A Heuristic Initialized Stochastic Memetic Algorithm for MDPVRP with Interdependent Depot Operations

Abdus Salam Azad, Md. Monirul Islam, Member, IEEE, and Saikat Chakraborty

***Abstract*—The Vehicle Routing Problem (VRP) is a widely studied combinatorial optimization problem. We introduce a variant of the Multi-Depot and Periodic Vehicle Routing Problem (MDPVRP) and propose a heuristic initialized stochastic memetic algorithm (HISMA) to solve it. The main challenge in designing such an algorithm for a large combinatorial optimization problem is to avoid premature convergence by maintaining a balance between exploration and exploitation of the search space. We employ intelligent initialization and stochastic learning to address this challenge. The intelligent initialization technique constructs a population by a mix of random and heuristic generated solutions. The stochastic learning enhances the solutions' quality selectively using simulated annealing with a set of random and heuristic operators. The hybridization of randomness and greediness in the initialization and learning process helps to maintain the balance between exploration and exploitation. Our proposed algorithm has been tested extensively on the existing benchmark problems and outperformed the baseline algorithms by a large margin. We further compared our results with that of the state-of-the-art algorithms working under MDPVRP formulation and found a significant improvement over their results.**

***Index Terms*—Memetic Algorithm, Combinatorial Optimization, Multi-Depot and Periodic Vehicle Routing Problem, Simulated Annealing, and Learning**

I. Introduction

THE Vehicle Routing Problem (VRP) proposed by Dantzig and Ramser in 1959 under the title "The Truck Dispatching Problem" [1] is one of the most important and studied combinatorial optimization problems in the literature. The VRP can be defined as designing the least-cost delivery routes from a depot to a set of geographically scattered customers, subject to some constraints [2]. It plays a central role in the fields of physical distribution and logistics such as school bus routing, meter readings, supply chain management, production planning, and telecommunications. Using computerized algorithms for distribution process has been shown to produce savings from 5% to 20% in the global transportation costs [3]. Thus designing an algorithm which produces the low-cost vehicle routes has a great impact on the transportation system.

Over the last several decades, quite a few VRP variants such as Capacitated VRP (CVRP), VRP with Time Windows (VRPTW), Periodic VRP (PVRP), Multi-Depot VRP (MDVRP), and Multi-Depot and Periodic VRP (MDPVRP) have been formulated. It is worth mentioning that not all variants received the same attention. For example, CVRP and VRPTW have been widely studied in the literature (see [3] and [4]). PVRP, which requires designing routes for a number of days and MDVRP, which deals with multiple depots, have also received a significant amount of attention (see [5] and [6]). However, only a few works deal with MDPVRP which is a generalization of both PVRP and MDVRP. Among them, MDPVRP has been solved mainly by meta-heuristic approaches (e.g., tabu search [7] [8] and population based methods [9] [10]). Only a few studies (e.g., [10]) adopt a memetic algorithm to solve MDPVRP, in spite of its success in solving different combinatorial optimization problems [11] [12]. A challenging issue in such an approach is to make a balance between exploration and exploitation to avoid premature convergence.

In this paper, we introduce a new variant of VRP that extends MDPVRP with interdependent depot operations (MDPVRPID) and propose a memetic algorithm to solve it. The new variant, by allowing interdependent depot operations, can reduce solution cost at the expense of increasing the search space. To find good solutions from the large search space by maintaining a balance between exploration and exploitation, our proposed heuristic initialized stochastic memetic algorithm (HISMA) employs intelligent initialization and stochastic learning. The essence of such an initialization and learning is the emphasis on both greediness and randomness in their working procedure. The hybridization of randomness and greediness helps to maintain the balance between exploration and exploitation.

The rest of this paper is organized as follows. We first discuss some previous works related to our topic in Section II. In Section III, we present the formulation of MDPVRPID. We discuss our HISMA in detail in Section IV. The performance of HISMA and its comparison with other algorithms are presented in Section V. At last, we conclude with an outlook to implied future works in Section VI.

II. Literature Review

In this section, we first provide a brief review on some well known VRP variants. Then, we describe existing memetic approaches used for solving such variants. Finally, we review the existing approaches that solve MDPVRP, the closest variant of the proposed MDPVRPID.

VRP has been extended in different ways to address various problems arising in the fields of logistics. CVRP, the simplest

A. S. Azad and M. M. Islam are with the Bangladesh University of Engineering and Technology, Dhaka 1000, Bangladesh (e-mail: azadsalam@cse.buet.ac.bd and mdmonirulislam@cse.buet.ac.bd).

S. Chakraborty is with University of Virginia, Charlottesville, VA 22903, USA (e-mail: saikat.chakraborty@virginia.edu).

extension of VRP, deals with a homogeneous vehicle fleet under a single depot where each member of the fleet is constrained by a uniform capacity. Another well-known variant VRPTW, deals with time windows defined by the customers within which they must be visited. CVRP can be extended to VRP with Pick-Up and Delivery (VRPPD) in which the vehicles can also collect goods from the customers along with their usual deliveries. The VRPPD can further be extended to VRPPD with Time Windows (VRPPDTW) to incorporate time window constraints. The CVRP can also be extended to prepare a plan that spans over multiple days, giving rise to PVRP. Another variant MDVRP extends CVRP by incorporating multiple depots, each of which can serve a customer by its own vehicle fleet. Finally, MDPVRP combines both PVRP and MDVRP to design routes originating from multiple depots, which cover a planning horizon of multiple number of days.

Memetic algorithms have been successfully employed to solve VRPs and other related problems. The works presented in [13] and [14] solve multiobjective capacitated arc routing problems using memetic algorithm with decomposition. The authors of [15] introduced an operator to explore composite VRP neighborhoods and incorporated it under a memetic algorithm to solve MDVRP. In [16], a local search was incorporated in cooperative coevolution to solve MDVRP. A memetic algorithm with greedy local search was employed in [17] to solve a variant of PVRP with profit. In [18], the authors modeled a variant of VRPPDTW with simultaneous pickup and delivery as a multi-objective problem and proposed a decomposition based multi-objective memetic algorithm to solve it.

A challenging aspect in designing a memetic algorithm is to maintain balance between exploration and exploitation. Hybrid genetic search with adaptive diversity control (HGSADC) [10] is one such algorithm that considers both exploration and exploitation to solve MDPVRP. It improves each offspring by a local search with a fix probability to exploit the search space. To facilitate exploration, HGSADC not only incorporates a diversity term in the fitness measure but also replaces the worst solutions with new ones when its evolutionary progress stalls. The authors of [19] adopted a distance measure to control the population diversity for ensuring exploration. More information about memetic algorithms can be found in [20].

Among the few works that solve MDPVRP, most are meta-heuristic based. To the best of our knowledge, [21] is the only exact method to solve MDPVRP. [7] proposed a tabu search based meta-heuristic to solve a single MDPVRP instance consisting of nine depots and 162 customers. The authors decomposed the problem into a set of depot-period VRP subproblems and used tabu search to solve them. [8] proposed a tabu search based heuristics to solve a complex variant of MDPVRP dealing with multiple products. The authors also studied the impact of interdependent depot operations and reported a significant improvement in small test instances. [22] solved MDPVRP by a hybrid electromagnetism algorithm. To the best of our knowledge, HGSADC [10] is the only memetic algorithm to solve MDPVRP and is discussed previously in this section.

[23] proposed a multi-threaded integrative cooperative search (ICS) which decomposes MDPVRP into partial problems, i.e., two sets of independent PVRPs and MDVRPs. ICS solves these partial problems independently and integrates them to obtain complete MDPVRP solutions, which may further be improved by a general solver. The authors provide six different versions of ICS. HGSADC-ICS1 and HGSADC-ICS2 utilize HGSADC in two different settings and GUTS-ICS uses a generalized version of the Unified Tabu Search (UTS) [24]. HGSADC-ICS1+, HGSADC-ICS2+, and GUTS-ICS+ enhance the prior three algorithms by including a general solver. [9] proposed a path relinking algorithm (PRA) to solve MDPVRP, which is a population based meta-heuristics employing non-random exploration and exploitation strategies. The authors addressed the problem in two different settings: as a stand-alone algorithm (PRA-Static) and as a part of ICS (PRA-Dynamic).

III. Problem Formulation

To formulate MDPVRPID, we extend the PVRP formulation provided by Cordeau *et. al.* [25]. We modify the cost function and constraints to incorporate multiple depots and also introduce a new constraint (Eq. (6)) and a binary variable z_{ik}. The MDPVRPID is defined on a multigraph $G = (V, A)$, where V is the vertex set and A is the arc set. The vertex set, $V = \{v_1, v_2, v_3, ..., v_n, v_{n+1}, v_{n+2}, ..., v_{n+u}\}$, contains n customers and u depots. Let us define $V^D = \{v_{n+1}, v_{n+2}, ..., v_{n+u}\}$ as the set of depots, in which each depot $v_i \in V^D$ (i.e., vertex $v_{n+i} \in V$) has m identical vehicles associated with it, resulting a total of $\theta = m \times u$ vehicles. The vehicles are subject to a fixed capacity $\epsilon_1, \epsilon_2, ..., \epsilon_\theta$ and a route time constraint $\delta_1, \delta_2, ..., \delta_\theta$. We also define $V^C = \{v_1, v_2, v_3, ..., v_n\}$, where each vertex $v_i \in V^C$ corresponds to a customer and is associated with a nonnegative demand q_i and a service duration d_i. The arc set A is defined as $A = \{(v_i, v_j)^{k,l}\}$, where k and l refer to vehicle number and visit day, respectively. A cost/travel time matrix $C = (c_{ij})$ is associated with A. Here c_{ij} is nonnegative and proportional to the travel time from vertex v_i to v_j. We keep the cost c_{ij} independent of vehicle k and day l for simplicity according to the benchmark problems we use. Extending the problem formulation and also our proposed algorithm from c_{ij} to c_{ijkl} is straight forward.

