Received July 20, 2020, accepted July 28, 2020, date of publication August 6, 2020, date of current version August 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3014736

New Discrete Cuckoo Search Optimization Algorithms for Effective Route Discovery in IoT-Based Vehicular Ad-Hoc Networks

HABEEB BELLO-SALAU[1], (Member, IEEE), ADEIZA JAMES ONUMANYI[2], (Member, IEEE), ADNAN M. ABU-MAHFOUZ[3,4], (Senior Member, IEEE), ACHONU O. ADEJO[2], (Member, IEEE), AND MUHAMMED BASHIR MU'AZU[1], (Member, IEEE)

[1]Department of Computer Engineering, Ahmadu Bello University, Zaria 810107, Nigeria
[2]Department of Telecommunication Engineering, Federal University of Technology Minna, Minna 920262, Nigeria
[3]Council for Scientific and Industrial Research (CSIR), Pretoria 0001, South Africa
[4]Department of Electrical Engineering, Tshwane University of Technology, Pretoria 0183, South Africa

Corresponding author: Adnan M. Abu-Mahfouz (a.abumahfouz@ieee.org)

This work was supported by the Council for Scientific and Industrial Research, Pretoria, South Africa, through the Smart Networks collaboration initiative and Internet of Things (IoT)-Factory Program (Funded by the Department of Science and Innovation (DSI), South Africa).

ABSTRACT Recently, the Internet of Things (IoT) is widely considered in vehicular ad-hoc networks (VANETs) for use in intelligent transportation systems. In particular, the pervasive deployment of different sensors in modern vehicles has unlocked interesting possibilities for improving routing performance in VANETs. Nevertheless, the discovery of short single loop-free routes for effective and efficient information dissemination in VANETs remains a challenge. This challenge proves more difficult to solve since it reduces to the case of finding the shortest Hamiltonian path for effective routing in VANETs. Consequently, in this paper, we propose two discretized variants of the cuckoo search optimization (CSO) algorithm, namely, the Lévy flight-based discrete CSO (LF-DCSO) and the random walk-based discrete CSO (RW-DCSO) for effective route discovery in VANETs. In addition, we investigated the inverse mutation operator gleaned from genetic algorithm (GA) in order to improve the exploration properties of our DCSO variants. We describe a new objective function that effectively models the reliability of individual links between nodes that comprise a single route. A detailed report of the routing protocol that controls the routing process is presented. Our proposed methods were compared against the roulette wheel-based GA and the improved k-means-based GA termed IGAROT. Specifically, our findings reveal that there was no significant difference in the performance of the different methods in the low vehicle density scenario, however, in the medium vehicle density scenario, the RW-DCSO algorithm achieved 2.56%, 100%, and 128.57% percentage increment in its route reliability score over the LF-DCSO, RW-GA, and IGAROT algorithms, respectively. Whereas in the high vehicle density scenario, the LF-DCSO algorithm achieved a percentage increment of 42.85%, 525%, and 733.33% in the route reliability score obtained over the RW-DCSO, IGAROT, and RW-GA algorithms, respectively. Such results suggest that our methods are able to guarantee effective routing in VANETs.

INDEX TERMS Discrete, cuckoo search optimization (CSO), route discovery, shortest path, VANET.

I. INTRODUCTION

Vehicular ad-hoc networks (VANETs) are mobile ad-hoc wireless networks created between vehicles, termed vehicle-to-vehicle (V2V) communication, or between vehicles and an infrastructure, termed V2I communication. In VANETs, each vehicle serves as a wireless router in order to communicate information from some source (transmitter) to some destination (receiver) within an average transmission radius of about 100 - 300 m [1]. Such communication links in VANETs for information transfer is made possible by the Internet of Things (IoT). Nowadays, most vehicles are deployed with

The associate editor coordinating the review of this manuscript and approving it for publication was Eyuphan Bulut.

a wide variety of sensors that measure and monitor several communication and physical parameters of a vehicle. For example, accelerometers are deployed to measure the velocity of a vehicle [2], Global Positioning System (GPS) sensors are embedded to determine a vehicle's location, and on-board sensor units are deployed for communication purposes. These sensors allow modern vehicles to easily initiate and maintain communication links. Thus, vehicles within the communication range of another transmitting vehicle are able to connect and in turn create a larger network of a wider range [1], [3]. The information often communicated via IoT-based VANETs (i.e either in V2V or V2I) are mostly related to traffic conditions [3]–[5], road surface conditions [2], [6], infotainment [1], [7], to name but a few. Essentially, IoT-based VANETs, which we simply refer to as VANETs hereafter, are geared toward safeguarding the lives and properties of road users while providing comfort to drivers and vehicle occupants.

Effective routing is key to the successful deployment of VANETs. However, information routing in VANETs is a difficult task to realize because of the highly volatile network topologies often created by constantly moving vehicles. Further, routing by broadcast methods in VANETs is an inefficient approach since network performance will ultimately deteriorate due to network congestion. Thus, it is pertinent to employ optimal reliable communication routes among multiple possible routes from some source to a destination node. To achieve optimal route discovery, a number of metaheuristic optimization algorithms (MOAs) have found great patronage in the literature.

MOAs are widely deployed in VANETs for their ability to find optimal routes within a wider solution-space of many possible routes. The list of available MOAs in the soft computing literature has grown exponentially in recent years. However, these different MOAs often yield varying performance levels for reasons such as the search strategy of the MOA, the effectiveness of the operators used in the MOA, as well as the robustness of the objective function deployed to describe the problem under consideration. Additionally, many existing MOAs are designed to solve continuous optimization problems and thus, they are not directly applicable to combinatorial optimization problems, which are discrete in nature, as obtained in VANETs. Some specific MOAs have been explored for routing purposes in VANETs (see Section II), however, they pose specific limitations that we seek to address in the present paper, such as, many of such articles often lack detailed reports about their adaptation process from the continuous to the discrete domain, which is required to solve combinatorial problems as obtained in VANETs. Further, it is pertinent to develop more robust and easily computable objective functions as explored in the present article. Furthermore, some effective MOAs are available, which have not been effectively explored for route discovery problems in VANETs.

Although the cuckoo search optimization (CSO) algorithm was considered in the present article, nevertheless, we do not claim that it is the best MOA in the literature for use in VANETs. Instead, we note that it has one of the least number of tunable internal parameters against many other MOAs in the literature, such as the genetic algorithm (GA) and the particle swarm optimization (PSO) algorithm. Specifically, apart from the general parameters, which are common to all MOAs, such as the population size and the stopping criteria, the CSO algorithm has only one extra tunable parameter, which is the probability of abandoning a nest. On the other hand, the GA and PSO algorithms both have a minimum of six and four internal parameters to be fine-tuned, respectively, which may limit their practical use under realtime conditions. This serves only as an initial justification for the use of the CSO algorithm in the present article, while further details with regards to the choice of the CSO algorithm are highlighted in Section II.

Thus, in this article, we address the aforementioned problems, hence leading to the following specific contributions to knowledge:

1) We propose two discrete variants of the cuckoo search optimization (CSO) algorithm, namely the Lévy flight-based discrete CSO (LF-DCSO) and the random walk-based discrete CSO (RW-DCSO). The CSO algorithm was adopted for use owing to its improved performance in solving different optimization problems [8]–[10]. Here, we describe in details the modification process introduced in order to codify both variants.
2) We introduce a new objective function to model the reliability of the individual links that comprise a single route from some source to some destination node. Our model considers both the signal-to-noise ratio (SNR) at a receiving node and the average velocity of mobile vehicles within the network to improve optimal route discovery.
3) We investigated the inverse mutation operator, gleaned from the genetic algorithm (GA), in order to improve the exploration property of our DCSO variants toward finding optimal routes.
4) Details of the routing protocol that initiates, discovers, and maintains candidate routes are presented. Comparison is made to demonstrate the validity of our DCSO variants as against the GA and IGAROT techniques [1]. We compared these methods under different vehicle density scenario and our methods were shown to provide improved performance.

The rest of the paper is structured as follows: Section 2 presents related work, our proposed routing algorithm and method of analysis is described in Section 3. While, Section 4 presents results and discussion. Conclusions are drawn in Section 5.

II. RELATED WORK

Several VANET routing protocols and their different topological classifications are discussed in [3]. The routing techniques in [3] were reviewed for different applications with regards to the mobility of vehicles typical of VANET systems. Similar surveys in [11], [12] have studied different VANET routing protocols and other bio-inspired classical concepts

deployed for vehicular routing. However, our present review focuses on examining different MOAs that have been used for routing purposes in VANETs. Specifically, we discuss the related works in terms of the evidences prior to our present study, and their limitations to be addressed in the present article:

A. EVIDENCE PRIOR TO THE PRESENT STUDY

Different routing algorithms have been proposed in order to determine the shortest and fastest path in V2V and V2I networks. An example can be found in [13], wherein a dynamic ad-hoc on-demand routing protocol based on the ant colony optimization (ACO) algorithm was proposed for VANETs. Experimental results show that the proposed protocol successfully withstood episodes of node disconnections during the communication process. Similarly, authors in [14] incorporated the ACO algorithm in the adhoc on-demand distance vector routing protocol (AODV) in order to determine the most stable route based on the simulated pheromone concentration of ants along the communication path. Other examples of an efficient greedy algorithm for transmitting data from some source to some destination can be found in [15], and the use of game theory to guarantee data dissemination in software-defined VANETs is discussed in [16].

A global exchange method involving the hybridization of a modified ant colony optimization (MACO) and particle swarm optimization (PSO) algorithm for reducing travel time in VANETs was presented in [17]. There, authors used the PSO algorithm to overcome the limitations associated with MACO, which includes frequent convergence to suboptimal solutions. As a consequence, the exploration properties of the ACO algorithm was thus improved, leading to the discovery of alternate routes, particularly when the optimal route becomes congested during peak hours. A similar approach that partitions VANETs into different zones using the ACO algorithm in order to determine reliable routes among multiple routes was presented in [18]. Authors used vehicle mobility, velocity, and fading conditions to develop a suitable communication model. They used the Nagakami probability distribution function (PDF) to model the dynamic reception of data packets, thus leading to better delivery ratios and reduced end-to-end delay with improved scalability far beyond four zones.

Other MOAs have been investigated in VANETs including the modified lion algorithm (LA) proposed in [19]. The performance of the modified LA was compared and shown to outperform the conventional GA and LA in terms of their complexity, cost, and convergence. Similarly, the grey wolf optimization (GWO) algorithm was explored in [20] for clustering purposes in VANET. The bees life algorithm (BLA) was proposed in [21] for optimization of the quality of service (QoS) in the multicast routing problem (QoS-MRP) in VANET with multiple objective functions. The proposed algorithm was shown to outperform the GA, bees algorithm (BA), and the marriage in honey bees optimization (MBO) algorithm. In a different article, authors in [22] used the firefly with levy distribution (FF-L) algorithm to determine optimal routes in VANETs. Experimental results show that the FF-L performed better than the GA, BA, BLA and MBO algorithms. Other MOAs such as the bacterial foraging optimization (BFO) [4], the tabu search process in GA [23], PSO [24], have been deployed with some success in VANETs.

In terms of the evidence of discretized MOAs deployed in VANET, authors in [25] investigated discrete versions of the bat, firefly, CSO, and PSO algorithms. The uniform cross over and swapping functions were adopted with results demonstrating better performances by the bat, firefly, and PSO algorithms. Similarly, the PSO algorithm was also discretized in [26], termed the DPSO, in order to determine optimal communication paths in VANET. Here, authors modelled the link stability upon the computed Euclidean distance in polar coordinates, and the fitness function was based on the probability of occurrence of obstacles along the communication path. Simulation results show improved performance in terms of packet delivery ratio, average throughput, and routing overhead compared to other routing protocols such as the QADD and GPSR. Interestingly, in other application areas, it is noted that an improved quantum-behaved PSO algorithm was developed for end member extraction in [27]. This approach was further extended to solve the multiobjective hyperspectral end member extraction problem [28]. A linear mixture model constrained PSO was again developed in [29] to solve the problem of end member extraction from highly mixed data. Indeed, these methods are considered noteworthy advancements in the development of discretized MOAs.

B. GAPS IN PRIOR STUDIES

Essentially, most MOAs are designed to solve continuous optimization problems, thus, it is pertinent to accurately describe the adaptation as well as the codification process of these continuous-based MOAs to their discrete versions, which is the form required to solve combinatorial problems in VANETs. However, with an exception to the approaches in [25], [26], most articles mentioned above lack the required details needed to understand the discretization processes used in their respective implementations, which limits their use and stifles the replication of these methods for further investigations.

