Mathematical Biosciences and Engineering

MBE, 19(7): 7284–7313.
DOI: 10.3934/mbe.2022344
Received: 31 March 2022
Revised: 30 April 2022
Accepted: 09 May 2022
Published: 18 May 2022

http://www.aimspress.com/journal/MBE

Research article

Parameter optimization of shared electric vehicle dispatching model using discrete Harris hawks optimization

Yuheng Wang[1], Yongquan Zhou[1,2]* and Qifang Luo[1,2]

[1] College of Artificial Intelligence, Guangxi University for Nationalities, Nanning 530006, China
[2] Guangxi Key Laboratories of Hybrid Computation and IC Design Analysis, Nanning 530006, China

* **Correspondence:** Email: yongquanzhou@126.com; Tel: +8613607882594; Fax: +867713265523.

Abstract: The vehicle routing problem (VRP) problem is a classic NP-hard problem. Usually, the traditional optimization method cannot effectively solve the VRP problem. Metaheuristic optimization algorithms have been successfully applied to solve many complex engineering optimization problems. This paper proposes a discrete Harris Hawks optimization (DHHO) algorithm to solve the shared electric vehicle scheduling (SEVS) problem considering the charging schedule. The SEVS model is a variant of the VPR problem, and the influence of the transfer function on the model is analyzed. The experimental test data are based on three randomly generated examples of different scales. The experimental results verify the effectiveness of the proposed DHHO algorithm. Furthermore, the statistical analysis results show that other transfer functions have apparent differences in the robustness and solution accuracy of the algorithm.

Keywords: discrete Harris hawks optimization; shared electric vehicle dispatching scheduling; transfer function; metaheuristic optimization

1. Introduction

With the advocacy of green travel, the development concept of low-carbon life, shared electric vehicles have recently become our answer to increasing environmental pollution and the energy crisis. The shared electric vehicle dispatching scheduling (SEVS) problem is a car rental model where customers can rent a car for a relatively short time, usually an operator whose owner is responsible for maintaining it [1]. Compared with traditional travel methods such as private cars, shared electric

vehicles can effectively improve vehicle utilization, reduce urban traffic congestion, reduce user travel costs, and effectively reduce urban carbon emissions. According to the different ways of providing car-sharing services, it can be divided into three models based on two-way stations, one-way stations, and free-floating [2]. Specifically, in the two-way mode, the user needs to go to a site designated by the service provider to find an available car, the parking site is a parking lot defined by the service provider or the local administration, and the user's journey must start and end at the same site [3]. Therefore, this operating model does not consider intermediate parking, which is a parking spot that customers may plan based on their individual needs. The collection of parking lots is predefined. The one-way mode is like the two-way mode, but in the one-way case, the stop at the end of the journey may be different from the stop at the start of the journey [3]. The collection of parking sites is predefined. The free-floating model is the last model to enter the market, where vehicles are free to park in public spaces within the operating area (i.e., the area served by the car-sharing company), and users can start and end services anywhere in the area [4]. This article mainly focuses on the one-way site pattern. While shared EVs bring benefits to users, they also present many challenges for vehicle managers, two of the most important being the need for charging infrastructure and the need to redistribute vehicles.