Our MDPVRPID works with a planning horizon of t days for n customers. Each customer v_i specifies a service frequency f_i and a set of allowable visit combinations, called pattern, R_i. For each day $l = 1, 2, \ldots, t$, the goal is to define one route for each of the θ vehicles such that the total duration of all routes is minimized without violating any vehicle capacity or route time constraints. To formulate MDPVRPID, we define binary variables, x_{ijkl} equal to 1 if and only if vehicle k visits the vertex v_j immediately after visiting v_i $(i \neq j)$ on day l and y_{ir} equal to 1 if and only if the visit combination $r \in R_i$ is assigned to the customer $v_i \in V^C$. Here, $x_{ijkl} = 0$ when both v_i and v_j are depots. We define binary constant w_{rl} equal to 1 if and only if day l belongs to the pattern r. The new binary constant z_{ik} is defined equal to 1 if and only if vehicle k is associated with the depot $v_i \in V^D$.

We assume $d_i = 0$ and $q_i = 0$ for $v_i \in V^D$. MDPVRPID can be stated as follows:

$$\text{Minimize} \sum_{v_i \in V} \sum_{v_j \in V} \sum_{k=1}^{\theta} \sum_{l=1}^{t} c_{ij} x_{ijkl}, \tag{1}$$

subject to

$$\sum_{r \in R_i} y_{ir} = 1 \quad (v_i \in V^C); \tag{2}$$

$$\sum_{v_j \in V} \sum_{k=1}^{\theta} x_{ijkl} - \sum_{r \in R_i} w_{rl} y_{ir} = 0 \quad (v_i \in V^C; l = 1, 2, \ldots, t); \tag{3}$$

$$\sum_{v_i \in V} x_{ihkl} - \sum_{v_j \in V} x_{hjkl} = 0 \quad (v_h \in V; k = 1, 2, \ldots, \theta; l = 1, 2, \ldots, t); \tag{4}$$

$$\sum_{i=n+1}^{n+u} \sum_{v_j \in V} x_{ijkl} \leq 1 \quad (k = 1, 2, \ldots, \theta; l = 1, 2, \ldots, t); \tag{5}$$

$$\sum_{v_j \in V} x_{ijkl} \leq z_{ik} \quad (i = n+1, n+2, \ldots, n+u; k = 1, 2, \ldots, \theta; l = 1, 2, \ldots, t); \tag{6}$$

$$\sum_{v_i \in V} \sum_{v_j \in V} q_i x_{ijkl} \leq \epsilon_k \quad (k = 1, 2, \ldots, \theta; l = 1, 2, \ldots, t); \tag{7}$$

$$\sum_{v_i \in V} \sum_{v_j \in V} (c_{ijkl} + d_i) x_{ijkl} \leq \delta_k \quad (k = 1, 2, \ldots, \theta; l = 1, 2, \ldots, t); \tag{8}$$

$$\sum_{v_i \in S} \sum_{v_j \in S} x_{ijkl} \leq |S| - 1 \quad (S \subseteq V^C; |S| \geq 2; k = 1, 2, \ldots, \theta; l = 1, 2, \ldots, t); \tag{9}$$

$$x_{ijkl} \in \{0, 1\} \quad (v_i \in V; v_j \in V; k = 1, 2, \ldots, \theta; l = 1, 2, \ldots, t); \tag{10}$$

$$y_{ir} \in \{0, 1\} \quad (v_i \in V^C; r \in R_i); \tag{11}$$

$$w_{rl} \in \{0, 1\} \quad (r \in R_i; l = 1, 2, \ldots, t); \tag{12}$$

$$z_{ik} \in \{0, 1\} \quad (i = n+1, n+2, \ldots, n+u; k = 1, 2, \ldots, \theta); \tag{13}$$

The two constraints represented by Eqs. (2) and (3) ensure that only one pattern from the allowable set is assigned to each customer and a customer gets visited (once) only in the days corresponding to the assigned pattern, respectively. Eq. (4) guarantees that a vehicle leaves a vertex on the same day it arrives, while Eq. (5) enforces that each vehicle is used at most once on a single day. Constraint represented by Eq. (6) ensures that a vehicle leaves (and returns) only from (to) the depot it is associated with. Eqs. (7) and (8) guarantee that the vehicle capacity and route time constraints are not violated. At last Eq. (9) eliminates sub tours.

It is worth mentioning that, in MDPVRP, the closest variant of MDPVRPID, each customer is served by the same depot on different days. Hence, every depot works independently within a fixed territory with a fixed set of customers. However, in MDPVRPID, interdependent operations among depots allows one customer to be served by different depots on different days. This relaxation allows MDPVRPID to find cheaper solutions, specially when the product supplies are limited in some depots and/or there are several customers near the territory boundaries. Furthermore, as we do not need to fix depots for customers, our formulation do not require any five-index variables like MDPVRP requires [10].

For homogeneous vehicle fleet, all the vehicles are subject to equal capacity and route time constraints, i.e., $\delta_k = \delta$ and $\epsilon_k = \epsilon$ for $k = 1, 2, \ldots, \theta$. This is the case for the benchmark problem instances used in our computational experiments. Throughout the text we use the notation depot i to denote the vertex $v_i \in V^D$ and customer i to represent the vertex $v_i \in V^C$.

TABLE I: Main Symbols

Symbol	Definition
V	Vertex Set
V^D	Set of depots
V^C	Set of customers
$v_i \in V^C$	Customer i, i.e., $v_i \in V$
$v_i \in V^D$	Depot i, i.e., $v_{n+i} \in V$
n	Number of customers
u	Number of depots
t	Number of days
m	Vehicles per depot
θ	Total number of vehicles
ϵ	Vehicle capacity
δ	Route time constraint
q_i	Demand of customer i
d_i	Service duration of customer i
R_i	Set of visit combinations
μ	Population size
λ	Number of offsprings
$P(i)$	Population of i^{th} generation
$O(i)$	Offspring generated in i^{th} generation
g_{max}	Stopping Criteria
L	Set of solutions undergoing local improvement
$iter$	Number of iterations for simulated annealing
$\mathbf{s}.\pi$	Pattern Component π of solution $\mathbf{s}$
$\mathbf{s}.\rho$	Route set ρ of solution $\mathbf{s}$
$\mathbf{s}.\pi_i$	Pattern for i^{th} customer in solution $\mathbf{s}$
$\mathbf{s}.\rho^{l,k}$	Route for vehicle k on day l in solution $\mathbf{s}$
$1/f(\mathbf{s})$	Total cost of $\mathbf{s}$ including penalty for infeasibility
α, β	Penalty Factors
η	Learning Rate
τ	Elitism ratio in survivor selection
κ_l	Total demand being served on day l

IV. Our Method

We propose a simple memetic algorithm HISMA to solve MDPVRPID. The novelty of the proposed algorithm lies in its intelligent initialization and stochastic learning. Our algorithm starts with an initial population of μ solutions, which undergo crossover and mutation for producing λ offspring. Then, we enhance a subset of solutions from the parent-offspring population by local learning. Finally, a set of μ solutions is chosen from the $(\lambda + \mu)$ candidates for the next generation. We describe the entire procedure in Algorithm 1 and discuss its different components in the following subsections.

Algorithm 1 HISMA

Initialize population $P(0)$ with μ solutions
$i \leftarrow 0$
while $i < g_{max}$ **do**
  $O(i) \leftarrow \emptyset$
  while $|O(i)| < \lambda$ **do**
    Select $\mathbf{s}_1 \in P(i)$ using Roulette Wheel Selection
    Select $\mathbf{s}_2 \in P(i)$ using Fitness Uniform Selection
    Apply crossover on $\mathbf{s}_1$ and $\mathbf{s}_2$ to create offspring $\mathbf{s}$
    Mutate $\mathbf{s}$
    $O(i) \leftarrow O(i) \cup \{\mathbf{s}\}$
  end while
  $L \leftarrow$ Select a set of candidates for local improvement from $P(i) \cup O(i)$
  Improve $\forall \mathbf{s} \in L$ locally by simulated annealing
  Update the solutions of $P(i) \cup O(i)$ which were improved in L
  $P(i+1) \leftarrow \mu$ survivors from $P(i) \cup O(i)$
  $i \leftarrow i + 1$
end while
Return the best feasible solution from $P(g_{max} - 1)$

A. Solution Representation

We represent each solution by a pattern component π and a route set ρ. The pattern component stores the pattern π_i for each customer i, where $\pi_i \in R_i$. The route set ρ is the set of routes, defined as $\rho = \{\rho^{l,k}\}$ for $l = 1, 2, \ldots, t$ and $k = 1, 2, \ldots, \theta$. $\rho^{l,k}$ denotes the route for vehicle k on day l. It is defined as a sequence of customers $\sigma_1, \sigma_2, \ldots, \sigma_{n_{l,k}}$ where $\sigma_i \in V^C$ and $n_{l,k}$ is the number of customers in the route $\rho^{l,k}$. The route $\rho^{l,k}$ starts from the depot $v \in V^D$ with which vehicle k is associated, visits customers $\sigma_1, \sigma_2, \ldots, \sigma_{n_{l,k}}$ sequentially and finally returns to depot v. Throughout the paper, notation like $\mathbf{s}.\pi$ and $\mathbf{s}.\rho$ refer to the pattern component π and route set ρ of a solution $\mathbf{s}$, respectively.

B. Fitness Function

We allow infeasible solutions to be part of the population, as this often helps improving algorithm's performance [26], [27]. Among the constraints defined by Eqs. (2) to (13), the solutions are allowed to violate only the vehicle capacity and/or route time constraints defined by Eqs. (7) and (8), respectively. For incorporating the infeasible solutions, we adopt the penalty function approach to define the fitness $f(\mathbf{s})$ a solution $\mathbf{s}$. $f(\mathbf{s})$ is defined as

$$f(\mathbf{s}) = \frac{1}{\text{total cost}} = \frac{1}{c(\mathbf{s}) + \alpha q(\mathbf{s}) + \beta d(\mathbf{s})} \tag{14}$$

where,

$$c(\mathbf{s}) = \sum_{v_i \in V} \sum_{v_j \in V} \sum_{k=1}^{\theta} \sum_{l=1}^{t} c_{ij} x_{ijkl},$$

$$q(\mathbf{s}) = \sum_{k=1}^{\theta} \sum_{l=1}^{t} max\{0, \sum_{v_i \in V} \sum_{v_j \in V} q_i x_{ijkl} - \epsilon\}, \text{ and}$$

$$d(\mathbf{s}) = \sum_{k=1}^{\theta} \sum_{l=1}^{t} max\{0, (\sum_{v_i \in V} \sum_{v_j \in V} (c_{ijkl} + d_i) x_{ijkl}) - \delta\}.$$

Here $c(\mathbf{s})$ is the actual cost of $\mathbf{s}$ and the expression $\alpha q(\mathbf{s}) + \beta d(\mathbf{s})$ represents the penalty by which s is penalized when it is infeasible. This penalty is a linear combination of the two constraint violations: vehicle capacity violation $q(\mathbf{s})$ and route duration violation $d(\mathbf{s})$. When s is feasible, both $q(\mathbf{s})$ and $d(\mathbf{s})$ is 0, making $f(\mathbf{s})$ equal to $\frac{1}{c(\mathbf{s})}$. The penalty factors α and β are positive constants, which are used to penalize an infeasible solution with an amount proportional to its violation.