The details of the DCSO algorithm adopted in [25] leaves much to be desired since authors simply mentioned that the uniform crossover and swapping functions were used, without providing sufficient details with regards to its implementation process. Instead, authors simply mentioned that they adopted the DCSO version proposed in [8], which was designed to address the travelling salesman problem (TSP). While the approach in [8] adopted a 2-opt and double-bridge operator to introduce new solutions, authors in [25] used the uniform crossover and swap operators, which may have been responsible for the contradictory report of a poorer DCSO performance in [25] as against the improved performance of the DCSO algorithm reported in [8]–[10]. Further, the DCSO

variant reported in [25] performed least in comparison to the bat, firefly, and PSO algorithms, which contradicts reports in [9] about the superior performance of the DCSO algorithm against other well-known protocols. Thus, such a poor performance of the DCSO algorithm as reported in [25] leaves much room for improvement since incidentally, authors claimed that the same operators, i.e the uniform crossover and swapping functions were used in all the MOAs compared therein. On the other hand, details of the implementation procedure of the DCSO algorithm used in [9] were conspicuously missing. Consequently, we explore in the present article the inverse mutation operator, which is a more robust type of swapping operator as against the uniform crossover and single swapping operators used in [25]. In effect, we present an entirely new approach to realizing the DCSO, which differs from the approach in [25].

The Hamming distance function was used in [25], whereas the Euclidean distance function was used in [9] with contrasting findings, thus opening an opportunity for further investigation as conducted in the present paper. Additionally, we explore the discovery of the shortest path between an infrastructure and all other vehicles in a VANET, which differs from the cluster-based approach in [25], which we consider to be a more complex approach to routing in VANET. Finally, we introduce a new objective function that differs from prior works in the literature, with the aim to better model the link reliability between transmitting and receiving nodes.

III. DESCRIPTION OF THE PROBLEM

In VANETs, communication takes place in two directions: in the downlink direction, which comprises information dissemination from an infrastructure to vehicles (I2V), and/or in the uplink direction, which transpires from vehicles to an infrastructure (V2I). Such bidirectional communication could transpire in an ad-hoc architecture as well, which concerns communication between vehicles, termed V2V. However, in this article, we refer to the communication between any infrastructure and a set of vehicles as V2I communication irrespective of the direction of communication. Here, an infrastructure may refer to a roadside base station or even a facility (a building) with rooftop antennas connected to servers. Thus, our problem reduces to the case of discovering the most effective and efficient route for information dissemination in V2I networks.

Our aim is to develop a routing algorithm that establishes a globalized, loop-free, single route between all nodes in a VANET. This is a non-trivial routing task owing to the volatile topological nature of VANETs, as well as the possibility for high vehicular densities in VANETs, which complicates and increases the computational complexity of discovering the best routes. We consider the case of push messages, wherein an infrastructure seeks to disseminate alert messages to all vehicles in a VANET, for example, to inform vehicles about a recent accident scene, traffic congestion on certain roads, insecurity conditions, road anomalies along a road, to name but a few. In this case, a broadcast (i.e flooding) approach may be ineffective, since the network may become easily congested and completely jammed, particularly in the case of long and many messages. Thus, the most scalable option is to discover a loop-free, single route from the infrastructure to all vehicles in the network. We shall discuss in the next subsection the routing protocol that initiates, discovers, and maintains such a candidate route. It is noted that the terms ''route'' and ''path'' are used interchangeably in the rest of this article.

From the viewpoint of graph theory, our problem can be described as the case of finding the shortest Hamiltonian path, which has the maximum route reliability score in a rooted graph. The function that computes the route reliability score for each discovered Hamiltonian path will be discussed in Section V-A. However, the rooted graph $G(V, E)$ has a source node (i.e. an infrastructure), which is distinguished as the root, whereas all other nodes in the VANET (i.e the graph) are the vehicles. In particular, V is the set of vertices (called the nodes or vehicles in the VANET, which includes the infrastructure), and E is a set of edges (i.e. the communication links) connecting different ordered pairs of vertices.

Essentially, since we attempt to solve the problem of finding the shortest Hamiltonian route, which is NP-hard, thus, the complexity of finding the solution grows as $(N_V - 1)!$, where N_V is the total number of vehicles in the network. Thus, it is noted that searching through such a huge solution space of possible routes for the shortest Hamiltonian path is a non-trivial task in VANETs, especially as N_V continues to grow exponentially. Hence, being NP-hard, such a route discovery problem cannot be efficiently solved in realtime by most sequential-based search methods, or even by localized greedy-based approaches. Consequently, we leverage metaheuristic optimization algorithms (MOA), which are population-based methods, in order to search for optimum loop-free, single routes in VANETs.

Additionally, most MOAs in the soft-computing literature are designed to solve continuous-based optimization problems. Hence, they are not directly applicable to the discrete-based graph problem as in VANETs. Thus, we consider the problem of discretizing a candidate MOA, chosen here to be the cuckoo search optimization (CSO) algorithm. We shall describe in subsequent subsections our discretized approach of the CSO for use in VANETs. Summarily, the above problems of discovering the candidate Hamiltonian route in VANETs and the discretization of the CSO algorithm are the specific focus of the present article.

IV. THE ROUTING PROTOCOL

We describe the routing protocol that initiates, discovers, and maintains a loop-free single route in V2I networks. The method for route discovery depends on the newly proposed DCSO variants and it will be described in the next section. The routing protocol comprises three phases namely, the initialization, route discovery, and route maintenance phase. They are discussed as follows:

1) THE INITIALIZATION PHASE

The initialization phase describes how vehicles get connected and registered to an infrastructure in a VANET. A simple model to describe the initialization phase is depicted in Fig. 1. Since the address of the infrastructure is not known at initialization, thus, any vehicle intending to access a VANET sends a broadcast message containing a request to connect (RTC) packet to an infrastructure, i.e. the nearest infrastructure. It is assumed that only vehicles moving at below 20 km/hr would be able to access the network. Results in this regard are presented in Section VI. This ensures that a more stable topology can be maintained in VANETs. The broadcast range of each vehicle is represented by the circles around each node in Fig. 1. The RTC packet is continuously rebroadcasted by each vehicle until it reaches the infrastructure. As gleaned from Fig. 1, the initialization phase can be a heavily congested phase, particularly if many vehicles are requesting access at about the same time. However, the infrastructure S sequentially registers each vehicle in an order in which they arrive, thus, sequentially populating and maintaining an efficient routing table. An example of the essential fields of an RTC packet is illustrated in Fig. 2.

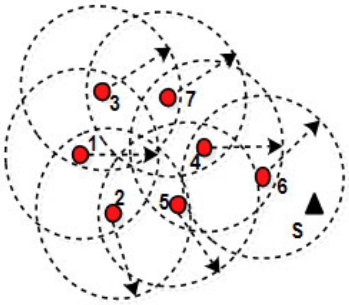


FIGURE 1. Initialization phase of the routing protocol. Red dots represent vehicles, the black triangle *S* denotes the infrastructure, the arrows are the transmission range and links.

RTC Packet Fields	
GPS Coordinate	
Average Vehicle Speed	
RSS	
Appended GPS Coordinate	

FIGURE 2. Essential fields of the RTC packets and discovered route packets.

The RTC packet sent from the requesting vehicle will contain its GPS coordinates and its average speed (in km/hr). However, before rebroadcasting the packet, the nearest re-forwarding vehicle(s) would append to the RTC packet the received signal strength (RSS) value measured from the requesting vehicle. Consequently, this ensures that only the nearest vehicle(s) one-hop away from the requesting node are able to update the RSS field of the RTC packet since the RSS field will be found empty at this point. Simultaneously, the GPS coordinates of the appending vehicle (i.e the vehicle that appends the RSS value) are included in the "Appended GPS Coordinate" field. This initialization process can be better understood by considering the simple model of Fig. 1. Let us assume that vehicle 1 in Fig. 1 is the requesting vehicle, then only vehicles 2 and 3 will receive the broadcast RTC packet since they are the only cars within the broadcast range of vehicle 1. At this point, vehicles 2 and 3 would check the RSS field of the RTC packet and they would append their independently measured RSS value to the RSS field since they would find the field empty at this point. Then, they would rebroadcast the updated RTC packet. Now, vehicle 5 receives the RTC packet from vehicle 2 and simply rebroadcasts without altering the RSS field since it is already occupied, and likewise with vehicle 7, which received from vehicle 3. Thus, henceforth, every other vehicle that receives the RTC packet simply rebroadcasts as long as the RSS field is occupied. For example, vehicle 4 would rebroadcast two different RTC packets with different RSS field values from both vehicles 5 and 7. Consequently, the RTC packet from vehicle 1 arrives at the infrastructure S along with the details of the measured RSS values from only vehicles 2 and 3. Hence, the infrastructure is able to obtain only the closest vehicles to vehicle 1 and consequently build an approximate map of its network topology.

The above process enables the infrastructure to construct an accurate position of each vehicle in its network. However, since the network topology may change rapidly in VANETs, this process would continue for every new vehicle that emerges in the network. Nevertheless, the computational and convergence demands of the route initialization phase would reduce over time, since different infrastructures would be networked to pass routing tables amongst themselves toward accelerating and improving network convergence and stable tracking of vehicles.

2) THE ROUTE DISCOVERY PHASE

The infrastructure S executes the route discovery phase following the completion of the route initialization phase from which an accurate network topology is constructed. For the case of push messages in the downlink mode, the infrastructure computes the most effective and efficient route through all vehicles in the network. To achieve this, the infrastructure executes our newly proposed DCSO variants (to be discussed in Section V-C) in order to discover the best loop-free, single route. For example, Fig. 3 illustrates the discovered route based on the node deployment in Fig. 1 that prevents any further use of flood messages in the network. In this case, the Hamiltonian route is S-6-4-7-3-1-2-5, whereas an example of a link (i.e. an edge) is $\{6, 4\}$, where $i = 6$ and $j = 4$. Thus, several links make up a route, and a single route is formed by a set of edges. This approach improves network efficiency as well as scalability.

3) THE ROUTE MAINTENANCE PHASE

Since network topologies change frequently in VANETs, it is essential to discuss how such loop-free, single routes are maintained. In this case, once a route has been discovered and

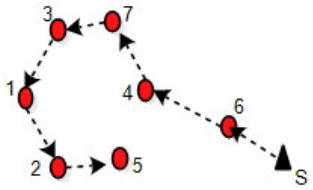


FIGURE 3. Route discovery phase of the routing protocol. Red dots represent vehicles, the black triangle *S* denotes the infrastructure, the arrows are the transmission range and links.

information has been disseminated, the infrastructure continues to check if new vehicles have been added or discarded from the routing table. For newly added vehicles via the initialization phase, the route discovery phase is re-initiated to compute a new route. This update can be conducted periodically to obtain the updated positions of older vehicles in the routing table. On the other hand, once an acknowledgement message is not received by the infrastructure from any vehicle, such vehicle(s) is/are expunged from the routing table and a new route is computed via the route discovery phase. This process ensures that loop-free, single routes are maintained.

V. THE DISCRETE CUCKOO SEARCH OPTIMIZATION ALGORITHMS

In this section, prior to describing our proposed discrete CSO variants, first, we describe the objective function deployed in our methods. Then, we describe the original CSO algorithm as a complete basis for introducing our discretized variants. Then, the details of our proposed variants are presented.

A. THE ROUTE RELIABILITY FUNCTION

We describe a new objective function that measures the reliability of the wireless communication link between any two communicating vehicles. This function is referred to as the route reliability function R. In this regard, two important factors that characterize the reliability of a link are considered in our function, which are the velocity ϑ of each vehicle and the signal-to-noise-ratio (SNR) γ at each receiving vehicle. Since most modern vehicles are designed with on-board processing units, the velocity of each vehicle can be obtained and reported in the RTC packet, whereas the γ at each receiving vehicle j relative to the signal transmitted from vehicle i is computed as

$$\gamma_{i,j} = \frac{RSS_{i,j}}{\sigma_n^j}, \tag{1}$$

where $RSS_{i,j}$ is the received signal strength measured at a receiver vehicle j relative to a signal transmitted from vehicle i, and σ_n^j is the noise level at vehicle j.

Thus, we model the reliability r of the link between any two vehicles i, j using the cumulative probability distribution function of an exponential random variable as

$$r_{i,j} = 1 - \exp\left(\frac{-\gamma_{i,j}^{dB}}{\vartheta_j + 1}\right), \tag{2}$$

for $\gamma_{i,j}^{dB} > 0$, where $\gamma_{i,j}^{dB}$ is the SNR dB equivalent of (1) obtained as $\gamma_{i,j}^{dB} = 10\log_{10}\gamma_{i,j}$, and ϑ_j is the average velocity of the vehicle j. It is seen in (2) that if the SNR value is increased while keeping the velocity fixed, then the exponential function value decreases toward zero. Thus, by decreasing toward zero, the entire reliability function converges asymptotically toward one, which implies that the link is becoming more reliable as desired. On the other hand, let us consider the case where the SNR is kept fixed, whereas the velocity is increased. Again, it is noted that the exponential function increases toward one, thus, causing the reliability function to decrease toward zero, which implies that the link is becoming an unreliable link as expected. For the converse case, the reliability function would typically decrease toward zero if either of the SNR or the velocity value is reduced, thus justifying the workability of the proposed model.