The metaheuristic optimization algorithm is a technology combining random and local search method, according to the principle of the internal mechanism of the algorithm, the population-based metaheuristic algorithm can be divided into three categories. The first type is the evolutionary algorithm, which is subject to the natural selection of the biological world and the law of survival of the fittest, using crossover, mutation, and selection operators to operate. The more famous ones are Genetic Algorithm (GA), Differential Evolution Algorithm (DE), Evolutionary Strategy (ES), Evolutionary Programming (EP), Cultural Algorithm (CA) [5–9], etc. The second category is algorithms inspired by physicochemical phenomena, including simulating physics, chemical laws, mathematical formulas, etc. The well-known algorithms are the Big Bang-Big Shrinking Algorithm (BBBC), Gravity Search Algorithm (GSA), Command System Search (CSS), Magnetic Field Optimization Algorithm (MOA), Center Force Optimization (CFO), Artificial Chemical Reaction Optimization Algorithm (ACROA) [10–15], Black Hole Algorithm (BH), Small World Optimization Algorithm (SWOA), Galaxy Search Algorithm (GBSA), Space Gravity Optimization (SGO), Henry Gas Solubility Optimization, Arithmetic Optimization Algorithm (AOA), Runge Kutta method (RUN) [16–22] etc. The third category is optimization algorithms inspired by the behavior of animals in nature. Such as Particle Swarm Optimization Algorithm (PSO), Artificial Bee Colony Algorithm (ABC), Ant Colony Optimization Algorithm (ACO), Firefly Optimization Algorithm (FA), Bat Algorithm (BA), Cuckoo Search Algorithm (CS) [23–28], Artificial Algae Algorithm (AAA), Tree Species Algorithm (TSA), Grey Wolf Optimization Algorithm (GWO), Social Spider Algorithm (SSA), Moth Flame Algorithm (MFA), Whale Optimization Algorithm (WOA), Dolphin Echo Localization Algorithm (DEA) [29–35], Cat Swarm Optimizer (CSO), Lion Swarm Optimization Algorithm (LOA), Fruit Fly Optimization Algorithm (FOA), Chimpanzee Optimization Algorithm (CHOA), Chameleon Optimization Algorithm (CSA), Slime Mould Algorithm (SMA), Hunger Games Search (HGS), Colony Predation Algorithm (CPA) [36–43], and so on. The metaheuristic algorithm can better solve the complex optimization problems that cannot be solved by traditional optimization algorithms and cannot be effectively solved and have better performance than traditional methods in many practical application problems.

The Vehicle Routing Problem (VRP) was proposed by (Dantzig & Ramser, 1959) and is one of the most attractive topics in operations research, communications, manufacturing, transportation, distribution, and logistics [44]. VRP is an np-hard problem, and its real-life applications are on a much

larger scale, so metaheuristics are often better suited for real-world applications. Many scholars have applied meta-heuristic algorithms to the VRP problem. Du et al. proposed to use SA to solve the vehicle routing problem in multiple depots for dangerous goods transportation [45]; García-Nájera et al. used GA to study the multi-objective vehicle routing problem with backhaul [46]; Cao et al. open vehicle routing problem based on DE research demand uncertainty [47]; Jabir et al. proposed the design and development of a multi-vehicle green vehicle hybrid ant colony variable neighborhood search algorithm [48]; Okulewicz et al. based on particle swarm optimization to solve the dynamic vehicle routing problem of specific components [49]; Iqbal et al. employed an artificial bee colony algorithm with a soft time window to solve the routing problem of multi-objective aircraft [50]; Teymourian presented improved Intelligent water droplets and cuckoo search algorithms solve the capable vehicle routing problem [51] et al.

The Harris Hawks optimization algorithm is a new metaheuristic algorithm [52], which has been applied to some practical problems by many scholars. Fan et al. proposed an improved Harris Hawks optimization training the neural network based on the learning of neighborhood centroid duality [53]; Zhang et al. employed a deep neural network and Harris hawks optimization algorithm to estimate the friction angle of clays in evaluating slope stability of a generalized artificial intelligence model [54]; Mouassa et al. based on Harris Hawks optimization algorithm to solve scheduling of smart home appliances for optimal energy management in smart grid [55]; Kumar et al. used extra tree classifier and enhanced Harris Hawks optimization algorithm for software component reusability prediction [56]; Naik et al. confirmed that a leader Harris hawks optimization is able to solve 2-D Masi entropy-based multilevel image thresholding [57]; Mossa et al. used Harris Hawks optimization algorithm and atom search optimization algorithm to address parameter estimation of PEMFC model [58]; Houssein et al. hybridized Harris Hawk optimization and support vector machines for drug design and discovery [59]; Setiawan et al. presented the use of Harris Hawks optimization for parameter optimization of support vector regression [60]; Dehkordi et al. proposes a non-linear chaotic Harris Hawk optimizer to solve the path planning problem of vehicle networks [61]. However, by reading the review [62], it is found that HHO has little research on VRP. This paper studies the performance of HHO in VRP variant shared electric vehicle scheduling problem considering the charging schedule (SEVS) [63]. As the SEVS model is discrete, discrete methods can impact the performance of the algorithm on the model, so this paper also analyses the impact of eight transfer functions on the performance of the model.

The rest of this paper is organized as follows: Section 2 introduces the SEVS mathematical model. Section 3 presented the DHHO algorithm and applied it to solving the SEVS problem. Section 4 is the analysis of the experimental results. Finally, Section 5 concludes future work.