C. Initial Population

We use two different techniques, random and heuristic, for constructing an initial population. The reason behind using such a hybridization is to start an evolutionary process using a mixture of good and random solutions. As MDPVRPID is a complex optimization problem, an evolutionary process starting with random solutions may fail to evolve good solutions or even feasible solutions using a reasonable amount of time. Contrarily, a well designed heuristic can provide good initial solutions both in terms of cost and feasibility, which may reduce evolution time. However, if the entire initial population is constructed using heuristic, it may lack the necessary genetic diversity, which may increase the chance of premature convergence. Thus, to balance these two contradictory phenomena, we construct half of the initial population using randomization and propose a heuristic to construct the other half.

The random initialization, termed Cyclic Random Initialization (Operator 1), constructs a solution $\mathbf{s}$ in two phases. At first, it constructs the pattern component π by assigning a random pattern from R_i to each customer i. Then it constructs the route set ρ. For a given day l, it constructs routes for the vehicles incrementally. We define $\chi^l = \{v_i \in V^C \mid w_{\mathbf{s}.\pi_i l} = 1\}$ as the set of customers which are to be served in day l by $\mathbf{s}$. The method initializes each route with an empty sequence. Then it keeps appending random customers from χ^l one at a time to the routes $\rho^{l,k}$, starting with $k = 1$ and then iterating in a cyclic manner i.e., $k = 2, 3, \ldots, \theta, 1, 2, \ldots$. This process is repeated for each day, $l = 1, 2, \ldots, t$.

Operator 1 Cyclic Random Initialization

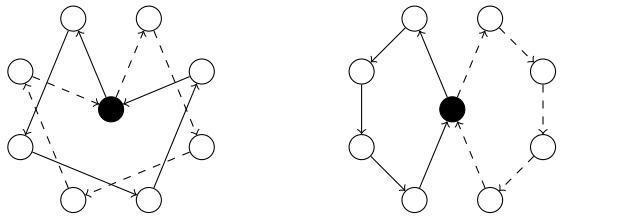
Input: None
Output: An initialized solution $\mathbf{s}$

for $i \leftarrow 1$ to n **do** $\mathbf{s}.\pi_i \leftarrow$ A random visit pattern from R_i
for $l \leftarrow 1$ to t **do**
  Construct $\chi^l = \{v_i \in V^C \mid w_{\mathbf{s}.\pi_i l} = 1\}$
  for $k \leftarrow 1$ to θ **do** $\mathbf{s}.\rho^{l,k} \leftarrow ()$
  $k \leftarrow 1$
  for $\forall v \in \chi^l$ taken in random order **do**
    Append v to $\mathbf{s}.\rho^{l,k}$
    $k \leftarrow k \bmod \theta + 1$
  end for
end for

Similar to cyclic random initialization, the proposed heuristic method first constructs the pattern component and then the

route set. The pattern component is built upon the observation that, the customers which are close to each other, are more likely to be neighbors in the optimal routes and visited consecutively. Therefore, visiting these customers in the same day will tend to improve the overall solution quality as it prefers shorter routes over the longer ones (Fig. 1). Our heuristic thus greedily assigns similar patterns to the close customers. The similarity is measured as the number of service days the patterns share in common. However, if we keep assigning similar patterns to close customers, we may end up on an imbalanced assignment in which many customers get served on a particular day while very few customers on the others. This may lead to an infeasible solution, violating the capacity and/or route time constraints. However, we cannot determine feasibility of a solution before it is completely constructed. We thus try to predict feasibility of a solution using the customers' demands and the vehicles' capacity during the assignment process. We track the total demand of the customers being served on a particular day l and denote it by κ_l. For limiting the number of customer visits per day, we keep κ_l within a $limit$, which is set to the total capacity (i.e., $\epsilon \times \theta$) of all the vehicles available. We do not take into account the route time constraints to predict the feasibility, as the book-keeping would have been complex and time-consuming.

The Pattern Assignment Heuristic, described in Operator 2, requires a guiding route ρ_g containing all the customers to assign patterns. We construct the guiding route ρ_g using the nearest insertion heuristic provided by Karg and Thompson [28], which starts with a route with two random customers and keeps inserting arbitrary customers in the route until all are added. The new customers are inserted between two adjacent customers in the route such that the insertion causes minimal increase in the route cost. To construct the pattern component, we utilize the guiding route ρ_g to assign similar patterns to close customers. We first assign a random pattern to an arbitrary customer of ρ_g and then move along it for assigning patterns to the remaining customers one by one. The pattern for a customer σ_i is selected with respect to r_{prev}, the pattern assigned to the customer visited just before σ_i in ρ_g. We consider each possible pattern r for σ_i and calculate its similarity with r_{prev}. We also maintain a binary variable ψ, which is set to 1 if assigning r can cause κ_l to exceed the predefined $limit$ for any l where $w_{rl} = 1$. The pattern which is most similar to r_{prev} and also keeps each κ_l within $limit$ (i.e., $\psi = 0$) is selected. However, in some cases, all choices for r may lead to $\psi = 1$. In such a case, the most similar pattern is taken. At last, we update κ_l for $l = 1, 2, \ldots, t$ and move along the guiding route to assign a pattern to the next customer.



(a) Initialized with random patterns

(b) Initialized with the proposed pattern assignment heuristic

Fig. 1: Two sample solutions for a simple problem with one depot (solid circle) containing one vehicle, eight customers (hollow circles), and two days. All the customers have a frequency of one and can be served on both days. The routes of day 1 and 2 are represented by solid and dashed edges, respectively. It can be seen that solution (b) costs less than solution (a) as the close customers are served on the same day.

To construct the route set, we first create temporary routes for the depots and the actual vehicle routes are extracted from those. For each day, we construct one temporary route for each depot with the customers which are to be visited on that particular day, according to the pattern component constructed earlier. Each customer is included only in the route of its nearest depot. The temporary routes are created using the well-known tour construction heuristic Generalized Insertion Procedure (GENI) [29]. GENI starts with a route of three random customers and repeatedly inserts new customers in it using two different kind of insertions. The insertions involve a choice of three or, four customers already present in the route. GENI restricts this choice within the p closest neighbors of the customer, which is called its p-neighborhood. After the route for a depot is designed, it is cut into m equal pieces and assigned as the routes of its associated vehicles. It should also be noted that, the heuristic is stochastic in nature. Thus the initialized solutions will be different from each other.

D. Parent Selection

Parent pool selection is usually done with random or heuristic methods. A random method can facilitate exploration but may struggle to maintain the selection pressure necessary for a decent evolutionary progress. On the other hand, a heuristic based method can create the sufficient selection pressure, but its greedy nature may limit the exploration of the search space. Thus to bring a balance between the exploration of search space and acquiring a reasonable selection pressure, HISMA adopts two different operators to select the parents for crossover. One is the roulette wheel selection operator, which facilitates the selection pressure by selecting better solutions with higher probabilities and the other is the Fitness Uniform Selection Strategy (FUSS) [30], which aids the exploration of the search space by selecting solutions uniformly from the entire fitness landscape. In this way, HISMA emphasizes both exploration and exploitation in selecting parents for crossover. It should be mentioned that, the selection method can select both feasible and infeasible solutions. Crossover between feasible and infeasible solution helps HISMA to find solutions lying in the edge of feasible regions, where good quality solutions are expected to be found.

E. Crossover

Our crossover operator produces one offspring from a pair of parent solutions. We denote the better parent by $\mathbf{s}_b$, the

Operator 2 Pattern Assignment Heuristic

Input: A guiding route $\rho_g = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ containing all customers $\sigma_i \in V^C$

Output: Pattern component π.

1: **for** $l = 1, 2, \ldots, t$ **do** $\kappa_l \leftarrow 0$
2: $i \leftarrow$ A random number from $[1, n]$
3: $\pi_{\sigma_i} \leftarrow$ A random pattern r from R_{σ_i}
4: $r_{prev} \leftarrow \pi_{\sigma_i}$
5: $i \leftarrow (i \mod n) + 1$, $j \leftarrow 1$
6: **while** $j < \left|V^C\right|$ **do**
7: $max_f \leftarrow -1$, $max \leftarrow -1$
8: **for** $\forall r \in R_{\sigma_i}$ **do**
9: $point \leftarrow$ Similarity between r and r_{prev}
10: **if** $point > max$ **then** $max \leftarrow point$, $best \leftarrow r$
11: **if** $(point > max_f)$ **then**
12: $\psi \leftarrow 0$
13: **for** $l = 1, 2, \ldots, t$ **do**
14: **if** $(w_{rl} = 1) \wedge ((\kappa_l + q_{\sigma_i}) > limit)$ **then**
15: $\psi \leftarrow 1$
16: **end if**
17: **end for**
18: **if** $(\psi = 0)$ **then** $max_f \leftarrow point$, $best_f \leftarrow r$
19: **end if**
20: **end for**
21: **if** $max_f \neq -1$ **then** $\pi_{\sigma_i} \leftarrow best_f$
22: **else** $\pi_{\sigma_i} \leftarrow best$
23: **for** $l = 1, 2, \ldots, t$ **do** $\kappa_l \leftarrow \kappa_l + q_{\sigma_i} \times w_{\pi_{\sigma_i} l}$
24: $r_{prev} \leftarrow \pi_{\sigma_i}$
25: $i \leftarrow (i \mod n) + 1$, $j \leftarrow j + 1$
26: **end while**

worse one by $\mathbf{s}_w$, and the offspring solution by $\mathbf{o}$. The crossover operator is hybrid in nature, using separate methods for the pattern component π and the route set ρ. Operator 3 discusses it in detail. First, π of the offspring is constructed using uniform crossover, as shown in Line 1. Once π is constructed, the crossover operator constructs ρ. Each route in ρ can be formed either solely from one parent or by a combination of both. This strategy enables the parents to facilitate the exploration of the search space, while at the same time pass some good routes or sub-routes to the offspring [10].