Recall that both the SNR and velocity values for each vehicle are easily obtained since the respective $RSS_{i,j}$ values of each vehicle are acquired as described in Section IV-1. Further, the reliability function $r_{i,j}$ produces values bounded between 0 and 1, thus, it reduces to a simple probability score of the link reliability between two communicating vehicles. Thus, a link is considered to be perfectly reliable when $r_{i,j} = 1$ and totally unreliable (i.e. non-existent) when $r_{i,j} = 0$. A link becomes totally unreliable if $\gamma_{i,j}^{dB} \leq 0$, which means the transmitted signal power is less than the receiver noise level and thus cannot be processed, whereas $r_{i,j} = 1$ is obtained only for $i = j$, which describes the case of a looped link or the distance d between i and j being zero ($d_{i,j} = 0$). Hence, since we aim to discover only loop-free routes and vehicles cannot be physically overlapped to obtain $d_{i,j} = 0$, thus $i \neq j$. Having established these facts, we obtain the total reliability R of a single route k, which comprises many links, as

$$R^{(k)} = \prod_{i,j} r_{i,j}, \tag{3}$$

for $k = 1, 2, 3, \ldots, K$, $i = j = 1, 2, 3, \ldots, N_V$, and $i \neq j$, where N_V is the total number of vehicles registered in the VANET. Thus, the best loop-free, single route R^* is obtained from among so many possible routes K as

$$R^* = \max_{k \in K} R^{(k)}, \tag{4}$$

where $K = (N_V - 1)!$. It is noted that searching through K in factorial space for an optimal solution is an NP-hard problem. However, it can be further considered to be NP-complete since it can be reduced to the special case of solving the travelling salesman problem, which is shown to be NP-complete in [30]. We shall present in Section V-C two new discrete variants of the CSO metaheuristic algorithm designed to search for R^*.

B. THE CONTINUOUS-BASED CSO

Prior to presenting our DSCO variants, it is instructive to discuss the original continuous-based CSO algorithm. A brief

summary is provided in this subsection. The CSO algorithm was proposed in [31] to model the brood parasitic nature of cuckoo birds as they aim to ensure that their eggs are hatched by other host birds. The success of this nature-inspired process served to motivate the construction of the CSO algorithm for optimization purposes. In this regard, the metaphoric relationship of terminologies in the CSO algorithm to known terms in general optimization parlance is given as follows: an egg or a nest represents an individual solution in the case of solving single objective function problems. In solving multi-objective function problems, an individual (nest) could contain multiple solutions (i.e. multiple eggs). However, we consider the single objective function problem in this article. A collection of nests corresponds to the total population of candidate solutions. The process of abandoning a nest by a foreign bird, i.e. the case where a host bird discovers the cuckoo's egg, refers to the process of discarding a poor solution or a set of poorer solutions from the population of solutions, whereas, the laying of a new egg(s) by a cuckoo in a nest or in several nests refers to introducing new solution(s) to the population.

Essentially, the continuous-based CSO algorithm finds new and better solutions using [31]

$$x_p^{(t+1)} = x_p^{(t)} + \alpha \otimes L\dot{e}vy(\lambda), \tag{5}$$

where $x_p^{(t+1)}$ denotes a new solution for a cuckoo p, obtained in a new iteration $t+1$ from an older solution $x_p^{(t)}$ obtained in the previous iteration t. The Lévy flight distribution function $L\dot{e}vy(\lambda)$ is used to update older solutions, where λ is the Lévy walk parameter, α is the step size related to the scale of the problem of interest, and the symbol $\otimes$ means entry wise multiplication. The Lévy flight function provides a random walk and the random step length is drawn from a Lévy distribution generally approximated using the Mantegna algorithm as

$$L\dot{e}vy(\lambda) \sim \frac{u}{v^{-\lambda}}, \tag{6}$$

where u and v are drawn from normal distributions defined as

$$u \sim N(0, \sigma_u^2), \tag{7}$$

$$v \sim N(0, \sigma_v^2), \tag{8}$$

where

$$\sigma_u^2 = \frac{\Gamma(1+\lambda) * \sin(\frac{\pi\lambda}{2})}{\Gamma(\frac{1+\lambda}{2}) * \lambda * 2^{(\frac{\lambda-1}{2})}}, \tag{9}$$

$$\sigma_v^2 = 1, \tag{10}$$

where Γ is the Gamma function and $1 < \lambda \leq 3$.

The summary of the steps in the CSO algorithm are as follows:

1) Configure the values of all parameters, including the probability of abandoning nest P_a, the population size N_P, and the stopping criterion, as well as the parameters of the problem to be solved.
2) **Initialization stage**: Generate initial population of solutions
3) **while** stopping criterion is not met **do**
   a) **Getcuckoo stage**: Get new solutions using the Lévy flight walk according to the Mantegna algorithm through (6) - (10).
   b) Evaluate the new solutions to obtain the global best solution
   c) **Empty nest stage**: Discard the poorest solutions based on P_a and replace them with new solutions
   d) Re-evaluate the new population of solutions to obtain new global best solution
4) Continue iteration until stopping criterion is satisfied.

Since the solution x can take on any continuous value along the real number line, thus, the present form of the CSO algorithm is not applicable to problems whose solutions take on only discrete values, particularly integer values. Consequently, we propose two new discretized versions of the CSO algorithm in order to solve the route discovery problem in VANETs.

C. THE PROPOSED DCSO

Two DCSO approaches are proposed, namely, the Lévy flight-based method, wherein we strive to maintain the steps as obtained in the original CSO algorithm, and the random flight-based method. The random flight-based method is proposed as a faster option as against the Lévy flight method. Following from Section V-B, we note that there are three main stages in the CSO algorithm, namely, the initialization, getcuckoo, and empty nest stages, and we describe our methods along these stages.

1) PARAMETER DESCRIPTION

We describe the different parameters required to execute our proposed DCSO route discovery methods. Since the infrastructure is able to obtain the SNR $\gamma_{i,j}$ between any two communicating vehicles i and j in the network, thus, the route reliability score $r_{i,j}$ of a link between any pair of communicating vehicles can be computed using (2) for $i = j = 1, 2, 3, \ldots, N_V$ and $i \neq j$.

The population size N_P and the probability of abandoning a nest P_a are the two main parameters to be configured *a priori* by users of our DCSO methods. The stopping criterion of the proposed DCSO algorithms is the total number of fitness evaluations N_{FE}, which ensures that all MOAs are fairly compared without presenting undue advantage to any MOA that deploys extra computational stages in its search strategy. Here, the term fitness function and objective function are used interchangeably to refer to the route reliability function of (2). Further classical details with regards to the fair comparison of MOAs based on the number of fitness function consumed can be found in [32], [33]. Thus, at the start of the algorithm, the iteration counter I_C is initialized as $I_C \leftarrow 0$ and incremented $++I_C$ each time the fitness function is evaluated. Hence, the algorithm stops once $I_C > N_{FE}$.

2) INITIALIZATION STAGE

The initialization stage applies to both DCSO variants. Here, both algorithms begin by constructing a population of randomly generated solutions. A single solution (i.e. an egg) in a nest is a single loop-free Hamiltonian path from the infrastructure S through every indexed vehicle (vertices) in the network. An example of a single solution is depicted in Fig. 4(a). The example in Fig. 4(a) depicts a network with seven nodes (i.e including the infrastructure and 6 other vehicles), thus, $N_V = 7$ in this case. Hence, a single solution in this example is a single loop-free route given as S-v4-v6-v1-v2-v5-v3. A population of solutions is illustrated in Fig. 4(b). In this case, several randomly generated Hamiltonian routes are obtained yielding a population matrix of size $N_P \times N_V$.

1	2	3	.	.	.	N_V
S	v4	v6	v1	v2	v5	v3

(a) A single solution

	1	2	3	.	.	.	N_V
1	S	v4	v6	v1	v2	v5	v3
2	S	v6	v2	v5	v3	v4	v1
3	S	v5	v3	v4	v6	v1	v2
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
N_P	S	v2	v3	v6	v4	v5	v1

(b) Population of solutions

FIGURE 4. Representation of a single route and the case of multiple possible single routes.

We shall refer to a single solution (i.e a route) as $V^{(k)}$, which is a set of vertices (i.e vehicles) that make up the route, thus, $V^{(k)} = \{v_1, v_2, v_3, \ldots, v_{N_V}\}$ and its corresponding route reliability score is denoted as $R^{(k)}$ computed using (3). Thus, the population of route scores R_P is a vector stated as

$$R_P = \left\{R^{(k)}\right\} \tag{11}$$

for $k = 1, 2, 3, \ldots, N_P$. The algorithm for the initialization stage is presented in Alg. 1. Since our problem pertains to maximizing the route reliability score that corresponds to the best route, thus, step 1 of Alg. 1 initializes the best route reliability score as $-\infty$ so that any subsequent score will always be greater this initial value. Each step of Alg. 1 contributes toward generating an initial population of solutions, which will be further iterated in subsequent stages. We note that the use of the *randperm* function in step 6 of Alg. 1 ensures that no vehicle index (i.e element) is duplicated in a solution (i.e individual). This is ensured since the *randperm*(N_V) function simply draws N_V random numbers between 1 and N_V without replacement. By not replacing the numbers that have been drawn, we avoid the problem of duplicating elements within an individual.

Algorithm 1 Initialization Stage of DCSO Algorithms

Require: Population size N_P, Number of vehicles registered in the VANET N_V

Ensure: Initial population of routes V_P, Initial population of route reliability scores R_P, Best route V^*, Best route reliability score R^*

1: $I_C \leftarrow 0$ *Initialize the fitness evaluation counter*
2: $R^* \leftarrow -\infty$
3: $V^* \leftarrow \emptyset$
4: $R_P \leftarrow \emptyset$
5: $V_P \leftarrow \emptyset$
6: **for** $k = 1$ to N_P **do**
7: Use a random permutation function denoted as *randperm*(N_v) to generate an individual route $V^{(k)}$ (as illustrated in Fig. 4(a)). The route reads $V^{(k)} =$ *randperm*(N_v). A simple Fisher-Yates shuffle algorithm can be deployed for this purpose.
8: Compute the reliability score $R^{(k)}$ of this route using (3)
9: $++I_C$ *Update fitness evaluation counter*
10: $R_P \leftarrow \left\{R_P, R^{(k)}\right\}$ *Save the route reliability score*
11: $V_P \leftarrow \left\{V_P, V^{(k)}\right\}$ *Save the route*
12: **if** $R^{(k)} > R^*$ **then**
13: $R^* \leftarrow R^{(k)}$ *Update best route reliability score*
14: $V^* \leftarrow V^{(k)}$ *Update best route*
15: **end if**
16: **if** $I_C > N_{FE}$ **then**
17: Terminate the algorithm
18: **return** R^*, V^*
19: **end if**
20: **end for**
21: **return** R^*, V^*

3) GETCUCKOO STAGE: THE Lévy FLIGHT APPROACH

The population of solutions obtained in the initialization stage is passed to the getcuckoo stage. The getcuckoo stage relates to the laying of eggs by a cuckoo in different nests within the population. Technically, this implies replacing initial solutions with new solutions. The original CSO algorithm achieves this stage using the Lévy flight approach, which we aim to replicate in our discrete version. The use of the Lévy distribution to generate step-sizes in order to update older solutions (recall (5)) is the hallmark of the CSO algorithm. The Lévy distribution yields both short and long step lengths owing to its long-tailed distribution, thus, it enables the CSO algorithm to jump out of possible local optimal solutions toward finding the global optimal solution.