2. Preliminaries

2.1. Mathematical model of SEVS

2.1.1. Problem description

Suppose a company provides shared car service in a certain area and has a dispatching center. The vehicles providing services are homogeneous pure electric vehicles. During the period of low demand every day (such as night), the platform sends employees to dispatch vehicles for two purposes. The first is to rebalance the distribution of vehicles between stations. The second is to transport the vehicles

with insufficient power to the station with a charging pile to charge the vehicles for continued service the next day. The way of transportation is that employees start from the dispatching center, ride folding bicycles to the station where vehicles need to be transferred out, drive a vehicle to a station where vehicles need to be transferred out (folding bicycles are carried with the vehicle), and then ride bicycles to the next station where vehicles need to be transferred out. In this way, we can complete the transportation task assigned to the employee. The parameters and symbols used in the problem are explained in Table 1.

Table 1. Parameter name and meaning explanation.

Symbol	Interpretation	Symbol	Interpretation
K	number of employees	d_{ij}	The distance from the i station to the j station
C	Collection of vehicles	n_j	The difference between the actual number of Parked vehicles at station j and the number of vehicles in the ideal state
C_1	Collection of fully charged vehicles	m_i	Remaining power of vehicle i
C_2	Collection of vehicles with insufficient battery	α	The lowest value for a fully charged vehicle
S	Collection of sites	β	The vehicle is transferred to a site without a charging station, the maximum power remaining
S_1	Saturated sites with charging piles	s_{ij}	Vehicle i parked at station j is 1, otherwise it is 0
S_2	Unsaturated sites with charging piles	v_B	The speed at which employees are riding bicycles
S_3	Saturated sites without charging piles	δ	Ride cost per unit distance
S_4	Unsaturated sites without charging piles	v_C	The speed at which the employee drives the vehicle
$\{0\}$	Dispatch center	γ	Driving cost per unit distance
L	The cruising range when the vehicle is fully charged	T	Maximum total working hours per employee

2.1.2. Feasibility analysis of vehicle dispatching among various types of stations

Considering the different types of stations and vehicles, some stations can only transfer specific vehicles. For example, vehicles transferred to a saturated site $j \in S_1$ with charging piles can only be vehicles that are parked at a certain station without charging piles and have insufficient power.

Correspondingly, vehicles with sufficient power can be transferred from this type of site to unsaturated sites and saturated sites without charging piles. Vehicles transferred to an unsaturated site $j \in S_2$ with charging piles can only be vehicles parked at a saturated site with sufficient power and vehicles parked at a site without charging piles with insufficient power, and vehicles should not be transferred from this type of site. Vehicles transferred to a saturated site $j \in S_3$ without charging piles can only be vehicles parked at a saturated site with charging piles and have sufficient power. Correspondingly, vehicles with sufficient power can be transferred from this type of site to an unsaturated site, and vehicles with insufficient power can be transferred from this type of site to a site with charging piles. Vehicles transferred to an unsaturated site $j \in S_4$ without charging piles can only be vehicles parked at a saturated site with sufficient power. Correspondingly, vehicles with insufficient power can be transferred from this type of site to a site with charging piles. The types of vehicles that can be transferred in and the types of vehicles that can be transferred and the source and destination of each type of station are shown in Table 2.

Table 2. Transfer in and out vehicle information of various types of sites.

Site type	Feasible transfer into the vehicle		Feasible transfer of the vehicle	
	Type	Source site	Type	Destination site
S1	low battery	S3, S4	Fully charged	S2, S3, S4
S2	Fully charged	S1, S3	-	-
	low battery	S3, S4		
S3	Fully charged	S1	Fully charged	S2, S4
			low battery	S1, S2
S4	Fully charged	S1, S3	low battery	S1, S2

Note:'-' means no vehicle can be transfer out.