The route set ρ is constructed incrementally, first the routes for day 1, then for day 2, and so on. For a particular day l, the crossover operator first randomly partitions the set of all vehicles K in three non-empty disjoint sets, K^B, K^M, and K^W. The sets are typically unequal in size, K^B being the largest and K^W being the smallest. The route $\rho^{l,k}$ of a vehicle $k \in K^B$ is directly copied from the better parent $\mathbf{s}_b$ (Line 5), while the routes of K^W are directly copied from the worse parent $\mathbf{s}_w$ (Line 7). The routes of K^M are designed by mixing the routes from both $\mathbf{s}_b$ and $\mathbf{s}_w$. First, a sub-route from $\mathbf{s}_b$ is copied to the offspring (Line 6). Then, we pick customers from the route of $\mathbf{s}_w$ one by one and append them to the offspring route in the same order (Line 8). At this point some customers may still remain unvisited in the offspring and lines 9 and 10 add such customers from $\mathbf{s}_b$ and $\mathbf{s}_w$ sequentially. It should be mentioned that, while copying or appending customers from the parents to an offspring, a customer is added only if it is to be visited on that particular day and has not been added to any other offspring route previously on that day. In Operator 3 the operation *copy* and *append* are designed accordingly.

Operator 3 Crossover

Input: Two parent solutions, $\mathbf{s}_b$ and $\mathbf{s}_w$, where $\mathbf{s}_b$ has a higher fitness value than $\mathbf{s}_w$

Output: An offspring solution $\mathbf{o}$

1: **for** $i \leftarrow 1$ to n **do** $\mathbf{o}.\pi_i \leftarrow \mathbf{s}_b.\pi_i$ or, $\mathbf{s}_w.\pi_i$ randomly
2: Define $K = \{1, 2, \ldots, \theta\}$ as the set of all vehicles
3: **for** $l = 1, 2, \ldots, t$ **do**
4: Partition K in three non-empty disjoint subsets K^B, K^M, and K^W randomly such that $|K^B| \geq |K^M| \geq |K^W|$
5: **for** $\forall k \in K^B$ **do** *copy* $\mathbf{s}_b.\rho^{l,k}$ to $\mathbf{o}.\rho^{l,k}$
6: **for** $\forall k \in K^M$ **do** *copy* a random sub-route from $\mathbf{s}_b.\rho^{l,k}$ to $\mathbf{o}.\rho^{l,k}$
7: **for** $\forall k \in K^W$ **do** *copy* $\mathbf{s}_w.\rho^{l,k}$ to $\mathbf{o}.\rho^{l,k}$
8: **for** $\forall k \in K^M$ **do** *append* $\mathbf{s}_w.\rho^{l,k}$ to $\mathbf{o}.\rho^{l,k}$
9: **for** $\forall k \in (K^W \cup K^M)$ **do** *append* $\mathbf{s}_b.\rho^{l,k}$ to $\mathbf{o}.\rho^{l,k}$
10: **for** $\forall k \in K^B$ **do** *append* $\mathbf{s}_w.\rho^{l,k}$ to $\mathbf{o}.\rho^{l,k}$
11: **end for**

F. Mutation Scheme

We use three different categories of mutation that work with different aspects of the solutions. The operators in the first category work with the order of the customers within a single route, which we term the intra-route operator. The second category consists of route re-assignment operators that manipulate the assignment of customers to routes/vehicles. The last category is pattern re-assignment operators and deals with the patterns associated with customers. Both route and pattern re-assignment operators change visit orders of multiple routes.

Our HISMA uses three mutation operators: intra-route random insertion, 1-0 exchange, and random pattern re-assignment, one from each of the three different categories. Each offspring created by crossover is refined by one of these mutation operators with uniform probability. The intra-route random insertion operator mutates the visit order of a random route by selecting a random customer from the route and reinserting it in a different random position. The well-known 1-0 exchange, a route re-assignment operator, relocates a random customer to a different route. It first removes the customer from its route and then inserts it in a different route randomly. The third operator, random pattern re-assignment, mutates the pattern component π of the offspring by changing the visit pattern of a random customer. It selects a random customer i and assigns a different visit pattern to it randomly from R_i. It also needs to update the route set ρ for consistency, as the customer i will now be visited on some days on which it previously did not. For such a day, it selects a random route and inserts the customer in a random position. It also removes all visits to customer i on any day, which does not belong to the newly assigned visit pattern.

G. Learning

We incorporate learning, a local search procedure, in our algorithm to enhance the solutions' quality by exploiting their neighborhood. We adopt the Lamarckian approach, which allows learning to alter the solution's genotype. This alternation helps in transmitting acquired characteristics to the new solutions. Learning takes place in every iteration (generation) i of our HISMA. Unlike previous work, it selectively applies stochastic learning i.e., simulated annealing on a subset of solutions L, which is selected from the combination of parent and offspring population $P(i) \cup O(i)$ by binary tournament. Several issues arise while employing learning inside the evolutionary process. These include determining the candidate pool to select L, the size of L, the procedure of selecting L, the choice of learning, and its operators to exploit the neighborhood.

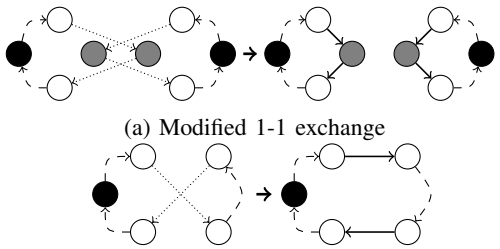
Two different approaches are found in literature for selecting the candidate pool. One approach selects only the parent population $P(i)$ (e.g., [31] and [32]) and the other only the offspring population $O(i)$ (e.g., [33] and [34]) as the candidate pool. If we consider only $P(i)$, the newly generated offspring solutions from $O(i)$ will have a lower chance to survive to the next generation. In contrast, if we consider only from $O(i)$, no solution can undergo the learning procedure more than once throughout evolution. This approach bears similarity to select the best solutions obtained from different local searches running with different initial solutions. Therefore, we consider solutions from both $P(i)$ and $O(i)$ for learning.

For determining the size of L, one simple option would be to select the whole parent-offspring population $P(i) \cup O(i)$. A significant amount of computational resource will be required in this strategy. The learning may also search in unpromising regions where the global optimum is not present. To avoid the aforementioned problems, we select a only a subset of $(\lambda + \mu) \times \eta$ solutions from $P(i) \cup O(i)$ to undergo learning. Here, η is a real valued parameter in $[0, 1]$ and called the learning rate. We also need a mechanism to select the solutions from the parent-offspring population as we are not improving every solution. It has been seen that, elitism in this selection procedure produces significantly better results, especially when the local search is selective [35]. Hence, we select the solutions by binary tournament as it supports a fair amount of elitism and is also faster than roulette wheel selection.

H. Simulated Annealing

HISMA uses simulated annealing as local search, which being a stochastic search method allows uphill moves (deterioration) in addition to downhill moves (improvement). Thus, it has a less probability to trap into local optima than other greedy search techniques (e.g. hill climbing). This is the main reason for choosing simulated annealing for improving the existing solutions by considering their neighborhood.

Similar to mutation, we use three different categories of operators for exploiting the neighborhood of the solutions. Each type contains a mix of random and heuristic based greedy operators. The greedy operators always improve (or at least do not deteriorate) a given solution. The random operators, on the other hand, make no such promises, allowing deterioration. In each iteration we first randomly select one of the three categories and then apply one operator from the selected category with uniform probability.



(a) Modified 1-1 exchange

(b) 2-opt

Fig. 2: Some neighborhood operators used by HISMA. Here, normal customers and depots are represented by the hollow and solid circles, respectively. Customers whose routes are changed by the operators are represented by the shaded circles. The dotted edges are removed by the operators, while the solid edges represent the newly created ones. The intermediate routes between two nodes (customers/depots) are represented by the dashed curves.

The first category contains five intra-route operators, i.e, intra-route random insertion, intra-route 1-1 exchange, 2-opt, 3-opt, and Or-opt. The intra-route random insertion operator randomly selects a customer and reinserts it in a different random position in the same route. The intra-route 1-1 exchange operator randomly selects a pair of customers and swaps them. The 2-opt operator removes two edges from a route and reconnects them in such a way that it does not cross over itself [36]. The 3-opt operator works in a similar fashion, but instead of two, three edges are reconnected [37]. The Or-opt operator attempts to improve a random route by moving a chain of three or fewer consecutive customers in a different location [38]. For 2-opt, 3-opt, and Or-opt, we adopt first improvement strategy, i.e., the first move resulting an improvement is chosen.

The route re-assignment category contains two operators: modified 1-1 exchange and inter-route Or-opt. The modified 1-1 exchange operator exchanges routes of two random customers for an arbitrary day. The customers are removed from their existing routes and inserted in the other ones using the nearest insertion heuristic [28]. The inter-route Or-opt operator considers moving three or fewer consecutive customers from one route and placing it (and also its reverse) between two consecutive customers in a different route to improve the solution's quality. We adopt first improvement strategy here too.

Random pattern re-assignment and pattern improvement operators work with the pattern component of the solutions. The former operator reassigns the pattern of a customer randomly. The later operator randomly selects a customer and selects a pattern for it which produces the minimal cost solution. After applying the operators, if the customer gets served on a day on which it previously did not, it is inserted using nearest insertion heuristic. The operators also remove the customer visits from the days which do not belong to the newly assigned pattern.

I. Survivor Selection

The survivor selection scheme is a combination of pure elitist and roulette wheel techniques. The best $\mu \times \tau$ solutions from the $(\lambda + \mu)$ candidates always propagate to the next generation for preserving the fittest solutions. Here τ, a real valued parameter in $[0, 1]$, is called the elitism ratio, which determines the degree of the elitism. The rest of the survivors are selected using roulette wheel selection to create a reasonable selective pressure or to give some avenue to maintain diversity.

V. Experimental Studies

In order to assess the performance of HISMA, the algorithm is tested on 10 benchmark problem instances taken from [10]. These problems have been studied under MDPVRP formulation (e.g., in [9], [10], and [23]) and suit our MDPVRPID formulation without any change. In these problems, the number of customers (n) varies from 48 to 288. The first six problems are characterized by an equal number of depots u(=4) and days t(=4). However, u and t are increased to 6 for the last four problems. The route time constraints δ and capacity constraints ϵ range from 400 to 500 and 170 to 200, respectively. The number of vehicles per depot (m) ranges from 1 to 3. Table II shows the details.