Alg. 2 describes the getcuckoo stage based on the Lévy distribution, which we term the LF-DCSO approach. It is noted that the entire output of the initialization stage becomes the input to the getcuckoo stage. The standard parameter values required to generate random values from the Lévy distribution are stated in steps 1 - 3. New solutions are generated via steps 5 - 23. The Lévy flight step value is computed in

Algorithm 2 Getcuckoo Stage: For the LF-DCSO Approach

Require: Initial population of routes V_P, Initial population of route reliability scores R_P, Best route V^*, Best route reliability score R^*, Lévy walk parameter λ

Ensure: New population of routes V_P, New population of route reliability scores R_P, Best route V^*, Best route reliability score R^*

1: $\lambda \leftarrow \frac{3}{2}$
2: $\sigma_v^2 \leftarrow 1$
3: $\alpha \leftarrow 0.01$
4: Compute σ_u^2 using (9).
5: **for** $k = 1$ to N_P **do**
6: Generate u using (7) and v using (8)
7: Compute the Lévy flight step $L\dot{e}vy(\lambda)$ as $L\dot{e}vy(\lambda) = \lceil \frac{u}{v^{\frac{1}{\lambda}}} \rceil$
8: Compute the distance D between the present route $V^{(k)}$ and the best route V^*. This can be realized using either the **Euclidean** or **Hamming** distance function.
9: Compute the step-size h as $h = \lceil \alpha * D * L\dot{e}vy(\lambda) \rceil$
10: Apply the **inverse mutation operator** f_{INV} to obtain a new solution (i.e a new route) as $V^{(k)} \leftarrow f_{INV}(V^{(k)}, Q1, Q2, h)$. The details of this operator are explained via (12).
11: Compute the reliability score $R^{(k)}$ of this new route using (3)
12: $++I_C$ *Update fitness evaluation counter*
13: $R_P \leftarrow \{R_P, R^{(k)}\}$ *Save the new route reliability score*
14: $V_P \leftarrow \{V_P, V^{(k)}\}$ *Save the new route*
15: **if** $R^{(k)} > R^*$ **then**
16: $R^* \leftarrow R^{(k)}$ *Update best route reliability score*
17: $V^* \leftarrow V^{(k)}$ *Update best route*
18: **end if**
19: **if** $I_C > N_{FE}$ **then**
20: Terminate the algorithm
21: **return** R^*, V^*
22: **end if**
23: **end for**
24: **return** R^*, V^*

step 7. The ceiling function $\lceil \bullet \rceil$ is used to ensure that integer values are obtained, since we aim to update discrete values. In step 8, we compute the distance D, which is a measure of how far a solution differs from the current best route. This distance measure can be estimated using either the Euclidean or Hamming distance function. It is noted that small values of D would be obtained for solutions that are close to the present best solution, which yields smaller step-size values in step 9. These smaller step-size values invariably improve the exploitation properties of the LF-DCSO algorithm. Whereas, for larger values of D, larger step-size values are obtained, which enhances the exploration properties of the LF-DCSO algorithm. Consequently, a new solution is obtained in step 10 via the use of our proposed inverse mutation operation described as follows:

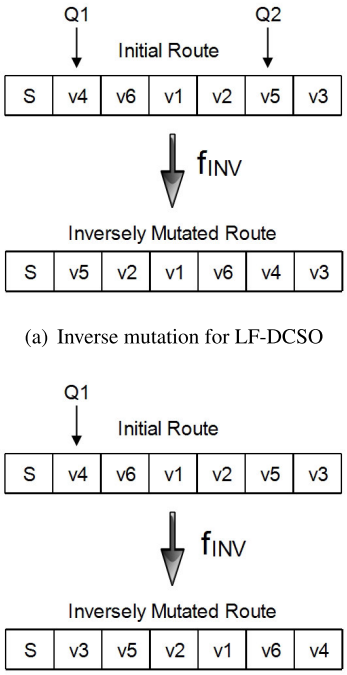


FIGURE 5. Inverse mutation operation for the LF-DCSO and RW-DCSO algorithms.

- **Inverse Mutation Operation for the LF-DCSO:** The inverse mutation operator f_{INV} works by reversing a sequence of vertices in a route starting from a first point of inversion $Q1$ to a stopping point $Q2$ (see the illustration in Fig. 5(a)). The point $Q1$ is a random number drawn from a discrete uniform distribution $\mathcal{U}$ as

$$Q1 \sim \mathcal{U}\{a, b\}, \tag{14}$$

where $a = 2$ and $b = N_V$. The value $a = 2$ is used to ensure that the infrastructure S remains rooted at index 1 during the operation such that inversion can only take place from index 2. Thus, once $Q1$ is drawn, $Q2$ can be

$$f_{INV}(V, Q1, Q2, h) = \begin{cases} \{v_1, \ldots, v_h, v_b, v_{b-1}, v_{b-2}, v_{b-h+1}\}, & \text{if } Q2 < 1 \\ \{v_1, \ldots, v_{b-h}, v_b, v_{b-1}, v_{b-2}, v_{b-h+1}\}, & \text{if } Q2 > N_V \\ \{v_1, \ldots, v_{Q1}, v_{Q2}, v_{Q2-1}, v_{Q2-2}, \ldots v_{Q1+1}, v_{Q2+1}, v_{Q2+2}, \ldots, v_b\}, & \text{if } 1 \le Q2 \le N_V \end{cases} \tag{12}$$

$$f_{INV}(V, Q1) = \{v_1, v_2, \ldots, v_{Q1-1}, v_b, v_{b-1}, v_{b-2}, v_{Q1}\} \tag{13}$$

Algorithm 3 Getcuckoo Stage: For the RW-DCSO Approach

Require: Initial population of routes V_P, Initial population of route reliability scores R_P, Best route V^*, Best route reliability score R^*

Ensure: New population of routes V_P, New population of route reliability scores R_P, Best route V^*, Best route reliability score R^*

1: **for** $k = 1$ to N_P **do**
2: Apply the **inverse mutation operator** f_{INV} in (13) to obtain a new solution (i.e a new route) as $V^{(k)} \leftarrow f_{INV}(V^{(k)}, Q1)$.
3: Compute the reliability score $R^{(k)}$ of this new route using (3)
4: $++I_C$ *Update fitness evaluation counter*
5: $R_P \leftarrow \{R_P, R^{(k)}\}$ *Save the new route reliability score*
6: $V_P \leftarrow \{V_P, V^{(k)}\}$ *Save the new route*
7: **if** $R^{(k)} > R^*$ **then**
8: $R^* \leftarrow R^{(k)}$ *Update best route reliability score*
9: $V^* \leftarrow V^{(k)}$ *Update best route*
10: **end if**
11: **if** $I_C > N_{FE}$ **then**
12: Terminate the algorithm
13: **return** R^*, V^*
14: **end if**
15: **end for**
16: **return** R^*, V^*

obtained as

$$Q2 = Q1 + h \tag{15}$$

where h is the step-size computed in step 9 of Alg. 2. Consequently, for any route $V = \{v_1, v_2, v_3, \ldots, v_b\}$, where $b = N_V$, the general function for the inverse mutation operator for the LF-DCSO algorithm is given in (12), as shown at the bottom of the previous page.

The route reliability score is then computed in step 11 of Alg. 2 for the new route generated using the inverse mutation operator in step 10. The fitness function evaluation counter is updated in step 12, and updates are conducted accordingly in steps 13 - 14. The global best route reliability score is updated between steps 15 - 18, and the stopping criterion is checked in steps 19 - 22. Then, the new population of solutions and their corresponding route reliability scores are outputted to the next stage.

4) GETCUCKOO STAGE: THE RANDOM FLIGHT APPROACH

It is obvious that the getcuckoo stage for the LF-DCSO algorithm in Alg. 2 involves a number of computations, which increases its complexity. Thus, we propose a simpler and faster approach termed the random flight/walk-based approach for the getcuckoo stage of the DCSO (RW-DCSO) algorithm. Essentially, we used the discrete uniform distribution to determine the first point of inversion $Q1$ and then we apply the inverse mutation operation as follows:

Algorithm 4 Empty Nest Stage: For Both DCSO Variants

Require: Output of getcuckoo stage: Population of routes V_P, Population of route reliability scores R_P, Best route V^*, Best route reliability score R^*

Ensure: New population of routes V_P, New population of route reliability scores R_P, Best route V^*, Best route reliability score R^*

1: Calculate the number of poorest solutions Z using (16).
2: **for** $k = 1$ to Z **do**
3: Obtain a first point of inversion $Q1$ using (14).
4: Apply the inversion mutation operation for the RW-DCSO algorithm to the best route V^* using (13) as $f_{INV}(V^*, Q1, h)$.
5: **end for**
6: Replace the Z poorest solutions with the newly mutated solutions.
7: **for** $k = 1$ to N_P **do**
8: Compute the new reliability score $R^{(k)}$ of this new route using (3)
9: $++I_C$ *Update fitness evaluation counter*
10: $R_P \leftarrow \{R_P, R^{(k)}\}$ *Save the new route reliability score*
11: $V_P \leftarrow \{V_P, V^{(k)}\}$ *Save the new route*
12: **if** $R^{(k)} > R^*$ **then**
13: $R^* \leftarrow R^{(k)}$ *Update best route reliability score*
14: $V^* \leftarrow V^{(k)}$ *Update best route*
15: **end if**
16: **if** $I_C > N_{FE}$ **then**
17: Terminate the algorithm
18: **return** R^*, V^*
19: **end if**
20: **end for**
21: **return** R^*, V^*

- **Inverse Mutation Operation for the RW-DCSO:** The sequence of vertices are reversed starting from $Q1$ to b, where $b = N_V$. A simple illustration of this process is shown in Fig. 5(b). The general function for the inverse mutation operator for the RW-DCSO algorithm is given in (13), as shown at the bottom of the previous page..

The getcuckoo stage for the RW-DCSO algorithm is summarized in Alg. 3. It is easily observed that Alg. 3 has fewer computational steps as against Alg. 2, thus providing a faster option for use under real-time scenario.

5) EMPTY NEST STAGE

The empty nest stage applies to both DCSO variants and it is the last operation performed by the CSO algorithm in order to obtain newer and improved solutions. The outputs of the getcuckoo stage are the inputs to the empty nest stage. An attempt is made in step 1 of Alg. 4 to discover the number of poorest solutions Z in the population based on the value of P_a as follows:

$$Z = \lceil P_a \times N_V \rceil \tag{16}$$

Algorithm 5 Overall Proposed DCSO Methods

Require: Population size N_P, Number of vehicles registered in the VANET N_V

Ensure: Best route V^*, Best route reliability score R^*

1: Run Alg.1
Main loop:
2: **while** Stopping criterion is not satisfied **do**
3: To use the LF-DCSO method, run Alg. 2,
4: Else, to use the RW-DCSO method, run Alg. 3
5: Run Alg. 4
6: **end while**
7: **return** R^*, V^*

Then, a number of new solutions corresponding to the number of the poorest solutions are obtained between steps 2 - 5 of Alg. 4. These new solutions are obtained by mutating the current global best route Z times with the aim to replace the poorest solutions by these new solutions. This approach allows us to preserve the quality of the current best route and improve the exploitation properties of our proposed algorithms.

The algorithm then replaces in step 6 the Z poorest routes in V_P by the new Z solutions obtained between steps 2 - 5. This can achieved by ranking the population of route reliability scores in R_P and then selecting the least Z scores as the poorest solutions. The positions of these routes with the poorest route reliability scores can be identified and then replaced with the new Z solutions.

The algorithm then evaluates this new population of solutions in steps 7 - 21 in order to obtain their new route reliability scores. It proceeds to update the best route reliability scores and the best route in steps 12 - 15. The stopping criterion is checked in steps 16 - 19 and the outputs are returned in step 21. This marks the end of the iterative process of our proposed DCSO variants. Consequently, an overall summary of the entire process is presented in Alg. 5.

D. COMPLEXITY ANALYSIS

The complexity analysis of the overall algorithm in Alg. 5 is evaluated in terms of the time complexity (TC) as well as the fitness function computational complexity (FC) of the algorithm. Essentially, the TC informs us about the time scalability of the algorithm as a function of an increasing input sample size, whereas the FC refers to the number of times the fitness function is computed in a defined number of iterations N_I.

The variables required to compute the FC include the population size N_P, the dimension of an individual solution N_V (i.e the number of vehicles), and the number of iterations N_I. It is noted that a population of solutions is typically generated once at the initialization stage in step 1 of Alg. 5 (i.e using Alg. 1) and mutated twice in steps 3 or 4 and in step 5 of Alg. 5, i.e. both at the getcuckoo stage (Alg. 2 or 3) and in the empty nest stage (Alg. 4). Consequently, the overall FC of Alg. 5 can generally be estimated as [32]

$$FC = N_P N_V + 2N_I N_V N_P \tag{17}$$

where the first term in (17) accounts for the number of fitness function computations in Alg. 1, whereas the second term accounts for the FC in the other two stages i.e (Alg. 2 or 3 and Alg. 4). It should be noted that only either of Alg. 2 or 3 can be used at the getcuckoo stage, which depends on the specific method (i.e the LF-DCSO or RW-DCSO method) deployed for use in the route discovery process. Hence, in the result section, we used the FC measure of (17) as the stopping criteria to compare the different MOAs evaluated in the present article.

On the other hand, the TC is analysed asymptotically by evaluating the number of *for* loops encountered in the entire process of Alg. 5. In this regard, we consider the input sample size as $N_P N_V$, which accounts for both the effect of the total population of solutions and the number of vehicles in the network. Thus, it can be observed that only a single *for* loop of N_P exists in Alg. 1 between steps 6 - 20, in Alg. 2 between step 5 - 23, in Alg. 3 between steps 1 - 15, and in Alg. 4 between steps 7 - 20. The only other *for* loop can be seen between steps 2 - 5 of Alg. 4, which is a function of Z. Thus, in the absence of any nested *for* loop and since all other steps in Alg. 1 - Alg. 4 are computed once in $O(1)$, hence, the overall TC of Alg. 5 can be approximated as

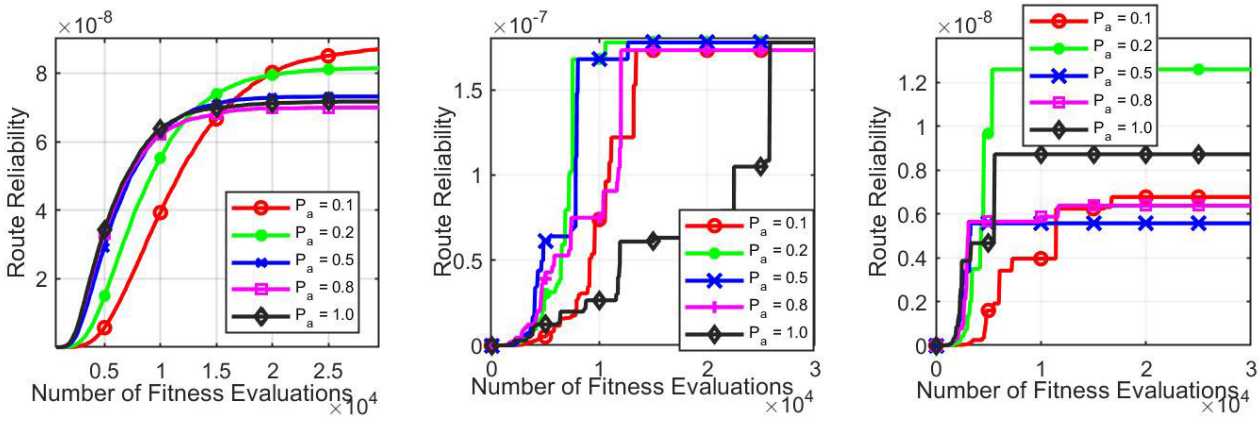
$$TC = 3 \times O(N_P N_V) + O(Z) \tag{18}$$

However, since $N_P N_V \gg Z$ and by neglecting the constant multiplicative term in (18), the overall asymptotic TC of our DCSO methods reduces to $O(N_P N_V)$. Then, the worst case TC of our methods would be $O(N_P^2)$ if $N_P = N_V$, which suggests that in the worst case, our methods are scalable in quadratic time. Similarly, our methods can be made to run faster (i.e. reduced to linear time) by ensuring that $N_P \ll N_V$.

