ILS algorithm in an adaptive manner. It uses an SVND algorithm that incorporates various neighborhood structures as an LS algorithm. A new perturbation procedure that uses a population of solutions and three different solution generation rules was proposed. A quality-and-diversity population updating strategy is devised to handle the dynamic changes and to promote diversity in the search. We verified the proposed approach's effectiveness using the 21 DVRP instances that other researchers in the literature have used. The proposed approach produced excellent results compared to its counterparts. Compared with the state-of-the-art methods, the proposed approach outperformed them on several instances using shorter computational times. Future work intends to integrate the proposed approach with genetic algorithms and test it on the VRP with loading constraints.

REFERENCES

[1] J. S. Tan, S. L. Goh, G. Kendall, and N. R. Sabar, "A survey of the state-of-the-art of optimisation methodologies in school timetabling problems," *Expert Syst. Appl.*, vol. 165, Mar. 2021, Art. no. 113943.

[2] S. L. Goh, G. Kendall, and N. R. Sabar, "Monte Carlo tree search in finding feasible solutions for course timetabling problem," *Int. J. Adv. Sci., Eng. Inf. Technol.*, vol. 9, no. 6, p. 1936, Dec. 2019.

[3] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Manage. Sci.*, vol. 6, no. 1, pp. 80–91, 1959.

[4] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "Heuristics for multi-attribute vehicle routing problems: A survey and synthesis," *Eur. J. Oper. Res.*, vol. 231, no. 1, pp. 1–21, 2013.

[5] K. Woiceshyn, Z. Kashino, G. Nejat, and B. Benhabib, "Vehicle routing for resource management in time-phased deployment of sensor networks," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 2, pp. 716–728, Apr. 2019.

[6] W. Cook, *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton, NJ, USA: Princeton Univ. Press, 2012.

[7] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. Hoboken, NJ, USA: Wiley, 1990.

[8] P. Toth and D. Vigo, *Vehicle Routing: Problems, Methods, and Applications*, vol. 18. Philadelphia, PA, USA: SIAM, 2014.

[9] O. Bräysy and M. Gendreau, "Vehicle routing problem with time windows, Part I: Route construction and local search algorithms," *Transp. Sci.*, vol. 39, no. 1, pp. 104–118, 2005.

[10] A. Mingozzi, R. Roberti, and P. Toth, "An exact algorithm for the multitrip vehicle routing problem," *INFORMS J. Comput.*, vol. 25, no. 2, pp. 193–207, May 2013.

[11] C. Groër, B. Golden, and E. Wasil, "A parallel algorithm for the vehicle routing problem," *INFORMS J. Comput.*, vol. 23, no. 2, pp. 315–330, May 2011.

[12] I. H. Osman, "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," *Ann. Oper. Res.*, vol. 41, no. 4, pp. 421–451, 1993.

[13] M. Gendreau, A. Hertz, and G. Laporte, "A tabu search heuristic for the vehicle routing problem," *Manage. Sci.*, vol. 40, no. 10, pp. 1276–1290, 1994.

[14] É. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin, "A tabu search heuristic for the vehicle routing problem with soft time windows," *Transp. Sci.*, vol. 31, no. 2, pp. 170–186, 1997.

[15] O. Bräysy, "A reactive variable neighborhood search for the vehicle-routing problem with time windows," *INFORMS J. Comput.*, vol. 15, no. 4, pp. 347–368, Nov. 2003.

[16] A. Imran, S. Salhi, and N. A. Wassan, "A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem," *Eur. J. Oper. Res.*, vol. 197, no. 2, pp. 509–518, 2009.

[17] B. M. Baker and M. A. Ayechew, "A genetic algorithm for the vehicle routing problem," *Comput. Oper. Res.*, vol. 30, no. 5, pp. 787–800, 2003.

[18] L. M. Gambardella, E. Taillard, and G. Agazzi, "MACS-VRPTW: A multiple colony system for vehicle routing problems with time windows," in *New Ideas in Optimization*. Princeton, NJ, USA: Citeseer, 1999.

[19] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *Eur. J. Oper. Res.*, vol. 225, no. 1, pp. 1–11, 2013.

[20] P. Kilby, P. Prosser, and P. Shaw, "Dynamic VRPs: A study of scenarios," School Comput. Sci., Univ. Strathclyde, Glasgow, Scotland, Tech. Rep. APES-06-1998, 1998, pp. 1–11.

[21] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, and A. V. Donati, "Ant colony system for a dynamic vehicle routing problem," *J. Combinat. Optim.*, vol. 10, no. 4, pp. 327–343, 2005.

[22] F. T. Hanshar and B. M. Ombuki-Berman, "Dynamic vehicle routing using genetic algorithms," *Appl. Intell.*, vol. 27, no. 1, pp. 89–99, 2007.

[23] X. Xiang, J. Qiu, J. Xiao, and X. Zhang, "Demand coverage diversity based ant colony optimization for dynamic vehicle routing problems," *Eng. Appl. Artif. Intell.*, vol. 91, May 2020, Art. no. 103582.