TABLE II: Characteristics of the 10 benchmark instances taken from [10]. Here n, u, m, and t represent the number of customers, depots, vehicles per depot, and days, respectively. δ and ϵ indicate, respectively, the route time and capacity constraints of the vehicles.

Problem Instance	n	m	u	t	δ	ϵ
pr01	48	1	4	4	500	200
pr02	96	1	4	4	480	195
pr03	144	2	4	4	460	190
pr04	192	2	4	4	440	185
pr05	240	3	4	4	420	180
pr06	288	3	4	4	400	175
pr07	72	1	6	6	500	200
pr08	144	1	6	6	475	190
pr09	216	2	6	6	450	180
pr10	288	3	6	6	425	170

A. Experimental Setup

The population size μ and offspring population size λ are both set equal to 100. The stopping criteria, i.e., the number of generations g_{max}, is assigned to 2000. For penalizing the infeasible solutions, the penalty factors α and β are set equal to the number of customers for keeping them proportional to the problem size, where the motivation is to facilitate a large search space to find feasible solutions for large problems. The elitist ratio τ used in survivor selection and the learning rate η are set to 0.5. It is worth mentioning that, none of the parameter values used here are intrinsic property of HISMA and can be made adaptive in future. According to [29], we define the *p-neighborhood* with $p = 7$ for constructing the routes in heuristic initialization.

We implement simulated annealing as outlined in [39]. The only exception is that instead of selecting a random neighbor, we generate both random and heuristic-generated neighbors. The scheduling function is given in Eq. (15).

$$schedule(\iota) = \begin{cases} \phi e^{-\gamma \iota}, & \text{if } \iota < iter \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

The values of ϕ, γ, and $iter$ are set to 20, 0.01, and 1000, respectively. Although HISMA uses a basic version of simulated annealing, it can easily incorporate any other improved versions (e.g., [40]).

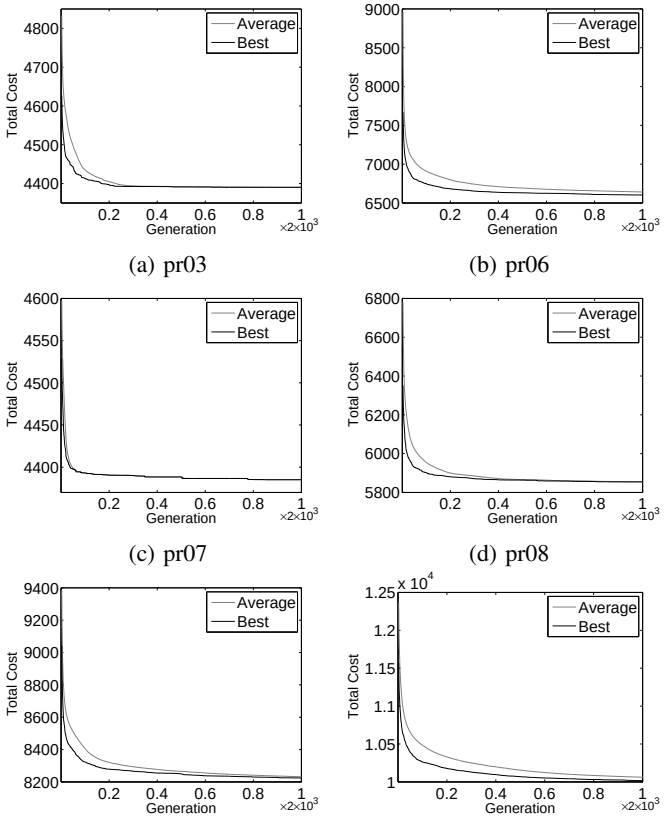
To the best of our knowledge, none of the previous studies deal with MDPVRPID. We thus compare HISMA with two baseline algorithms, Basic Genetic Algorithm (BGA) and Basic Simulated Annealing (BSA), to show the efficacy of our method. BGA constructs its entire initial population with random solutions, which are created with a slightly different version of Cyclic Random Initialization (Operator 1) used by HISMA. Unlike Operator 1, BGA selects the routes randomly. Both BGA and HISMA employ the same crossover and mutation. The values of μ, λ, α, and β for BGA are also set same as HISMA. However, as HISMA evolves for g_{max} generations and applies simulated annealing for $iter$ iterations, the number of generations for BGA is set to $g_{max} \times iter$ for a fair comparison. BSA, similar to the simulated annealing in HISMA, uses Eq. 15 as the scheduling function. However, it uses HISMA's mutation operators to generate neighbors and runs for $g_{max} \times iter$ iterations. The values of ϕ and γ are also changed to 4×10^4 and 5×10^{-6}, respectively.

HISMA, along with BGA and BSA, is implemented in Java SE 7 (1.7.0_71). All the experiments are run on a PC with a Intel(R) Core(TM) i5-3470 CPU running at 3.20 GHz with 4GB RAM.

B. Result

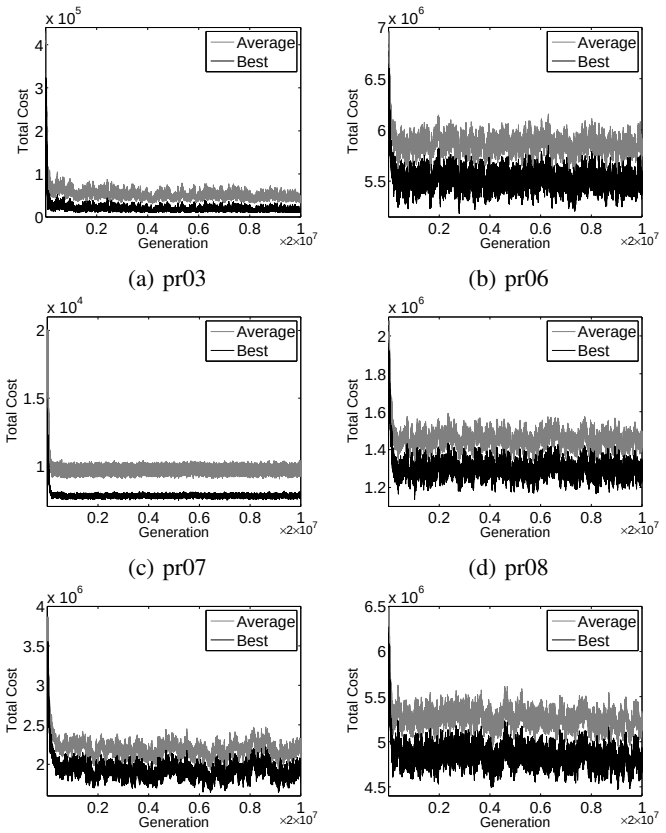
1) Evolutionary Progress: We here analyze the progress of HISMA, BGA, and BSA. For brevity, we consider pr03, pr06, pr07, pr08, pr09, and pr10 problem instances. Figures 3 and 4 show average and minimum (i.e., population best) total costs (i.e., $1/f(\mathbf{s})$) of solutions over the generations for HISMA and BGA, respectively. As BSA is not a population based method, we plot the total solution costs over its iterations in Fig. 5. All the values in the figures are an average of 10 independent runs. The first few generations of HISMA show a very rapid improvement. Hence, we omit the first $g_{max} \times 0.25\% = 5$ generations of the plots in Fig. 3 for proper inspection. To maintain similarity for comparison Figs. 4 and 5 also omit the first 0.25% of their total generations/iterations.

Observing Fig. 3 it can be noticed that, HISMA progresses very rapidly in the initial generations and the progress slows down in the later phase of evolution. For problem instances with less than 200 customers (i.e., pr03, pr07, and pr08), the average population and best solution curves converge together (Fig. 3a, 3c, and 3d). Contrarily, for the problems with more than 200 customers (i.e., pr06, pr09, and pr10) the curves show a little gap (Fig. 3b, 3e, and 3f). This reflects that for smaller instances the whole population converge to a single solution or a very concentrated region of the search space, while for the larger instances the population holds its diversity. On the other



(a) pr03 (b) pr06 (c) pr07 (d) pr08 (e) pr09 (f) pr10

Fig. 3: Evolutionary progress of HISMA for the selected problem instances. The upper and lower curves plot the average and minimum (i.e., population best) solution cost against generation, respectively. All the values are obtained by averaging data from 10 independent runs.



(a) pr03 (b) pr06 (c) pr07 (d) pr08 (e) pr09 (f) pr10

Fig. 4: Evolutionary progress of BGA for the selected problem instances. The upper and lower curves plot the average and minimum (i.e., population best) solution cost against generation, respectively. All the values are obtained by averaging data from 10 independent runs.

hand, the evolutionary progress of BGA stops very early and exhibits large and rapid fluctuations (Fig. 4), irrespective of the problem size. Lastly, similar to HISMA, BSA shows fast improvement in its initial iterations and the progress slows down afterwards (Fig. 5). However, the initial iterations of BSA show large fluctuations unlike HISMA, which gradually decreases with the algorithm progress.

2) Baseline Comparison: Table III presents the average total cost (i.e., $1/f(\mathbf{s})$), normalized standard deviation, and percentage of feasible solutions obtained from 10 independent runs of HISMA, BGA, and BSA. To compare the algorithms, we make the following observations.

i) In terms of average solution quality, HISMA outperforms the other baseline methods by a great margin, while BGA shows the worst performance. For example, the average total cost of solutions provided by HISMA for pr05 is 5643.21, while it is 7435.58 and 1817584.59 for BSA and BGA, respectively.

ii) HISMA has the least normalized standard deviation among the three methods. It is the indication of HISMA's consistency of producing good solutions in different runs. As an example, for pr07, the normalized standard deviation of HISMA, BSA, and BGA are 0.0004, 0.0138, and 0.0268, respectively.

iii) Both BGA and BSA perform poorly in terms of feasibility of the solutions. BSA never finds any feasible solution for any of the 10 problems. BGA also struggles to find feasible solutions, obtaining 100% feasibility only for the problems pr01 and pr07. Our HISMA, on the other hand, yields feasible solutions in all the runs.