VI. RESULTS AND DISCUSSION

We discuss our findings under two main subsections. In the first subsection, we present and discuss results with regards to the parameter configuration of our proposed DCSO variants. In the subsequent subsection, we compare these algorithms under different vehicle density scenarios, including in low (10 vehicles), medium (30 vehicles), and high (50 vehicles) density deployment scenarios. All graphical results were obtained following 1000 different Monte Carlo trials in order to improve the statistical significance of our findings. Thus, we report the mean, best, and worst results obtained per parameter. The mean results were averaged over the entire 1000 Monte Carlo trials, whereas the best and worst results are single realizations selected from the 1000 trials. In order to simulate the received SNR γ in dB at each receiving vehicle j, we considered for simplicity sake the use of the free space model stated as

$$\gamma_{i,j} = P_T^i - 20\log_{10}(d_{i,j}) + 20\log_{10}(f) - 147.5 - w_j, \tag{19}$$



(a) Mean Route Reliability (b) Best Route Reliability (c) Worst Route Reliability

FIGURE 6. Choice of probability of abandoning nests, P_a, for the variants of the DCSO. (a) The mean route reliability was obtained as an average over 1000 Monte Carlo trials, (b) The best route reliability represents the best route found out of 1000 Monte Carlo trials, (c) The worst route reliability corresponds to the worst route discovered out of 1000 Monte Carlo trials.

TABLE 1. Parameter settings for simulation.

Parameter	Values
P_T	0 dBW
f	5.925 GHz
A	1 km × 1 km

where P_T^i is the transmit power of each i^{th} transmitting vehicle, f is the operating frequency of the VANET, w_j is the noise power at each j^{th} receiving vehicle modelled as additive white Gaussian noise $w_j \sim \mathcal{N}(0, \sigma_w^2)$, $\sigma_w^2 = 1$ is the noise variance, and $d_{i,j}$ is the Euclidean distance between any i, j paired communicating vehicle computed as

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \tag{20}$$

where the coordinates $\{x_i, y_i\}$ and $\{x_j, y_j\}$ of any paired vehicles in the VANET were distributed over an area A via a uniform random distribution $\mathcal{U}(0, A)$, for $A \in \Re^2$. A summary of the parameter values used in our simulation are presented in Table 1. All codes were written and simulations were conducted in MATLAB R2019a

Dataset Generation: We considered a square spatial area of length $L = 1$ km within which vehicles were randomly deployed in the VANET. Consequently, it is noted that vehicles were uniformly distributed over an area of $A = 1$ km^2. Thus, the coordinates $\{x, y\}$ of each vehicle v was obtained as

$$x_v = L \times \varphi_v, \text{ for } v = 1, 2, 3, \ldots, N_V \tag{21}$$
$$y_v = L \times \phi_v, \text{ for } v = 1, 2, 3, \ldots, N_V \tag{22}$$

where φ and ϕ are random numbers generated between 0 and 1 via a uniform distribution function $\mathcal{U}(0, 1)$. Thus, any number of vehicle locations N_V can be simulated within a defined space A. We used this approach to generate the different vehicle density scenarios considered in the present article within a 1 km^2 area, where the set $N_V = \{10, 30, 50\}$ was used, which corresponds to the low, medium, and high vehicle density scenarios, respectively.

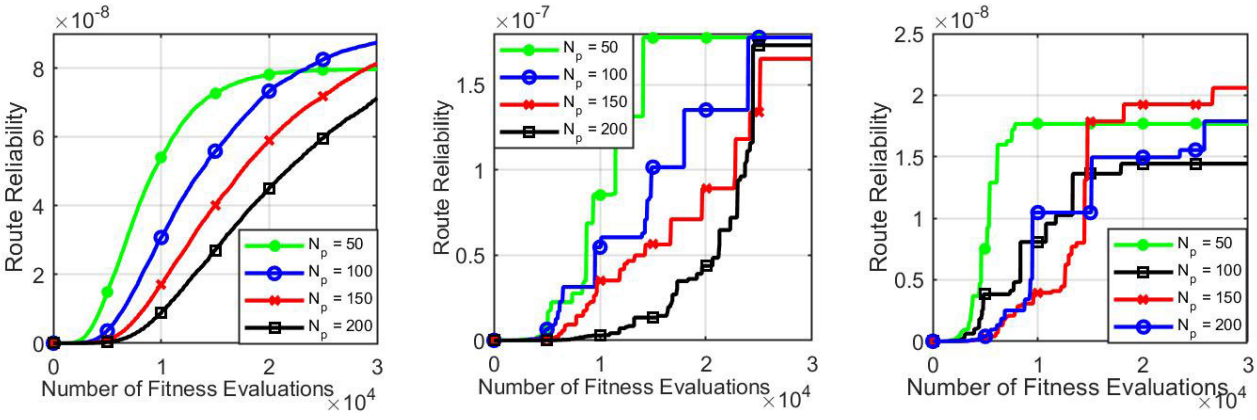
A. PARAMETER CONFIGURATION

The performance of our DCSO algorithms is governed by three main parameters, namely, the probability of discovering alien eggs (also called the probability of abandoning a nest), P_a, the population size N_p, and the number of times the fitness function is evaluated, which we term the number of fitness evaluations. Usually, the number of times the fitness function is computed typically controls the number of iterations as well as the running time of the algorithm, thus, all other parameters and performance metrics are compared as a function of the number of fitness evaluations. Other variables such as the average velocity of the vehicles involved in the network and the distance function deployed in the DCSO variants are also compared in this subsection. The results reported in this subsection were obtained for the case of the medium vehicle density scenario (i.e for 30 vehicles in the network), wherein vehicles were assigned an average velocity of 20 km/hr (see Section VI-A4).

1) EFFECT OF THE PROBABILITY OF ABANDONING NEST ON THE ROUTE DISCOVERY PERFORMANCE OF THE DCSO VARIANTS

Fig. 6 graphs the route reliability scores, which describes the quality of the discovered routes as a function of the number of times the fitness function was evaluated by our DCSO algorithms for different P_a values. Here, we present only the results obtained for the RW-DCSO algorithm, since the same performance results were obtained for the LF-DCSO variant.

Fig. 6(a) reports the mean values of the route reliability scores for different P_a values. We section the range of the fitness evaluations (i.e the x-axis) into three segments, namely, the fast-run range (i.e. between 1 - 10000 fitness computations), the mid-run range (i.e. between 10000 - 20000 fitness



(a) Mean Route Reliability (b) Best Route Reliability (c) Worst Route Reliability

FIGURE 7. **Choice of Population size, N_P, for the variants of the DCSO. (a) The mean route reliability was obtained as an average over 1000 Monte Carlo trials, (b) The best route reliability represents the best route found out of 1000 Monte Carlo trials, (c) The worst route reliability corresponds to the worst route discovered out of 1000 Monte Carlo trials.**

computations), and the long-run range (i.e. between 20000 - 30000 fitness computations). It can be observed in Fig. 6(a) that using larger P_a values (i.e. $P_a \geq 0.5$) accelerates the DCSO variants into discovering better routes as against the use of smaller P_a values. However, by the mid-run range, the use of smaller P_a values, particularly at $P_a = 0.2$, obviously outperforms the larger values in finding better routes. Interestingly, in the long-run, the use of smaller P_a values clearly outperforms the use of larger values. These findings clearly suggest that using smaller P_a values improves the exploration properties of the DCSO algorithm, which ensures that the algorithm avoids being stuck in local optimal values, as against the observation made for the use of larger P_a values. Thus, we concluded that the use of $P_a = 0.2$ in order to configure the DCSO algorithms yields the most satisfactory results.

Figs. 6(b) and 6(c) present the best and worst route reliability scores found under different P_a values. We observed that the use of $P_a = 0.2$ again yielded the best route (i.e the highest route reliability score) quicker than other values. In particular, the best route (see Fig. 6(a)) was discovered at about the end of the fast-run range, and which was well sustained into the long-run range. With regards to the worst routes discovered (see Fig. 6(c)), we observed that using $P_a = 0.2$ produced the highest reliability score from amongst the worst routes discovered, which implies that this value yet again yielded the best route from among the set of worst routes discovered under different parameter values.

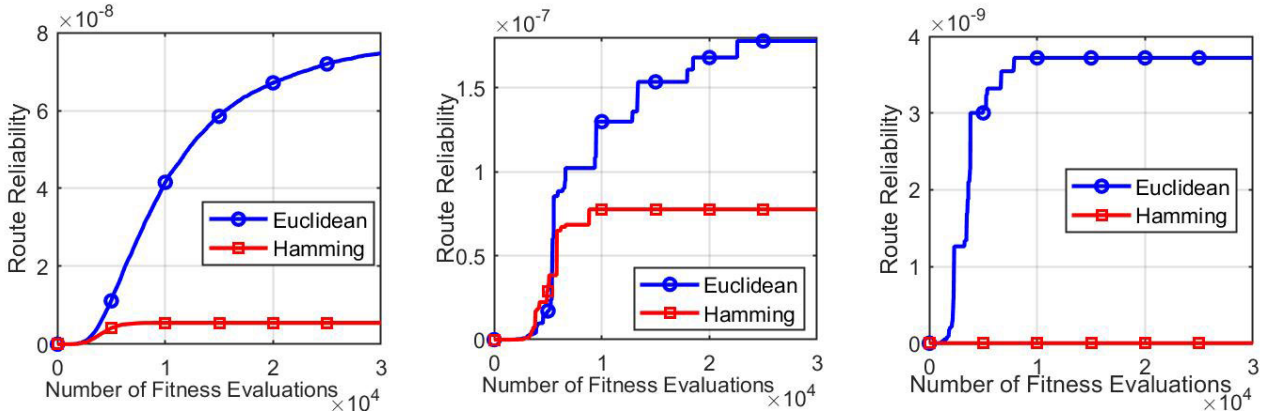
2) EFFECT OF POPULATION SIZE ON THE ROUTE DISCOVERY PERFORMANCE OF THE DCSO VARIANTS

We studied the effect of different population sizes N_P on the performance of our DCSO variants and our findings are presented in Fig. 7. We used $P_a = 0.2$ to configure our algorithms during the course of examining the different population sizes. The mean route reliability scores in Fig. 7(a) reveal that the use of $N_P = 50$ yielded the highest score starting in the fast-run range and sustained into the early stages of the long-run range, until it was surpassed only by $N_P = 100$. It should be noted that under real deployment scenarios, iterations may not necessarily persist into the long-run range since this might demand huge computational and memory resources, as well as prolonged delay. Thus, the algorithms are required to converge to the best routes either in the fast or mid-run range. On another note, the process of fairly evaluating different MOAs based on the number of fitness function consumed simply ensures that using larger population sizes does not yield any accuracy advantage for any MOA. This can be easily explained by noting that the number of batch computations θ over the computational process is obtained as $\theta = \frac{N_{FE}}{N_P}$, where N_{FE} is the total number of fitness function evaluations. Thus, it is easily observed that using larger N_P values in the denominator only reduces the number of batch computations, thus implying a faster computation time albeit at the expense of less accurate reliability scores. Thus, we concluded that the use of $N_P = 50$ yields a satisfactory result in order to configure our DCSO variants.

Further, Fig. 7(b) shows that the best route was discovered using $N_P = 50$ from early in the mid-run range, whereas all other population sizes could only discover the best route far in the long-run range. Again, we observe in Fig. 7(c) that using $N_P = 50$ produced the best route reliability score amongst the set of worst routes discovered. This route was discovered in the fast and into the mid-run range. Thus, these results suggest that $N_P = 50$ realizes the best performance for both DCSO variants.