2.1.3. Model and interpretation

The 0-1 nonlinear programming model is a classic mathematical problem as follow:

$$x_{ijk} = \begin{cases} 1, & if \rightleftarrows \ employee\ k \in K\ transitions\ from\ node\ i\ to\ node\ j \\ 0, & otherwise \end{cases} \tag{1}$$

The nodes include the dispatch center, the collection of vehicles, and the collection of stations. The shared electric vehicle dispatching problem considering charging schedule can be described as the following 0-1 nonlinear programming model,

$$\min \gamma \sum_{k\in K}\sum_{i\in C}\sum_{j\in S} d_{ij}x_{ijk} + \delta\left(\sum_{k\in K}\sum_{i\in S}\sum_{j\in C} d_{ij}x_{ijk} + \sum_{k\in K}\sum_{i\in C} d_{0i}x_{0ik} + \sum_{k\in K}\sum_{j\in S} d_{j0}x_{j0k}\right) \tag{2}$$

Subject to:

$$\sum_{k\in K}\sum_{i\in C}\sum_{j\in C} x_{ijk} + \sum_{k\in K}\sum_{i\in S}\sum_{j\in S} x_{ijk} + \sum_{k\in K}\sum_{i\in C} x_{i0k} + \sum_{k\in K}\sum_{j\in S} x_{0jk} + \sum_{k\in K}\sum_{i\in C}\sum_{j\in S} s_{ij}x_{ijk} + \sum_{k\in K}\sum_{i\in C_1}\sum_{j\in S_1} x_{ijk} + \sum_{k\in K}\sum_{i\in C_2}\sum_{j\in S_3\cup S_4} x_{ijk} = 0 \tag{3}$$

$$\sum_{k\in K}\sum_{i\in C_2}\sum_{h\in S_3\cup S_4} s_{ih}x_{ijk} = \sum_{k\in K}\sum_{i\in C_1}\sum_{h\in S_2\cup S_3\cup S_4} s_{ij}x_{ihk} - n_j \ , \forall j \in S_1 \tag{4}$$

$$\sum_{k\in K}\sum_{i\in C_1}\sum_{h\in S_1\cup S_3} s_{ih}x_{ijk} + \sum_{k\in K}\sum_{i\in C_2}\sum_{h\in S_3\cup S_4} s_{ih}x_{ijk} = -n_j \ , \forall j \in S_2 \tag{5}$$

$$\sum_{k\in K}\sum_{i\in C_1}\sum_{h\in S_1} s_{ih}x_{ijk} = \sum_{k\in K}\sum_{i\in C_1}\sum_{h\in S_2\cup S_4} s_{ij}x_{ihk} + \sum_{k\in K}\sum_{i\in C_2}\sum_{h\in S_1\cup S_2} s_{ij}x_{ihk} - n_j \ , \forall j \in S_3 \tag{6}$$

$$\sum_{k\in K}\sum_{i\in C_1}\sum_{h\in S_1\cup S_3} s_{ih}x_{ijk} = \sum_{k\in K}\sum_{i\in C_2}\sum_{h\in S_1\cup S_2} s_{ij}x_{ihk} - n_j \ , \forall j \in S_4 \tag{7}$$

$$\sum_{k\in K}\sum_{i\in C_2}\sum_{j\in S} x_{ijk} = \sum_{i\in C_2}\sum_{j\in S_3\cup S_4} s_{ij} \tag{8}$$

$$\sum_{i\in C} x_{0ik} \le 1 \, , \forall k \in K \tag{9}$$

$$\sum_{k\in K}\sum_{j\in S\cup\{0\}} x_{jik} \le 1 \, , \forall i \in C \tag{10}$$

$$\sum_{i\in C\cup S\cup\{0\}}\sum_{j\in C\cup S\cup\{0\}} x_{ijk}\left(2\sum_{i\in C} x_{0ik} - 1\right) - \sum_{i\in C} x_{0ik} \ge 0 \, , \forall k \in K \tag{11}$$

$$\sum_{j\in S} x_{j0k} = \sum_{i\in C} x_{0ik} \ , k \in K \tag{12}$$

$$\sum_{j\in C\cup S\cup\{0\}} x_{ijk} = \sum_{j\in C\cup S\cup\{0\}} x_{jik} \ , \forall i \in C\cup S\cup\{0\}, \forall k \in K \tag{13}$$

$$\sum_{i\in C}\sum_{j\in S} d_{ij}x_{ijk}\frac{1}{v_C} + \left(\sum_{i\in C}\sum_{j\in S} d_{ji}x_{jik} + \sum_{i\in C} d_{0i}x_{0ik} + \sum_{j\in S} d_{j0}x_{j0k}\right)\frac{1}{v_B} \le T \, , k \in K \tag{14}$$

$$d_{ij}x_{ijk} \le m_i L \, , \forall i \in C, \forall j \in S_1 \cup S_2, \forall k \in K \tag{15}$$

$$d_{ij}x_{ijk} \le (m_i - \beta)L \, , \forall i \in C_1, \forall j \in S_3 \cup S_4, \forall k \in K \tag{16}$$

$$x_{ijk} \in \{0,1\}, \forall i \in C\cup S\cup\{0\}, \forall j \in C\cup S\cup\{0\}\setminus\{i\}, \forall k \in K \tag{17}$$