3) Comparison with other algorithms: Here, we compare the performance of HISMA with the state-of-the-art algorithms: HGSADC [10], two versions of PRA, PRA-Static and PRA-Dynamic [9], and six different versions of ICS provided by [23]. It is worth mentioning that, these algorithms solve the benchmark instances as MDPVRP, while HISMA solves the same under MDPVRPID formulation. Brief description of these algorithms can be found in Section II. All the solutions obtained by HISMA and these algorithms are feasible, which makes the total cost equal to actual cost (see Eq. 14). Thus, for the rest of this section, the "total cost" of the solutions are referred only as "cost" for better readability.

We use the average solution cost as the comparison metric. Table IV shows the average solution costs of 10 independent runs for HISMA and the other methods. We collect the results

TABLE III: Comparison among HISMA, BSA, and BGA based on the results of 10 independent runs. The column labeled Avg shows average total cost and Nsd is the normalized standard deviation. Fsb indicates the percentage of feasible solutions obtained from different runs. The best results are shown in bold-face type.

Problem Instance	HISMA			BSA			BGA		
	Avg	Nsd	Fsb	Avg	Nsd	Fsb	Avg	Nsd	Fsb
pr01	**1976.06**	**0.0020**	**100.00**	2055.72	0.0137	0.00	3191.25	0.0188	**100.00**
pr02	**3494.10**	**0.0012**	**100.00**	4020.57	0.0278	0.00	167890.89	0.1075	0.00
pr03	**4390.49**	**0.0059**	**100.00**	5105.30	0.0257	0.00	12207.30	0.1576	10.00
pr04	**5109.34**	**0.0034**	**100.00**	6385.48	0.0228	0.00	1709239.35	0.0662	0.00
pr05	**5643.21**	**0.0031**	**100.00**	7435.58	0.0323	0.00	1817584.59	0.0883	0.00
pr06	**6602.85**	**0.0024**	**100.00**	9801.75	0.0910	0.00	5508975.98	0.0323	0.00
pr07	**4385.11**	**0.0004**	**100.00**	4788.61	0.0138	0.00	7801.17	0.0268	**100.00**
pr08	**5854.45**	**0.0028**	**100.00**	7331.07	0.0346	0.00	1316758.63	0.0718	0.00
pr09	**8223.37**	**0.0026**	**100.00**	10914.75	0.0213	0.00	1887536.98	0.1233	0.00
pr10	**10015.21**	**0.0034**	**100.00**	15059.85	0.0157	0.00	4695954.27	0.0431	0.00

TABLE IV: Comparison among HGSADC [10], static and dynamic versions of PRA [9], different versions of ICS [23], and HISMA for the 10 benchmark problem instances. Each result represents an average of 10 independent runs. The best results are shown in bold-face type.

Problem Instance	HGSADC	PRA		HGSADC				GUTS		HISMA
		-Static	-Dynamic	-ICS1	-ICS1+	-ICS2	-ICS2+	-ICS	-ICS+	
pr01	2019.07	2019.07	2019.07	2019.07	2019.07	2019.07	2019.07	2019.07	2019.07	**1976.06**
pr02	3547.45	3547.45	3547.45	3547.45	3547.45	3547.45	3547.45	3547.45	3547.45	**3494.10**
pr03	4491.08	4480.87	4478.12	4480.87	4480.87	4480.91	4480.87	4481.51	4482.21	**4390.49**
pr04	5151.73	5149.64	5147.75	5149.17	5148.79	5148.27	5144.42	5163.29	5153.83	**5109.34**
pr05	5605.6	5598.32	5585.16	5598.38	5587.97	5604.86	**5582.86**	5633.24	5630.05	5643.21
pr06	6570.28	6568.79	6541.8	6555.72	6551.72	6551.77	**6538.54**	6585.77	6586.48	6602.85
pr07	4502.06	4502.02	4502.02	4502.02	4502.02	4502.02	4502.02	4502.02	4502.02	**4385.11**
pr08	6029.58	6027.51	6023.98	6023.99	6024.01	6023.98	6023.98	6024.08	6024.15	**5854.45**
pr09	8310.19	8304.26	8279.55	8281.60	8273.77	8277.61	8274.42	8306.93	8302.88	**8223.37**
pr10	9972.35	9963.55	**9839.07**	9949.62	9902.44	9984.54	9907.24	10071.34	10040.94	10015.21
Average	5619.94	5616.15	5596.40	5610.79	5603.81	5614.05	5602.09	5633.47	5628.91	**5569.42**

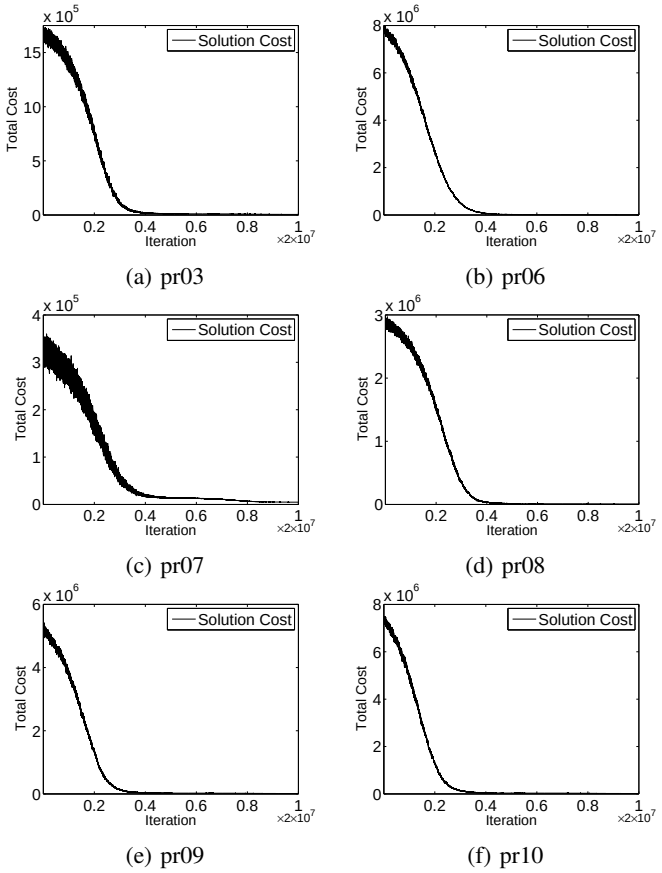
of the other algorithms from [9], [10], and [23]. It is seen that HISMA provides the smallest overall average cost i.e., 5569.42 among all the algorithms. PRA-Dynamic has the second lowest 5596.40, while GUTS-ICS has the highest average solution cost 5633.47. It is also observed that our HISMA provides the best result for 7 out of the 10 benchmark instances, except pr05, pr06, and pr10. HGSADC-ICS2+ obtains the best results for pr05 and pr06, while PRA-Dynamic provides the best result for pr10.

For assessing the statistical significance, we conduct the Wilcoxon signed-rank test [41] in a pairwise manner, HISMA vs. the other algorithms one at a time. We conduct the test on the z-scores of the average solution costs. The reason is that the Wilcoxon signed-rank test works on the difference of pair values. It is thus necessary to normalize these values before taking the difference. z-score provides the normalized values when upper and lower bounds are not known. To compute the z-scores, we first calculate the sample mean and standard deviation of the average solution costs for each problem instance over all the algorithms. Then for each algorithm, the z-score for a problem instance is calculated by subtracting the average solution cost obtained by that algorithm from the sample mean for that particular problem instance and then by dividing the resultant difference with the sample standard deviation.

We conduct the Wilcoxon signed-rank test under a significance level of 0.10. For each problem, we give a positive rank to HISMA if it was better than the competing algorithm and vice versa. Table V provides the detailed result of the tests. $R+$ and $R-$ represent the sum of all positive and negative ranks, respectively. The test statistic T_{value}, minimum of $R+$ and $R-$, should be less than or equal to the critical value 10

TABLE V: Detailed result of Wilcoxon Signed Rank Test between HISMA and other algorithms. Each row shows the name of the competing algorithm and the values of $R+$, $R-$, and T_{value} resulted from the test. HISMA's superiority is indicated when $R+$ exceeds $R-$. A T_{value} less than or equal to 10 indicates statistical significance under a significance level of 0.10.

Competing Algorithm	$R+$	$R-$	T_{value}
HGSADC	49	6	6
PRA - Static	49	6	6
PRA- Dynamic	45	10	10
HGSADC -ICS1	49	6	6
HGSADC -ICS1+	47	8	8
HGSADC -ICS2	48	7	7
HGSADC-ICS2+	45	10	10
GUTS -ICS	52	3	3
GUTS-ICS+	50	5	5



(a) pr03 (b) pr06

(c) pr07 (d) pr08

(e) pr09 (f) pr10

Fig. 5: Progress of BSA for the selected problem instances. The curves plot solution costs for each algorithm iteration. The values are obtained by averaging data from 10 independent runs.

to indicate a significantly different result. A greater $R+$ than $R-$ conclude the superiority of HISMA over the competing algorithm. The Wilcoxon signed-rank test reveals that HISMA outperforms all the other methods under a significance level of 0.10, obtaining a greater $R+$ in every case with no T_{value} greater than 10.

4) Comparison with Best Known Solution (BKS): Table VI compares the best solutions obtained by HISMA during its 10 runs (under MDPVRPID formulation) with the BKS obtained from the literature under MDPVRP formulation. It is seen that, HISMA improves the BKS for 7 out of the 10 benchmark instances. For the problems pr01, pr03, pr07, and pr08, the improvement is more than 2%. The greatest improvement, i.e., 2.95%, is found for problem pr08. It is worth mentioning that, the BKS shown for HGSADC and HGSADC-ICS2+ are obtained from all of their experimental runs, rather than the 10 runs for which the average costs were shown in Table IV.

C. Analysis of HISMA

This section analyzes the sensitivity of HISMA on its different parameters and components.

1) Effect of Parameters: HISMA involves two intrinsic parameters i.e., the elitism ratio τ in survivor selection and learning rate η. We analyze the sensitivity of HISMA on these parameters by repeatedly varying one parameter at a time while holding the other one fixed. We run the experiments on pr06, pr09, and pr10 for brevity.

Table VII shows average total cost, normalized standard deviation, and feasibility percentage of HISMA for different values of τ and a fixed η i.e., 0.25. It is observed that HISMA performs better when the survivor selection incorporates some level of elitism by setting $\tau > 1$. The least average total cost is obtained with pure elitism ($\tau = 1$), while the largest is found when there is no elitism ($\tau = 0$). Similarly, HISMA shows lower normalized standard deviation i.e., more consistency when τ is greater than 1. However, irrespective of the value of τ, HISMA obtains 100% feasibility.