3) EFFECT OF DIFFERENT DISTANCE FUNCTIONS ON THE ROUTE DISCOVERY PERFORMANCE OF THE LF-DCSO

We recall that different distance functions were used in the literature with contradictory conclusions. Authors in [25] adopted the Hamming distance function in their DCSO



(a) Mean Route Reliability (b) Best Route Reliability (c) Worst Route Reliability

FIGURE 8. Choice of the distance function used in the LF-DCSO. (a) The mean route reliability was obtained as an average over 1000 Monte Carlo trials, (b) The best route reliability represents the best route found out of 1000 Monte Carlo trials, (c) The worst route reliability corresponds to the worst route discovered out of 1000 Monte Carlo trials.

variant with poorer results to show, whereas authors in [9] used the Euclidean distance function and obtained better performances against other routing protocols. Thus, we investigated the use of both functions in our LF-DCSO variant and our findings are presented in Fig. 8.

The mean, best, and worst results shown in Figs. 8(a), 8(b), and 8(c) respectively, show that using the Euclidean distance function clearly outperforms the use of the Hamming distance function. An explanation for this result is that since both functions typically aim to compute the similarity (in terms of distance) between the best route and other routes within the population, however, the Euclidean distance function produces larger numerical values than the Hamming distance function, hence resulting in larger step sizes drawn from the Lévy flight walk and thus, better routes are found. Consequently, we concluded on the use of the Euclidean distance function in our LF-DCSO variant. Summarily, based on the mean results of Fig. 8(a), it is deduced that the Euclidean distance metric produces approximately about 1300% increase in its route reliability score over the use of the Hamming distance metric. This is a significantly large degree of improvement, thus, supporting the overwhelming decision to use the Euclidean distance function in our methods.

4) EFFECT OF CHANGING VELOCITY ON THE ROUTE DISCOVERY PERFORMANCE OF THE DCSO VARIANTS

We examined the required velocity of mobile vehicles needed by our DCSO variants in order to discover the best routes in VANETs. Here, we investigated different average velocities and our findings are reported in Fig. 9.

We observed that extremely low route reliability scores were obtained at high velocities, thus, we used the log-scale on the y-axis to ensure that results are properly visualized. Figs. 9(a), 9(b), and 9(c) reveal that the best routes are established when the average velocities of all vehicles in the VANET are below 20 km/hr. This observation agrees with intuition since the topology becomes steadier at lower mobility rates. Thus, the rest of our comparative analysis in the next subsection were conducted at an average velocity of 20 km/hr. It is seen that the reliability score at and above 20 km/hr approximates to zero, thus implying that it is practically difficult to form reliable routes above such a threshold value. Further, these results ensure that a threshold velocity of 20 km/hr can be used in VANETs in order to determine when vehicles are granted access to the network for onward routing of information.

B. COMPARATIVE PERFORMANCE ANALYSIS UNDER DIFFERENT VEHICLE DENSITY SCENARIOS

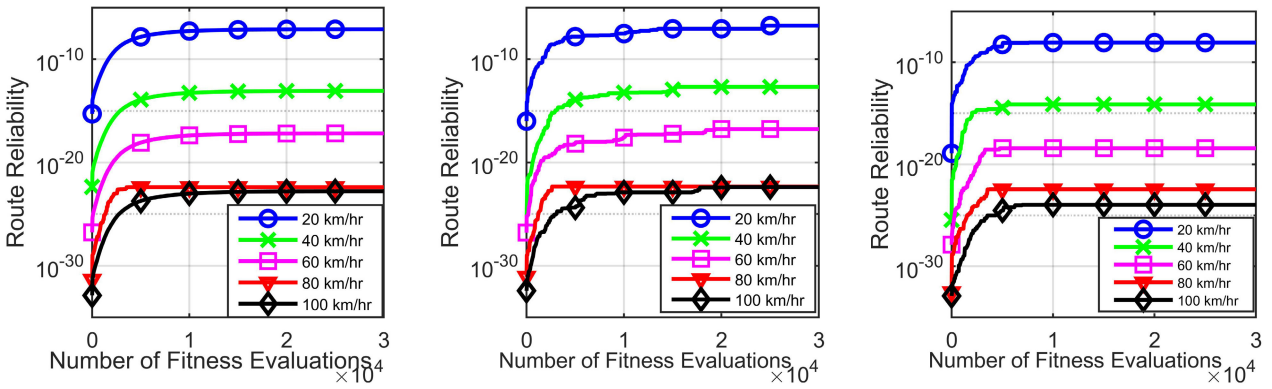
In this subsection, we compare the route discovery performance of our DCSO variants against the GA with roulette wheel (RW-GA) and the GA with k-means (IGAROT) [1]. The parameter configuration of these algorithms is documented in Table 2. We explore the case of low, medium, and high vehicle density scenarios. In each case, we present the route reliability graphs as well as the actual best routes found. Our findings are discussed as follows:

TABLE 2. Parameter settings of the methods compared in our experiments (population size = 50).

Algorithms	Parameter values
DCSO variants	$P_a = 0.2$
GA variants	Crossover rate = 0.8 Mutation rate = 0.2 Selection rate = 0.8 Selection operator = Roulette wheel for RW-GA Selection operator = k-means for IGAROT Elitism rate = 0.5

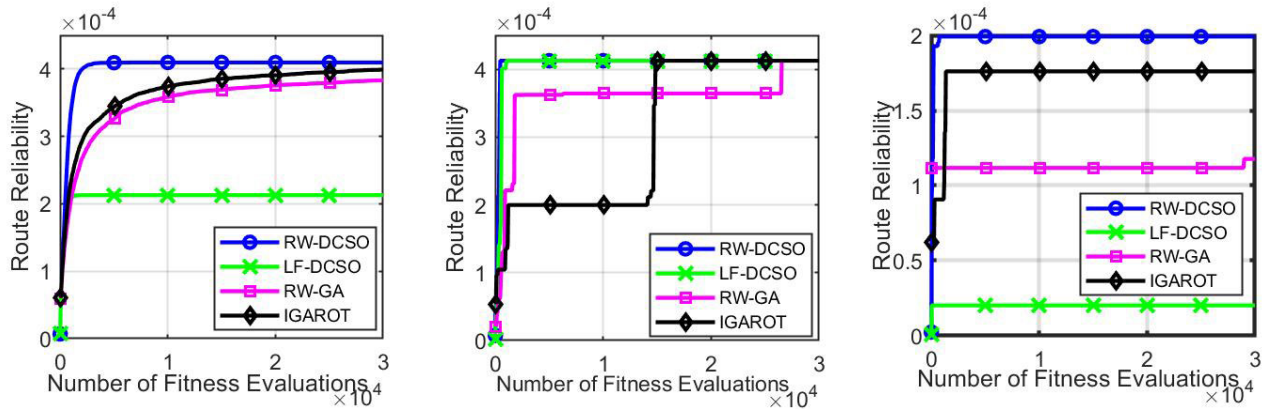
1) IN LOW VEHICLE DENSITY SCENARIOS

The low vehicular density scenario comprised 10 nodes assumed to be vehicles deployed randomly using the uniform distribution function within a 1 square km. The case of V2I



(a) Mean Route Reliability (b) Best Route Reliability (c) Worst Route Reliability

FIGURE 9. Choice of the average velocity at which vehicles may join the network to establish reliable routes (a) The mean route reliability was obtained as an average over 1000 Monte Carlo trials, (b) The best route reliability represents the best route found out of 1000 Monte Carlo trials, (c) The worst route reliability corresponds to the worst route discovered out of 1000 Monte Carlo trials.



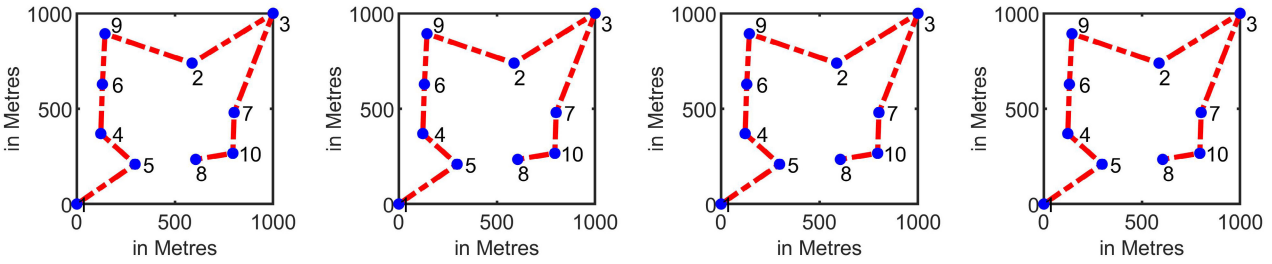
(a) Mean Route Reliability (b) Best Route Reliability (c) Worst Route Reliability

FIGURE 10. In low vehicular density (10 vehicles): Comparative route reliability performance of the different algorithms.

communication was considered, wherein the infrastructure, which could be a server, aims to send information to all vehicles in the VANET. This could be the dissemination of information about an accident scene or a dangerous road anomaly along a certain road. In our simulation, the infrastructure was located at the origin (0,0) and it aims to ensure that the best route is found in order to effectively disseminate information while avoiding the case of multiple routes via the same node. Fig. 10(a) presents the mean route performance obtained over 1000 Monte Carlo trials. It shows that the RW-DCSO algorithm typically discovered the better routes most of the time during the entire run ranges. This performance was followed by the IGAROT, RW-GA, and the LF-DCSO. The least performance of the LF-DCSO algorithm arises from larger step sizes computed during the Lévy walk, which led to solutions that fell outside the range of the discrete routes. Thus, in the event of violating the constraints, the LF-DCSO algorithm was forced into using smaller step sizes, which limited its average route discovery performance in most of the Monte Carlo trials.

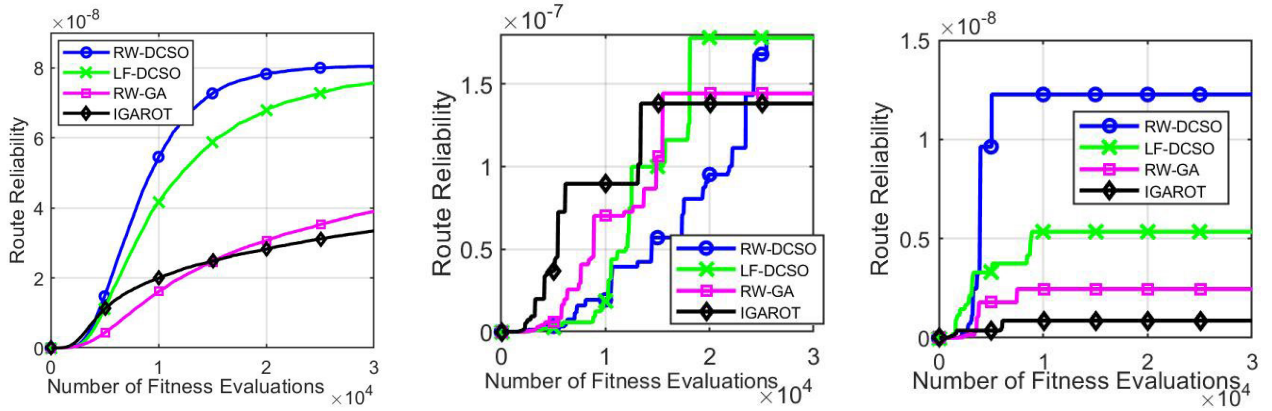
Nevertheless, Fig. 10(b) shows the best route reliability score discovered by each algorithm. Here, it is seen that the RW-DCSO and LF-DCSO algorithms were able to discover the best route starting from the fast-run range onwards. The IGAROT algorithm was able to find the best route only in the mid-run range, whereas the RW-GA discovered the best route only in the long run. The worst route scores in Fig. 10(c) show that the RW-DCSO algorithm produced the best scores in this category. Thus, the RW-DCSO algorithm is considered to be the best performer in the low density deployment case. In terms of percentage increment, it is deduced based on the mean results of Fig. 10(a) that in the long run our RW-DCSO algorithm achieved a percentage increment of 5.26%, 8.12%, and 90.48% in its route reliability score over the IGAROT, RW-GA, and LF-DCSO algorithms, respectively. Although not significantly better than the IGAROT and RW-GA in this case, nevertheless, it outperforms the LF-DCSO algorithm considerably as already noted above.

The validity of these route reliability scores is corroborated by the actual discovered best routes provided in Figs. 11.



(a) RW-CSO (b) LF-CSO (c) RW-GA (d) IGAROT

FIGURE 11. In low vehicular density (10 vehicles): Best discovered route.



(a) Mean Route Reliability (b) Best Route Reliability (c) Worst Route Reliability

FIGURE 12. In medium vehicular density (30 vehicles): Comparative route reliability performance of the different algorithms.