[24] H. Xu, P. Pu, and F. Duan, "Dynamic vehicle routing problems with enhanced ant colony optimization," *Discrete Dyn. Nature Soc.*, vol. 2018, pp. 1–13, Feb. 2018.

[25] A. M. F. M. Abdallah, D. L. Essam, and R. A. Sarker, "On solving periodic re-optimization dynamic vehicle routing problems," *Appl. Soft Comput.*, vol. 55, pp. 1–12, Jun. 2017.

[26] B. Sarasola, M. R. Khouadjia, E. Alba, L. Jourdan, and E.-G. Talbi, "Flexible variable neighborhood search in dynamic vehicle routing," in *Applications of Evolutionary Computation*. Turin, Italy: Springer, 2011, pp. 344–353.

[27] J. Mańdziuk and A. Żychowski, "A memetic approach to vehicle routing problem with dynamic requests," *Appl. Soft Comput.*, vol. 48, pp. 522–534, Nov. 2016.

[28] S. Chen, R. Chen, and J. Gao, "A monarch butterfly optimization for the dynamic vehicle routing problem," *Algorithms*, vol. 10, no. 3, p. 107, Sep. 2017.

[29] M. Liu, Y. Shen, and Y. Shi, "A hybrid brain storm optimization algorithm for dynamic vehicle routing problem," in *Proc. Int. Conf. Swarm Intell.* Belgrade, Serbia: Springer, 2020, pp. 251–258.

[30] M. R. Khouadjia, B. Sarasola, E. Alba, L. Jourdan, and E.-G. Talbi, "A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests," *Appl. Soft Comput.*, vol. 12, pp. 1426–1439, 2012.

[31] M. Okulewicz and J. Mańdziuk, "The impact of particular components of the PSO-based algorithm solving the dynamic vehicle routing problem," *Appl. Soft Comput.*, vol. 58, pp. 586–604, Sep. 2017.

[32] M. Okulewicz and J. Mańdziuk, "A metaheuristic approach to solve dynamic vehicle routing problem in continuous search space," *Swarm Evol. Comput.*, vol. 48, pp. 44–61, Aug. 2019.

[33] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "A dynamic multi-armed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 217–228, Feb. 2015.

[34] N. R. Sabar, A. Bhaskar, E. Chung, A. Turky, and A. Song, "A self-adaptive evolutionary algorithm for dynamic vehicle routing problems with traffic congestion," *Swarm Evol. Comput.*, vol. 44, pp. 1018–1027, Feb. 2019.

[35] Q. Wu, M. Zhou, Q. Zhu, Y. Xia, and J. Wen, "MOELS: Multiobjective evolutionary list scheduling for cloud workflows," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 1, pp. 166–176, Jan. 2020.

[36] H. Jia *et al.*, "Multiobjective bike repositioning in bike-sharing systems via a modified artificial bee colony algorithm," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 2, pp. 909–920, Apr. 2020.

[37] N. R. Sabar and M. Ayob, "Examination timetabling using scatter search hyper-heuristic," in *Proc. 2nd Conf. Data Mining Optim.*, Oct. 2009, pp. 127–131.

[38] J. S. Tan, S. L. Goh, S. Sura, G. Kendall, and N. R. Sabar, "Hybrid particle swarm optimization with particle elimination for the high school timetabling problem," *Evol. Intell.*, 2020, doi: 10.1007/s12065-020-00473-x.

[39] S. L. Goh, G. Kendall, N. R. Sabar, and S. Abdullah, "An effective hybrid local search approach for the post enrolment course timetabling problem," *Opsearch*, vol. 57, no. 4, pp. 1131–1163, Dec. 2020.

[40] N. R. Sabar and G. Kendall, "Population based Monte Carlo tree search hyper-heuristic for combinatorial optimization problems," *Inf. Sci.*, vol. 314, pp. 225–239, Sep. 2015.

[41] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search: Framework and applications," in *Handbook of Metaheuristics* (International Series in Operations Research & Management Science), M. Gendreau and J.-Y. Potvin, Eds. Springer, 2019, pp. 363–397.

[42] N. R. Sabar and G. Kendall, "An iterated local search with multiple perturbation operators and time varying perturbation strength for the aircraft landing problem," *Omega*, vol. 56, pp. 88–98, Oct. 2015.

[43] P. Hansen, N. Mladenović, and J. A. M. Pérez, "Variable neighbourhood search: Methods and applications," *Ann. Oper. Res.*, vol. 175, no. 1, pp. 367–407, 2010.

[44] M. Ayob and G. Kendall, "A Monte Carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine," in *Proc. Int. Conf. Intell. Technol. (InTech)*, vol. 3, 2003, pp. 132–141.

[45] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, T. Stützle, and M. Birattari, "The irace package: Iterated racing for automatic algorithm configuration," *Oper. Res. Perspect.*, vol. 3, pp. 43–58, Jan. 2016.

[46] P. Garrido and M. C. Riff, "DVRP: A hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic," *J. Heuristics*, vol. 16, no. 6, pp. 795–834, 2010.

[47] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Inf. Sci.*, vol. 180, no. 10, pp. 2044–2064, 2010.