In this model, the objective function (2) aims to minimize the combination of total driving cost and total riding cost of employees. Constraint (3) removed the infeasible conversion. The seven parts, in turn, correspond to the conversion between vehicle nodes, the conversion between site nodes, the conversion from vehicle nodes to the dispatch center, the conversion from the dispatch center to the site node, and the vehicle nodes to the conversion of the site node where it is located, the conversion from a fully charged vehicle node to a saturated site node with charging piles, and the conversion from

a vehicle station with insufficient power to a node without charging piles. Specifically, constraint (3) is established, which is equivalent to that all seven parts are zero.

Constraints (4)–(7) realize the net number of vehicles transferred in/out of different types of sites. Specifically, constraint (4) corresponds to a saturated site with charging piles, constraint (5) corresponds to an unsaturated site with charging piles, constraint (6) corresponds to a saturated site without charging piles, and constraint (7) corresponds to a saturated site without charging piles, among them, n_j satisfies the condition $\sum_{j\in S} n_j = 0$. Constraint (8) Realize that vehicles with insufficient power parked at sites without charging piles are transferred to sites with charging piles. Constraint (9) means that each employee can be dispatched at most once. Constraint (10) guarantees that each vehicle node can be visited at most once. Constraint (11) ensures that only employees starting from the dispatch center can participate in the dispatch of vehicles, that is, eliminate the sub-loop between the vehicle node and the site node. Constraint (12) guarantees that if an employee leaves the dispatch center, he must return to the dispatch center. Constraint (13) ensures that employees have a balance between the entry and exit of the vehicle node and the site node, that is, if and only when an employee enters a certain node, he must leave the site. Constraint (14) ensures that the total working hours of each employee do not exceed T. Constraint (15) realizes the vehicle's cruising range limit, that is, there is sufficient power to reach the next stop. Constraint (16) realizes the restriction of the remaining power after the vehicle is hoisted to a site without charging piles, that is, to ensure that the vehicle can work normally the next day. Constraint (17) defines all decision variables.

2.2. Harris Hawks optimization (HHO)

HHO is a metaheuristic algorithm based on swarm intelligence, proposed by Heidari [52]. The main inspiration for the algorithm comes from the predation behavior of Harris Hawk. The Harris Hawk is a famous bird of prey in southern Arizona and other regions of the United States. They forage through the coordination of group behaviors. The main strategy for capturing prey is "surprise pounce", that is, the Harris hawk can hide near the prey many times, in a short time, and quickly, waiting for the opportunity. Through teamwork, Harris Hawks can confuse their escaped prey and make them unable to recover their defense capabilities, thereby efficiently capturing tired prey.

2.2.1. Exploration phase

During the exploration phase, Harris Hawks can track and detect prey through their powerful eyes, but sometimes prey is not easy to spot. Therefore, the Harris Hawk determines the habitat position based on the random prey position and the vector difference between the prey position and the center position of the group. The opportunities for both strategies are equal.

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1 |X_{rand}(t) - 2r_2 X(t)| & q \geq 0.5 \\ \left(X_{rabbit}(t) - X_m(t)\right) - r_3\left(LB + r_4(UB - LB)\right) & q < 0.5 \end{cases} \tag{18}$$

where t is the current iteration. $X(t+1)$ represent the position of the agent in the next iteration. $X_{rabbit}(t)$ represent the position of the rabbit, $X(t)$ is the current position vector of the agent. $X_{rand}(t)$ represent a randomly determined Harris hawk position. UB and LB represent the upper and

lower bounds of the search space. $\mathrm{R}_1, \mathrm{r}_2, \mathrm{r}_3$ and r_4 are random numbers inside [0, 1]. $\mathrm{X}_\mathrm{m}(\mathrm{t})$ is the center position of the Harris hawks in the current population, calculated as follows,

$$\mathrm{X}_\mathrm{m}(\mathrm{t}) = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{X}_\mathrm{i}(\mathrm{t}) \tag{19}$$

where $X_i(t)$ indicates the location of each hawk in iteration t. N indicate the all number of Harris Hawks.