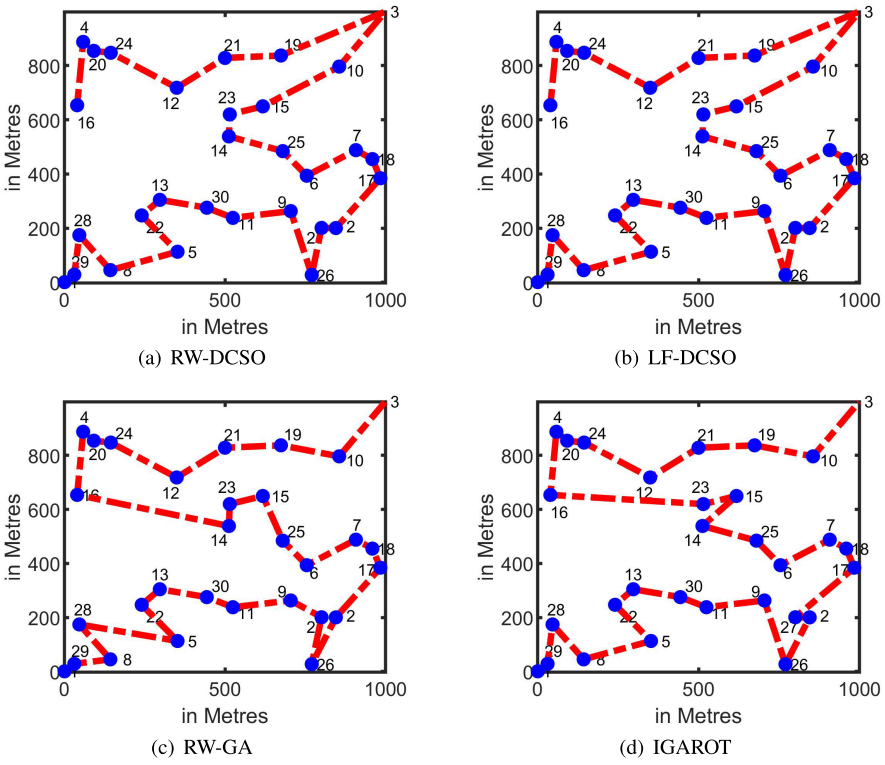
In this case, since all the algorithms were able to converge to the same best route reliability score (see Fig. 10(b)), thus it is observed that they all estimated the same best route as seen in Figs. 11(a) - 11(d), respectively.

2) IN MEDIUM VEHICLE DENSITY SCENARIOS

Thirty nodes were randomly distributed to simulate the case of the medium vehicle density scenario. The increased number of vehicles deployed in the VANET obviously increases the complexity of the route discovery process. The infrastructure (i.e. the server) is located at the origin and aims to disseminate information to all vehicles in the VANET.

The route reliability scores obtained are presented in Fig. 12. The mean scores in Fig. 12(a) indicate that the RW-DCSO algorithm demonstrated the greater ability to discover better routes most of the time. This performance was followed by the LF-DCSO algorithm, whereas poorer performances were recorded by the RW-GA and the IGAROT algorithms. It is evident that discovering the most effective route in dense vehicular deployment scenarios is a difficult task to realize, and thus both the GA variants found it quite challenging to discover good routes in this scenario. In terms of the best route score, although the GA variants achieved an early start in finding better routes, yet they never converged to the best score, whereas it is seen that in the mid and long-run range, the DCSO variants were able to discover the best routes. In the case of the worst routes, we observe in Fig. 12(c) that the RW-DCSO algorithm produced the best route scores followed by the LF-DCSO algorithm. Thus, we concluded that the RW-DCSO algorithm suffices as the best performer in this case. Based on Fig. 12(a), it can be deduced that in the long run the RW-DCSO algorithm achieved 2.56%, 100%, and 128.57% percentage increment in its route reliability score over the LF-DCSO, RW-GA, and IGAROT algorithms, respectively. Although the RW-DCSO algorithm does not significantly outperform the LF-DCSO algorithm, nevertheless, both algorithms are shown to significantly outperform the RW-GA and IGAROT algorithms.

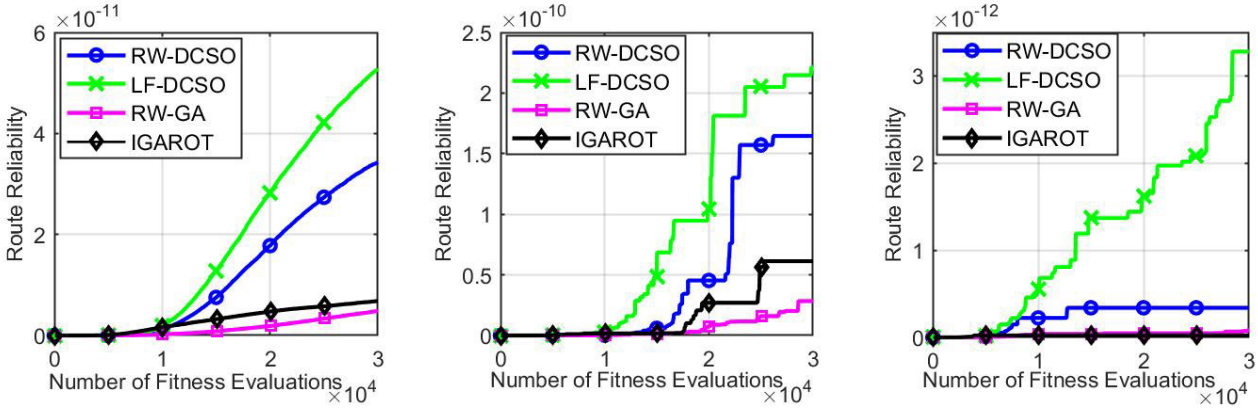
To further corroborate our findings, the best discovered routes are plotted per method in Figs. 13(a) - 13(d). It can be observed that the same routes are displayed for the RW-DCSO and the LF-DCSO algorithms. However, it is seen that the GA variants converged to less efficient routes. For example, we can observe in Fig. 13(c) that the RW-GA algorithm opted for the less efficient route from vehicles 29 - 8 - 28 - 5, instead of the shorter route from vehicles 29 - 28 - 8 - 5 as obtained by the RW-DCSO algorithm. In addition, the IGAROT algorithm selected a less efficient route in Fig. 13(d) from vehicles 23 - 16, instead of the more efficient option from vehicles 23 - 15 - 10, which



(a) RW-DCSO (b) LF-DCSO

(c) RW-GA (d) IGAROT

FIGURE 13. In medium vehicular density (30 vehicles): Best discovered route.



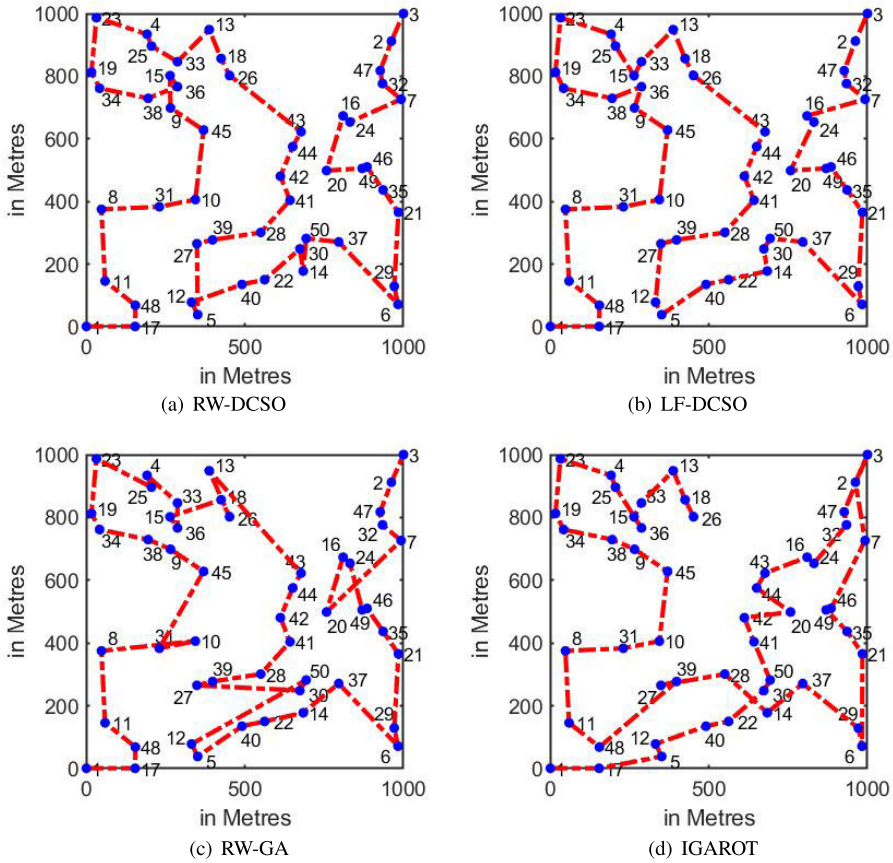
(a) Mean Route Reliability (b) Best Route Reliability (c) Worst Route Reliability

FIGURE 14. In high vehicular density (50 vehicles): Comparative route reliability performance of the different algorithms.

was selected by the DCSO variants. Thus, the results of Figs. 13(a) - 13(d) have substantiated the numerical scores earlier graphed in Fig. 12, which indicates that estimating higher reliability scores validly corresponds to discovering better routes in physical terms.

3) IN HIGH VEHICLE DENSITY SCENARIOS

The number of vehicles was increased to fifty within the 1 square km area to model a higher density deployment scenario. The mean scores of Fig. 14() show that it is obviously more difficult to find better routes in the fast-run range. However, starting from the mid-run range, it is observed that the LF-DCSO algorithm races to the better routes and onward into the long-run range. This is followed by the RW-DCSO algorithm, whereas the GA variants ranked lower in terms of finding good routes. Similarly the best route is found by the LF-DCSO algorithm in Figs. 14(b) and 14(c), respectively. We can deduce from Fig. 13(a) that in the long run the LF-DCSO algorithm achieved 42.85%, 525%, and 733.33% percentage increment in its route reliability score over the RW-DCSO, IGAROT, and RW-GA algorithms, respectively. Although marginally superior to the RW-DCSO algorithm,



(a) RW-DCSO (b) LF-DCSO

(c) RW-GA (d) IGAROT

FIGURE 15. In high vehicular density (50 vehicles): Best discovered route.

both the LF-DCSO and RW-DCSO algorithms are seen to have significantly outperformed the RW-GA and IGAROT algorithms.

The best discovered physical routes using each algorithm are plotted in Figs. 15. It can be observed that indeed, the LF-DCSO algorithm found the best route by closely following the traced routes. We note that the LF-DCSO algorithm opted for the route from vehicles 9 - 36 - 38, whereas the RW-DCSO converged to the route from vehicles 9 - 15 - 36 - 38. It is seen that the RW-DCSO algorithm converged to a longer and less efficient route, thus reducing its best route reliability score as shown in Fig. 14(b). Further, we observed the route disparity between both DCSO algorithms at vehicle 22 in Figs. 15(a) and 15(b), respectively. Here, the LF-DCSO algorithm converged to the more efficient route from vehicles 22 - 14 - 30 - 50, whereas the RW-DCSO algorithm opted for the less efficient route from vehicles 22 - 30 - 14 - 50. Thus, we concluded that the LF-DCSO algorithm suffices as the best performer, nevertheless, the RW-DCSO algorithm produced a relatively competitive performance in this regard.

C. SUMMARY OF FINDINGS

Our findings are summarized as follows:

1) In terms of the proper fine tuning of the parameters of our DCSO variants, we suggest the use of smaller values for the probability of discovering alien eggs, which leads to a better performance of the algorithm. Essentially, we have demonstrated that smaller P_a values improves the exploration property of our proposed DCSO algorithms, thus ensuring that they do not get stuck in local optimal.
2) Increasing the population size does not necessarily lead to better convergence properties. Instead, it simply increases the computational speed at the expense of reduced accuracy. This is explained in Section VI-A2, wherein larger population sizes only increased the step size and thus exhausted the total number of fitness evaluations within a shorter time frame, at the expense of reduced randomness. Further, reduced randomness leads to reduction in the diversity of the solutions in the population. Thus, smaller population sizes lead to better accuracy although at longer processing times.
3) Lower velocity of vehicles, typically at 20 km/hr and below is required to discover and establish reliable routes for information dissemination in VANETs.
4) The use of the Euclidean distance function in the LF-DCSO algorithm leads to improved performance since it produces larger step sizes in the Lévy flight walk.
5) The RW-DCSO algorithm generally yielded the best performance across all the density scenarios investigated in the present article. Essentially, it performed

best in the low and medium density scenarios, and competed well enough in the high density scenario.

6) The use of the Lévy flight probability distribution function to model step sizes in the LF-DCSO algorithm does not necessarily lead to better performance as against the random walk version, which is based on the uniform random probability distribution function. This is an interesting finding that may be peculiar to the case of the discrete combinatorial) optimization problem as obtained in VANETs. We note as a caveat, that this improved performance of the random walk may not necessarily apply in the case of continuous optimization problems.