Table VIII summarizes performance of HISMA for different values of learning rate η and a fixed τ i.e., 0.25. It is observed that, HISMA performs satisfactorily when learning is allowed. However, the performance deteriorates remarkably when it is disabled by setting $\eta = 0$. The solution costs become significantly higher and the algorithm's consistency deteriorates too. Also the algorithm occasionally fails to produce feasible solutions. For example, all the solutions for pr06 are infeasible when $\eta = 0$. The above discussion justifies our choice of introducing learning in the algorithm.

2) Effect of Components: Here we inspect the impact of different components of HISMA on its performance. We first analyze the effect of heuristic initialization on HISMA by modifying it to construct the entire initial population randomly using Operator 1. We call this version HISMA-WH. Table IX compares the performance of these two versions. It is seen that without heuristic initialization the performance degrades. For example the average solution cost obtained by HISMA for pr10 is 10015.21, and for HISMA-WH it is 10680.8, which is 6.65% worse. HISMA-WH is also remarkably less consistent, showing much higher normalized standard deviation than HISMA. However, both the approaches always provide feasible solutions.

TABLE VI: Comparison between the BKS under MDPVRP formulation and HISMA's best solutions. The column PI stands for problem instance. "BKS Method" specifies the name of the method which achieves the MDPVRP BKS, where "all" means the BKS is achieved by all the versions of PRA and ICS, and HGSADC. Gap represents the percent deviation of HISMA from BKS and is calculated as $\frac{(HISMA-BKS)}{BKS} \times 100\%$. The best results are shown in boldface type.

PI	MDPVRP BKS	BKS Method	HISMA	Gap
pr01	2019.07	All	**1973.04**	-2.2798
pr02	3547.45	All	**3490.81**	-1.5966
pr03	4472.22	PRA-Dynamic	**4363.32**	-2.4350
pr04	5134.17	HGSADC, PRA	**5093.48**	-0.7925
pr05	**5558.02**	HGSADC-ICS2+	5602.4	0.7985
pr06	**6524.92**	HGSADC	6568.82	0.6728
pr07	4502.02	All	**4382.11**	-2.6635
pr08	6023.98	All	**5846.4**	-2.9479
pr09	8254.73	HGSADC-ICS2+	**8193.96**	-0.7362
pr10	**9776.28**	HGSADC-ICS2+	9946.62	1.7424

TABLE VII: Performance of HISMA for different values of τ based on the results of 10 independent runs. The column labeled Avg shows average total cost and Nsd is the normalized standard deviation. Fsb indicates the percentage of feasible solutions obtained from different runs. The best results are shown in bold-face type.

τ	pr06			pr09			pr10		
	Avg	Nsd	Fsb	Avg	Nsd	Fsb	Avg	Nsd	Fsb
0.00	7011.01	0.0123	**100**	8757.34	0.0084	**100**	11469.82	0.0153	**100**
0.25	6653.21	0.0040	**100**	8310.89	0.0033	**100**	10212.14	0.0050	**100**
0.50	6672.72	**0.0030**	**100**	8310.79	0.0031	**100**	10212.43	**0.0022**	**100**
0.75	6706.82	0.0042	**100**	8315.37	**0.0014**	**100**	10219.99	0.0045	**100**
1.00	**6643.62**	0.0041	**100**	**8266.52**	0.0017	**100**	**10105.43**	0.0044	**100**

TABLE VIII: Performance of HISMA for different values of η based on the results of 10 independent runs. The column labeled Avg shows average total cost and Nsd is the normalized standard deviation. Fsb indicates the percentage of feasible solutions obtained from different runs. The best results are shown in bold-face type.

η	pr06			pr09			pr10		
	Avg	Nsd	Fsb	Avg	Nsd	Fsb	Avg	Nsd	Fsb
0.00	152371.15	0.1973	0	12435.49	0.0130	**100**	17090.78	0.0487	90
0.25	6680.08	0.0029	**100**	8317.22	**0.0030**	**100**	10175.61	**0.0034**	**100**
0.50	**6652.11**	0.0031	**100**	8291.96	0.0037	**100**	**10149.14**	0.0076	**100**
0.75	6652.96	0.0044	**100**	8284.88	0.0054	**100**	10167.86	0.0051	**100**
1.00	6659.22	**0.0026**	**100**	**8264.40**	0.0037	**100**	10169.93	0.0041	**100**

Next we assess the importance of local learning in our proposed HISMA. For this, we disable the local learning in the proposed approach to obtain HISMA-WL. Table IX presents its average performance from 10 independent runs. It is seen that HISMA performs poorly without learning. The average total cost obtained by HISMA-WL is much higher than HISMA. HISMA-WL is also less consistent than HISMA and suffers from infeasible solutions too. For example, all 10 runs of HISMA-WL resulted in infeasible solutions for pr06. The experiments thus suggest that local learning plays an inevitable role in HISMA's performance.

We also assess the impact of the stochasticity in local learning by simply replacing simulated annealing with a greedy search method to obtain HISMA with greedy learning (HISMA-GL). Similar to simulated annealing, the greedy search also runs for a fixed number of iterations ($iter$) and works with the same set of operators. However it only moves to a better neighbor. The results of HISMA-GL along with HISMA's are provided in Table X for comparison. It can be seen that HISMA-GL provides suboptimal solutions i.e., 2.90%, 2.87%, and 4.08% worse results than HISMA for pr06, pr09, and pr10, respectively. HISMA-GL's normalized standard deviation is also higher than HISMA's for pr06 and pr10, while for pr09 it is slightly lower. All the runs for both the algorithms resulted in feasible solutions.

3) Scalability: For assessing how our HISMA performs on large scale problems, we extend two existing instances, i.e., pr05 and pr10, by adding 20% and 40% new customers. Each new customer is given a random co-ordinate, demand, and service duration, taken uniformly from their original range. The frequency and visit patterns are taken randomly from the existing values. Vehicle capacity and the route time constraints are increased by 20% and 40%. However, we keep m, u, and t intact. For comparison, we also create two smaller versions of both the instances by randomly removing 20% and 40% of the existing customers. The average results obtained by HISMA from 10 independent runs on these modified instances reveal that the cost is linear with the number of customers, n. The normalized standard deviation shows no such relationship with n and all the runs yield feasible solutions. Further investigation reveals that, the average cost of an edge in the solutions, decreases linearly with n. This is expected, as with increasing n there are more customers in the same euclidean space. This experiment thus suggests that our HISMA performs fairly similar with increasing problem size. The detailed results can be found in the Appendix and the modified instances from *https://github.com/azadsalam/HISMA_modified_instances*.

VI. Conclusion

VRP and its different variants have received intense interest from the research community for its important application in distribution networks. MDPVRP, a lesser known variant, has also received some attention lately. Being an NP-Hard problem, the meta-heuristic approaches have been commonly used in solving MDPVRP. However, only a few works employ memetic algorithms.

Designing a memetic algorithm for large combinatorial optimization problems like MDPVRP is difficult. Especially, balancing between exploration and exploitation to avoid premature convergence is challenging. We have introduced a simple memetic algorithm HISMA, which employs intelligent initialization and stochastic learning to avoid premature convergence for solving MDPVRPID, a variant of MDPVRP that allows interdependent operations among depots to reduce cost. HISMA has been tested extensively on the 10 benchmark instances provided by [10]. It has shown remarkably better results than the baseline methods and also the state-of-the-art methods which work under MDPVRP formulation. HISMA improved best results of seven out of the 10 benchmark instances.

In future, HISMA can further be improved by introducing a classification task to select the most promising solutions

TABLE IX: Comparison of among HISMA, HISMA-WH, and HISMA-WL. The columns labeled Avg and Nsd show the average and normalized standard deviations of the solution costs from 10 independent runs. The column Fsb indicates the percentage of feasible solutions obtained. The best results are shown in bold-face type.

Problem	HISMA			HISMA-WH			HISMA-WL		
Instance	Avg	Nsd	Fsb	Avg	Nsd	Fsb	Avg	Nsd	Fsb
pr06	**6602.85**	**0.0024**	**100**	6869.98	0.0236	**100.00**	106551.07	0.2072	0.00
pr09	**8223.37**	**0.0026**	**100**	8515.21	0.0076	**100.00**	11356.18	0.0155	**100.00**
pr10	**10015.21**	**0.0034**	**100**	10680.80	0.0124	**100.00**	15209.16	0.0178	**100.00**

TABLE X: Comparison between HISMA with greedy learning (HISMA-GL) and HISMA. Here, Avg and Nsd indicate the average and normalized standard deviations of 10 independent runs. All the experiment runs yielded feasible solutions, hence we omit the columns for feasibility percentage. The best results are shown in bold-face type.

Problem	HISMA		HISMA-GL	
Instance	Avg	Nsd	Avg	Nsd
pr06	**6602.85**	**0.0024**	6794.62	0.0038
pr09	**8223.37**	0.0026	8459.08	**0.0021**
pr10	**10015.21**	**0.0034**	10424.16	0.0050

for learning. Adopting niching techniques like [42] can improve HISMA too. In its current implementation, HISMA has two user specified parameters (e.g., τ and η), which can be made adaptive. Reducing the number of operators used by HISMA would be an interesting avenue to explore, too. Another fascinating direction will be to formulate MDPVRPID as a multiobjective optimization problem and solve it by algorithms like [43]- [45]. HISMA can also be extended to solve molecular optimization problems (e.g., [46], [47]).

REFERENCES

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

[2] G. Laporte, "Fifty years of vehicle routing," *Transportation Science*, vol. 43, no. 4, pp. 408–416, 2009.

[3] P. Toth and D. Vigo, *The vehicle routing problem*. Society for Industrial and Applied Mathematics, 2001.

[4] J.-F. Cordeau, G. Laporte, M. W. Savelsbergh, and D. Vigo, "Vehicle routing," *Transportation, handbooks in operations research and management science*, vol. 14, pp. 367–428, 2006.

[5] A. M. Campbell and J. H. Wilson, "Forty years of periodic vehicle routing," *Networks*, vol. 63, no. 1, pp. 2–15, 2014.