2.2.2. Transition from exploration to exploitation

With the continuous movement of the prey, the prey will transition from an energetic state to a tired state. Inspired by this, the Harris Hawk algorithm proposed a prey escape energy mechanism, enabling the algorithm to transition from exploration to development. The energy mechanism of the game is modelled as follows,

$$E = 2E_0\left(1-\frac{t}{T}\right) \tag{20}$$

where E means the escaping energy of the prey. T indicates the maximum number of iterations. E_0 is the initial state of prey escape energy. E_0 varies randomly between -1 and 1 in each iteration. when the value of E_0 decreases from 0 to −1, the energy of the rabbit gradually decreases. When the value of E_0 increases from 0 to 1, the energy of the rabbit is increasing. $|E| \geq 1$ means that the Harris hawk is still looking for prey in the area, and $|E| < 1$ means that the Harris hawk starts to use the area to attack its prey. The time-dependent trajectory E of is presented in Figure 1.

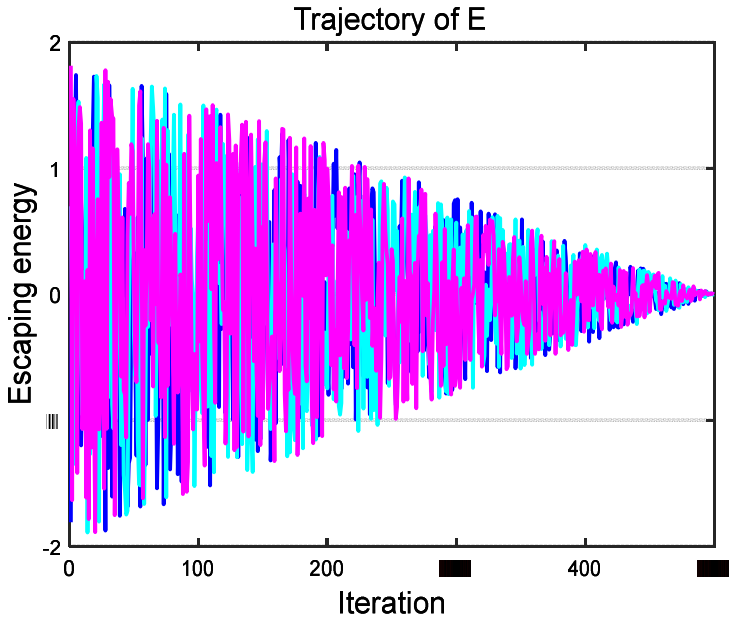

Figure 1. Behavior of *E* during three runs and 500 iterations.

2.2.3. Exploitation phase

In the development phase, Harris hawk will use a surprise strategy to attack its prey. When the prey tries to escape, it may succeed or fail. Even so, Harris Hawk will still adopt other strategies to capture prey. According to the energy of the prey $|E|$ when it escapes and the chance of the prey avoiding the attack, the Harris Hawk will have four attack strategies at this stage. Specifically,

Soft besiege: When $r \geq 0.5$ and $|E| \geq 0.5$, the prey still has enough energy and tries to escape the pursuit of Harris Hawk through some random misleading jumps, but in the end, it cannot. During these attempts, the Harris Hawk gradually surrounded it, making the rabbit more exhausted, and then launched a surprise attack. The mathematical description is as follows,

$$X(t+1) = \Delta X(t) - E\left|JX_{rabbit}(t) - X(t)\right| \tag{21}$$

$$\Delta X(t) = X_{rabbit}(t) - X(t) \tag{22}$$

where ΔX(t) is the difference between the rabbit's position vector and the current position of the Harris Hawk in iteration t. $J = 2(1\text{-}r_5)$ show the strength of the prey random jump, and r_5 is a random number within $(0, 1)$. The J value changes randomly during each iteration, which can simulate the nature of rabbit movement.