VII. CONCLUSION

Our proposed discrete variants of the cuckoo search optimization (DCSO) algorithm, namely, the Lévy and the random walk approach have been shown to improve route discovery performance in VANETs. The improved performance of our methods was demonstrated against the genetic algorithm (GA) and its variant termed IGAROT. Our methods were developed following some freshly investigated ideas, including the application of the inverse mutation operator in our DCSO algorithms and a proposed objective function for modelling the reliability of the communication link between pairs of communicating vehicles. The proposed methods are shown to provide improved performance under different vehicle density scenario. The routing protocol that demonstrates the viability of our new methods for route discovery in VANETs was discussed. Although shown to provide improved performance under different operating conditions, our proposed methods can be further enhanced in the following areas: the exploitation property of our methods can be improved to obtain better solutions in a shorter time range. Further, they can be extended to the case of actual data transmission in more realistic VANET scenarios to evaluate their performances under realtime use. We shall consider in future works the more complex case of vehicles and road network modelling to cover vehicles moving at different velocities across the network in the presence of obstacles. The use of IoT-based long range communication technologies such as Sigfox, LoRa, NB-IoT can be explored in further works to increase single hop communication range in VANETs. Nevertheless, the discovery of candidate Hamiltonian routes in VANETs for effective communication via single, loop-free routes is demonstrated to be realizable and should be pursued and further developed in future IoT-based VANETs.

REFERENCES

[1] H. Bello-Salau, A. M. Aibinu, Z. Wang, A. J. Onumanyi, E. N. Onwuka, and J. J. Dukiya, "An optimized routing algorithm for vehicle ad-hoc networks," *Eng. Sci. Technol., Int. J.*, vol. 22, no. 3, pp. 754–766, 2019.

[2] H. Bello-Salau, A. M. Aibinu, A. J. Onumanyi, E. N. Onwuka, J. J. Dukiya, and H. Ohize, "New road anomaly detection and characterization algorithm for autonomous vehicles," *Appl. Comput. Informat.*, pp. 1–10, Jul. 2020.

[3] A. Dua, N. Kumar, and S. Bawa, "A systematic review on routing protocols for vehicular ad hoc networks," *Veh. Commun.*, vol. 1, no. 1, pp. 33–52, Jan. 2014.

[4] K. Mehta, P. R. Bajaj, and L. G. Malik, "Fuzzy bacterial foraging optimization zone based routing (FBFOZBR) protocol for VANET," in *Proc. Int. Conf. ICT Bus. Ind. Government (ICTBIG)*, 2016, pp. 1–10.

[5] M. A. Gawas and S. S. Govekar, "A novel selective cross layer based routing scheme using ACO method for vehicular networks," *J. Netw. Comput. Appl.*, vol. 143, pp. 34–46, Oct. 2019.

[6] H. Bello-Salau, A. J. Onumanyi, A. M. Aibinu, E. N. Onwuka, J. J. Dukiya, and H. Ohize, "A survey of accelerometer-based techniques for road anomalies detection and characterization," *Int. J. Eng. Sci. Appl.*, vol. 3, no. 1, pp. 8–20, 2019.

[7] F. Abbas and P. Fan, "Clustering-based reliable low-latency routing scheme using ACO method for vehicular networks," *Veh. Commun.*, vol. 12, pp. 66–74, Apr. 2018.

[8] A. Ouaarab, B. Ahiod, and X.-S. Yang, "Discrete cuckoo search algorithm for the travelling salesman problem," *Neural Comput. Appl.*, vol. 24, nos. 7–8, pp. 1659–1669, Jun. 2014.

[9] B. Ramakrishnan, S. Sreedivya, and M. Selvi, "Adaptive routing protocol based on cuckoo search algorithm (ARP-CS) for secured vehicular ad hoc network (VANET)," *Int. J. Comput. Netw. Appl.*, vol. 2, no. 4, pp. 173–178, 2015.

[10] K. Bibiks, Y.-F. Hu, J.-P. Li, P. Pillai, and A. Smith, "Improved discrete cuckoo search for the resource-constrained project scheduling problem," *Appl. Soft Comput.*, vol. 69, pp. 493–503, Aug. 2018.

[11] T. E. Ouahmani, A. Chehri, and N. Hakem, "Bio-inspired routing protocol in VANET networks—A case study," *Procedia Comput. Sci.*, vol. 159, pp. 2384–2393, 2019.

[12] S. Bitam, A. Mellouk, and S. Zeadally, "Bio-inspired routing algorithms survey for vehicular ad hoc networks," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 843–867, 2nd Quart., 2015.

[13] A. M. Oranj, R. M. Alguliev, F. Yusifov, and S. Jamali, "Routing algorithm for vehicular ad hoc network based on dynamic ant colony optimization," *Int. J. Electron. Electr. Eng.*, vol. 4, no. 1, pp. 79–83, 2016.

[14] H. Dong, X. Zhao, L. Qu, X. Chi, and X. Cui, "Multi-hop routing optimization method based on improved ant algorithm for vehicle to roadside network," *J. Bionic Eng.*, vol. 11, no. 3, pp. 490–496, Sep. 2014.

[15] M. Chahal and S. Harit, "A stable and reliable data dissemination scheme based on intelligent forwarding in VANETs," *Int. J. Commun. Syst.*, vol. 32, no. 3, p. e3869, Feb. 2019.

[16] M. Chahal and S. Harit, "Network selection and data dissemination in heterogeneous software-defined vehicular network," *Comput. Netw.*, vol. 161, pp. 32–44, Oct. 2019.

[17] V. Jindal and P. Bedi, "An improved hybrid ant particle optimization (IHAPO) algorithm for reducing travel time in VANETs," *Appl. Soft Comput.*, vol. 64, pp. 526–535, Mar. 2018.

[18] H. Rana, P. Thulasiraman, and R. K. Thulasiram, "MAZACORNET: Mobility aware zone based ant colony optimization routing for VANET," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 2948–2955.

[19] M. B. Wagh and N. Gomathi, "Route discovery for vehicular ad hoc networks using modified lion algorithm," *Alexandria Eng. J.*, vol. 57, no. 4, pp. 3075–3087, Dec. 2018.

[20] M. Fahad, F. Aadil, S. Khan, P. A. Shah, K. Muhammad, J. Lloret, H. Wang, J. W. Lee, and I. Mehmood, "Grey wolf optimization based clustering algorithm for vehicular ad-hoc networks," *Comput. Electr. Eng.*, vol. 70, pp. 853–870, Aug. 2018.

[21] S. Bitam and A. Mellouk, "Bee life-based multi constraints multicast routing optimization for vehicular ad hoc networks," *J. Netw. Comput. Appl.*, vol. 36, no. 3, pp. 981–991, May 2013.

[22] M. Elhoseny, "Intelligent firefly-based algorithm with Levy distribution (FF-L) for multicast routing in vehicular communications," *Expert Syst. Appl.*, vol. 140, Feb. 2020, Art. no. 112889.

[23] R. Hajlaoui, E. Alsolami, T. Moulahi, and H. Guyennet, "Construction of a stable vehicular ad hoc network based on hybrid genetic algorithm," *Telecommun. Syst.*, vol. 71, no. 3, pp. 433–445, Jul. 2019.

[24] X. Bao, H. Li, G. Zhao, L. Chang, J. Zhou, and Y. Li, "Efficient clustering V2V routing based on PSO in VANETs," *Measurement*, vol. 152, Feb. 2020, Art. no. 107306.

[25] A. D. Masegosa, E. Osaba, J. S. Angarita-Zapata, I. Laña, and J. D. Ser, "Nature-inspired metaheuristics for optimizing information dissemination in vehicular networks," in *Proc. Genetic Evol. Comput. Conf. Companion*, Jul. 2019, pp. 1312–1320.

[26] M. Chahal and S. Harit, "Optimal path for data dissemination in vehicular ad hoc networks using meta-heuristic," *Comput. Electr. Eng.*, vol. 76, pp. 40–55, Jun. 2019.
[27] B. Du, Q. Wei, and R. Liu, "An improved quantum-behaved particle swarm optimization for endmember extraction," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 8, pp. 6003–6017, Aug. 2019.
[28] L. Tong, B. Du, R. Liu, and L. Zhang, "An improved multiobjective discrete particle swarm optimization for hyperspectral endmember extraction," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 10, pp. 7872–7882, Oct. 2019.
[29] M. Xu, B. Du, and Y. Fan, "Endmember extraction from highly mixed data using linear mixture model constrained particle swarm optimization," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 8, pp. 5502–5511, Aug. 2019.
[30] C. H. Papadimitriou, "The Euclidean travelling salesman problem is NP-complete," *Theor. Comput. Sci.*, vol. 4, no. 3, pp. 237–244, Jun. 1977.
[31] X. S. Yang and S. Deb, "Cuckoo search via *Lévy* flights," in *Proc. World Congr. Nature Biol. Inspired Comput. (NABIC)*, 2009, pp. 210–214.
[32] M. Črepinšek, S.-H. Liu, L. Mernik, and M. Mernik, "Is a comparison of results meaningful from the inexact replications of computational experiments?" *Soft Comput.*, vol. 20, no. 1, pp. 223–235, Jan. 2016.
[33] N. Veček, M. Mernik, B. Filipič, and M. Črepinšek, "Parameter tuning with chess rating system (CRS-Tuning) for meta-heuristic algorithms," *Inf. Sci.*, vol. 372, pp. 446–469, Dec. 2016.

HABEEB BELLO-SALAU (Member, IEEE) received the B.Tech. degree in electronic/electrical engineering from the Ladoke Akintola University of Technology, Ogbomoso, Nigeria, in 2009, the M.Sc. degree in communication engineering from International Islamic University Malaysia, Kuala-Lumpur, in 2012, and the Ph.D. degree in communication engineering from the Federal University of Technology Minna, Minna, Nigeria, in 2017. He is currently with the Department of Computer Engineering, Ahmadu Bello University, Zaria, Nigeria. He has authored and coauthored more than 20 different research articles in peer reviewed journals and over 30 conference papers. His research interests include digital signal and image processing, vehicle ad-hoc networks, artificial intelligence, cognitive radio, and wireless sensor networks.

ADEIZA JAMES ONUMANYI (Member, IEEE) received the B.Eng. degree in electrical and electronics engineering from Abubakar Tafawa Balewa University, Bauchi, Nigeria, in 2005, and the M.Eng. and Ph.D. degrees in communication engineering from the Federal University of Technology (FUT) Minna, Minna, Nigeria, in 2010 and 2014, respectively. He was a Postdoctoral Research Fellow with the University of Pretoria, South Africa, from 2018 to 2019. He is currently a Lecturer with the Department of Telecommunication Engineering, FUT Minna. He has published several research articles in different peer reviewed journals, and in different IEEE flagship conferences. He has won several grants at FUT Minna, served on several organizing committees for different conferences, including the IEEE conferences, reviewed several articles for high impact journals. He has participated in different technical workshops. His research interests include spectrum sensing in cognitive radio, wireless sensor networks, radar systems, image processing, cyber physical systems, and low powered wireless area networks.

ADNAN M. ABU-MAHFOUZ (Senior Member, IEEE) received the M.Eng. and Ph.D. degrees in computer engineering from the University of Pretoria. He is currently a Principal Researcher with the Council for Scientific and Industrial Research (CSIR), a Professor Extraordinaire with the Tshwane University of Technology, a Visiting Professor with the University of Johannesburg, and an Extraordinary Faculty member with the University of Pretoria. He has participated in the formulation of many large and multidisciplinary Research and Development successful proposals (as a Principal Investigator or a main Author/Contributor). He is the founder of the Smart Networks collaboration initiative that aims to develop efficient and secure networks for the future smart systems, such as smart cities, smart grid, and smart water grid. His research interests include wireless sensor and actuator networks, low power wide area networks, software defined wireless sensor networks, cognitive radio, network security, network management, and sensor/actuator node development. He is a member of many IEEE Technical Communities. He is an Associate Editor of IEEE Access, the IEEE Internet of Things, and the IEEE Transaction on Industrial Informatics.

ACHONU O. ADEJO (Member, IEEE) received the first degree in electrical/computer engineering from the Federal University of Technology Minna, Minna, Nigeria, in 2006, the M.Sc. degree from the University of Nottingham Malaysia, in 2010, and the Ph.D. degree in electrical engineering from Newcastle University, U.K., in 2018. His doctorate program was carried out in the Communications, Sensors and Signal processing group at the School of Engineering. Since 2010, he has been with the Federal University of Technology Minna. His current research interests include resource management and modeling of cellular communications with focus on 5G communications and D2D networks.

MUHAMMED BASHIR MU'AZU (Member, IEEE) received the M.Sc. and Ph.D. degrees from Ahmadu Bello University, Zaria, Nigeria. He is currently the Research Team Lead of the Concept to Product Laboratory and the Head of the Department of computer Engineering, Ahmadu Bello University. He was the Coordinator of the Cisco Network Academy and the Deputy Director of the ICT Directorate and the immediate past Director of the Institute (ICICT), ABU Zaria. He led the University Fibre-Optic Backbone project toward networking the whole University. A project to which he won the Vice Chancellor's distinguishing Award during the 2014 convocation. He is the Chairman of the Nigerian Communications Commission (NCC), and the Inter-Agency Committee on Telecommunications Research Proposals from the Academia. He has authored and coauthored more than 60 different research articles in peer reviewed journals and over 40 conference papers. His research interests include control systems, computational intelligence and bio-inspired systems, computing and networking, microcontroller applications, and energy systems.