[6] J. R. Montoya-Torres, J. L. Franco, S. N. Isaza, H. F. Jiménez, and N. Herazo-Padilla, "A literature review on the vehicle routing problem with multiple depots," *Computers & Industrial Engineering*, vol. 79, pp. 115–129, 2015.

[7] E. Hadjiconstantinou and R. Baldacci, "A multi-depot period vehicle routing problem arising in the utilities sector," *Journal of the Operational Research Society*, pp. 1239–1248, 1998.

[8] P. Parthanadee and R. Logendran, "Periodic product distribution from multi-depots under limited supplies," *IIE Transactions*, vol. 38, no. 11, pp. 1009–1026, 2006.

[9] A. Rahimi-Vahed, T. G. Crainic, M. Gendreau, and W. Rei, "A path relinking algorithm for a multi-depot periodic vehicle routing problem," *Journal of Heuristics*, vol. 19, no. 3, pp. 497–524, Jun. 2013.

[10] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei, "A hybrid genetic algorithm for multidepot and periodic vehicle routing problems," *Operations Research*, vol. 60, no. 3, pp. 611–624, 2012.

[11] M. Tang, "A memetic algorithm for the location-based continuously operating reference stations placement problem in network real-time kinematic," *IEEE Transactions on Cybernetics*, vol. 45, no. 10, pp. 2214–2223, Oct 2015.

[12] C.-P. Chen, S. Mukhopadhyay, C.-L. Chuang, T.-S. Lin, M.-S. Liao, Y.-C. Wang, and J.-A. Jiang, "A hybrid memetic framework for coverage optimization in wireless sensor networks," *IEEE Transactions on Cybernetics*, vol. 45, no. 10, pp. 2309–2322, Oct 2015.

[13] R. Shang, Y. Wang, J. Wang, L. Jiao, S. Wang, and L. Qi, "A multi-population cooperative coevolutionary algorithm for multi-objective capacitated arc routing problem," *Information Sciences*, vol. 277, pp. 609 – 642, 2014.

[14] R. Shang, K. Dai, L. Jiao, and R. Stolkin, "Improved memetic algorithm based on route distance grouping for multiobjective large scale capacitated arc routing problems," *IEEE Transactions on Cybernetics*, vol. PP, no. 99, 2015.

[15] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "Implicit depot assignments and rotations in vehicle routing heuristics," *European Journal of Operational Research*, vol. 237, no. 1, pp. 15–28, 2014.

[16] F. B. de Oliveira, R. Enayatifar, H. J. Sadaei, F. G. Guimarães, and J.-Y. Potvin, "A cooperative coevolutionary algorithm for the multi-depot vehicle routing problem," *Expert Systems with Applications*, vol. 43, pp. 117–130, 2016.

[17] Z. Zhang, O. Che, B. Cheang, A. Lim, and H. Qin, "A memetic algorithm for the multiperiod vehicle routing problem with profit," *European Journal of Operational Research*, vol. 229, no. 3, pp. 573–584, 2013.

[18] J. Wang, Y. Zhou, Y. Wang, J. Zhang, C. Chen, and Z. Zheng, "Multiobjective vehicle routing problems with simultaneous delivery and pickup and time windows: Formulation, instances, and algorithms," *IEEE Transactions on Cybernetics*, vol. 46, no. 3, pp. 582–594, March 2016.

[19] M. Boudia, C. Prins, and M. Reghioui, "An effective memetic algorithm with population management for the split delivery vehicle routing problem," in *Hybrid Metaheuristics*. Springer, 2007, pp. 16–30.

[20] Y.-S. Ong, M.-H. Lim, N. Zhu, and K.-W. Wong, "Classification of adaptive memetic algorithms: a comparative study," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 1, pp. 141–152, Feb 2006.

[21] K. H. Kang, Y. H. Lee, and B. K. Lee, "An exact algorithm for multi depot and multi period vehicle scheduling problem," in *Computational Science and Its Applications–ICCSA 2005*. Springer, 2005, pp. 350–359.

[22] M. Mirabi, "A hybrid electromagnetism algorithm for multi-depot periodic vehicle routing problem," *The International Journal of Advanced Manufacturing Technology*, vol. 71, no. 1-4, pp. 509–518, 2014.

[23] N. Lahrichi, T. G. Crainic, M. Gendreau, W. Rei, G. C. Crişan, and T. Vidal, "An integrative cooperative search framework for multi-decision-attribute combinatorial optimization," CIRRELT, Tech. Rep. 42, Aug 2012.

[24] J.-F. Cordeau, G. Laporte, A. Mercier *et al.*, "A unified tabu search heuristic for vehicle routing problems with time windows," *Journal of the Operational research society*, vol. 52, no. 8, pp. 928–936, 2001.

[25] J.-F. Cordeau, M. Gendreau, and G. Laporte, "A tabu search heuristic for periodic and multi-depot vehicle routing problems," *Networks*, vol. 30, no. 2, pp. 105–119, 1997.

[26] R. Sarker, M. Mohammadian, and X. Yao, *Evolutionary optimization*. Springer, 2002, vol. 48.

[27] L. While and P. Hingston, "Usefulness of infeasible solutions in evolutionary search: An empirical and mathematical study," in *2013 IEEE Congress on Evolutionary Computation*, June 2013, pp. 1363–1370.

[28] R. L. Karg and G. L. Thompson, "A heuristic approach to solving travelling salesman problems," *Management Science*, vol. 10, no. 2, pp. 225–248, 1964.

[29] M. Gendreau, A. Hertz, and G. Laporte, "New insertion and postoptimization procedures for the traveling salesman problem," *Oper. Res.*, vol. 40, no. 6, pp. 1086–1094, Nov. 1992.

[30] M. Hutter, "Fitness uniform selection to preserve genetic diversity,"

in *Proceedings of the Congress on Evolutionary Computation, 2002 (CEC'02)*, vol. 1. IEEE, 2002, pp. 783–788.
[31] N. Krasnogor, J. Smith *et al.*, "A memetic algorithm with self-adaptive local search: Tsp as a case study." in *GECCO*, 2000, pp. 987–994.
[32] J. E. Mendoza, B. Castanier, C. Guéret, A. L. Medaglia, and N. Velasco, "A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands," *Computers & Operations Research*, vol. 37, no. 11, pp. 1886–1898, 2010.
[33] B. Bontoux, C. Artigues, and D. Feillet, "A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem," *Computers & Operations Research*, vol. 37, no. 11, pp. 1844–1852, 2010.
[34] D. Cattaruzza, N. Absi, D. Feillet, and T. Vidal, "A memetic algorithm for the multi trip vehicle routing problem," *European Journal of Operational Research*, vol. 236, no. 3, pp. 833–848, 2014.
[35] W. E. Hart, "Adaptive global optimization with local search," Ph.D. dissertation, University of California, San Diego, 1994.
[36] G. A. Croes, "A method for solving traveling-salesman problems," *Operations research*, vol. 6, no. 6, pp. 791–812, 1958.
[37] S. Lin, "Computer solutions of the traveling salesman problem," *Bell System Technical Journal, The*, vol. 44, no. 10, pp. 2245–2269, 1965.
[38] I. Or, "Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking," Ph.D. dissertation, Northwestern University, Evanston, Illinois., 1976.
[39] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach (3rd edition)*. Prentice Hall, 2009.
[40] D. R. Thompson and G. L. Bilbro, "Sample-sort simulated annealing," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 3, pp. 625–632, June 2005.
[41] G. W. Corder and D. I. Foreman, *Nonparametric statistics for non-statisticians: a step-by-step approach*. John Wiley & Sons, 2009.
[42] M. M. H. Ellabaan, Y. S. Ong, M. H. Lim, and K. Jer-Lai, "Finding multiple first order saddle points using a valley adaptive clearing genetic algorithm," in *Computational Intelligence in Robotics and Automation (CIRA), 2009 IEEE International Symposium on*, Dec 2009, pp. 457–462.
[43] L. Ke, Q. Zhang, and R. Battiti, "Moea/d-aco: A multiobjective evolutionary algorithm using decomposition and antcolony," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1845–1859, 2013.
[44] C. Liu, J. Liu, and Z. Jiang, "A multiobjective evolutionary algorithm based on similarity for community detection from signed social networks," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2274–2287, 2014.
[45] P. C. Roy, M. Islam, K. Murase, X. Yao *et al.*, "Evolutionary path control strategy for solving many-objective optimization problem," *IEEE Transactions on Cybernetics*, vol. 45, no. 4, pp. 702–715, 2015.
[46] M. Ellabaan, Y. S. Ong, S. D. Handoko, C. K. Kwoh, and H. Y. Man, "Discovering unique, low-energy transition states using evolutionary molecular memetic computing," *IEEE Computational Intelligence Magazine*, vol. 8, no. 3, pp. 54–63, Aug 2013.
[47] M. M. Ellabaan, S. D. Handoko, Y. S. Ong, C. K. Kwoh, S. A. Bahnassy, F. M. Elassawy, and H. Y. Man, "A tree-structured covalent-bond-driven molecular memetic algorithm for optimization of ring-deficient molecules," *Computers & Mathematics with Applications*, vol. 64, no. 12, pp. 3792–3804, Dec 2012.

Abdus Salam Azad received the B.Sc. in Computer Science and Engineering degree from Bangladesh University of Engineering and Technology (BUET), Bangladesh, in 2014. He is currently pursuing the M. Sc. in Computer Science and Engineering degree from the same. Currently, he is also a lecturer in the Department of Computer Science and Engineering at BUET.

His current research interests include computational intelligent and machine learning techniques and their application in text mining, machine comprehension and image processing.

Md. Monirul Islam (M'10) received the B. Sc Engineering degree from the Khulna University of Engineering and Technology (KUET), Bangladesh, in 1989, the M.Sc. Engineering degree from the Bangladesh University of Engineering and Technology (BUET), Bangladesh, in 1996, and the Ph.D. degree from the University of Fukui, Japan, in 2002. Currently, he is a Professor in the Department of Computer Science and Engineering at BUET. His major research interests include evolutionary robotics, evolutionary computation, neural networks, machine learning and data mining.

Saikat Chakraborty received the B.Sc. in Computer Science and Engineering degree from Bangladesh University of Engineering and Technology (BUET), Bangladesh, in 2014. He is currently pursuing the Ph.D. degree in the University of Virginia, Charlottesville, VA 22903, USA. His research interests include code analysis in software engineering and data mining.