Hard besiege: When $r \geq 0.5$ and $|E| < 0.5$, it shows that the prey is tired and the energy to escape is very low. Therefore, Harris Hawk finally executed the act of surprise pounce. The current positions are updated using Eq (23):

$$X(t+1) = X_{rabbit}(t) - E\left|\Delta X(t)\right| \tag{23}$$

Soft besiege with progressive rapid dives: When $r < 0.5$ and $|E| \geq 0.5$, the rabbit has enough energy to escape, so the Harris Hawk needs a soft encirclement strategy before the raid. This process is more complicated and smarter than before. To mathematically simulate the escaping patterns of the prey, the concept of Levi flight (LF) was referenced in the HHO algorithm [64]. The mathematical formula for this stage is modeled as follows,

$$Y = X_{rabbit}(t) - E\left|JX_{rabbit}(t) - X(t)\right| \tag{24}$$

$$Z = Y + S \times LF(D) \tag{25}$$

$$LF(x) = 0.01 \times \frac{\mu \times \sigma}{|v|^{\frac{1}{\beta}}},\ \sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{\frac{1}{\beta}} \tag{26}$$

$$X(t+1) = \begin{cases} Y & if\ F(Y) < F(X(t)) \\ Z & if\ F(Z) < F(X(t)) \end{cases} \tag{27}$$

where D is the dimension of problem and S is a random vector by size $1 \times \mathrm{D}$. LF is the levy flight function, which is calculated using Eq (26) [65].

Hard besiege with progressive rapid dives: When $\mathrm{r} < 0.5$ and $|E| < 0.5$, the rabbit has not had enough energy to escape due to long-term movement. The Harris hawk rounds up based on the position of the prey and the center of the group to reduce the distance between the group and the prey. If the raid fails (the fitness is not improved), perform a random walk, if the walk fails, return to the original place. The formula is defined as follows,

$$X(t+1) = \begin{cases} Y & if\ F(Y) < F(X(t)) \\ Z & if\ F(Z) < F(X(t)) \end{cases} \tag{28}$$

$$Y = X_{rabbit}(t) - E\left|JX_{rabbit}(t) - X_m(t)\right| \tag{29}$$

$$Z = Y + S \times LF(D) \tag{30}$$

3. Our proposed algorithm

This section proposes our algorithm to solve the shared electric vehicle scheduling model considering the charging schedule. We will introduce it in three parts. The first part is to encode the problem to facilitate the model solution. On the other hand, there is also a corresponding decoding method to explain the optimal solution. The second part describes the algorithm, including the transfer function used and the specific algorithm steps. The third part is an explanation of how to deal with model constraints.

3.1. Encoding and decoding

Coding is the first step to solving the problem, which is very important. It is known that there are five employees. Each employee needs to ride a bicycle from the dispatch center to the station where the car needs to be transferred first, then drive the car to the transfer station that meets the constraints, and then ride a bicycle from the station to the next need call out the station of the vehicle, and so on, until the deployment task assigned to the employee is completed, and then return to the dispatch center by bicycle. Specifically, suppose we have 1 S_1, 1 S_2, 1 S_3, and 1 S_4, a total of four sites. Therefore, we define the code to include two parts: location and path, as shown below,

Location: 1 0 1 0 | 1 0 0 1 | 0 0 0 0 | 1 1 1 1 | 0 1 0 1

Path: 1 3 0 0 | 4 1 0 0 | 0 0 0 0 | 3 1 4 2 | 4 2 0 0

where the location vector indicates whether the employee has been to the station or not. If the employee has been there, it is represented by 1, otherwise, it is 0. Its size is $\mathrm{S} \times \mathrm{K}$. The path vector is used to assist the location vector to record the employee's visit to the station footprint, its size is as large as the position vector.

The decoding process is the inverse process of the encoding process. The code given above can be decoded in this way. First, look at the location code. The first part means that the first employee starts from the dispatch center and calls out the vehicle from station 1 (saturated station with charging

piles), and then drives the vehicle to station 3 (saturated sites without charging piles), completes all assigned dispatching tasks and return to the dispatch center. The path code reflects the order in which employees visit the site. The second employee departs from the dispatch to station 4 (unsaturated station without charging piles) and transfers the vehicle to station 1 (saturated station with charging piles), completes the dispatched task, and rides back to the dispatched center. The third employee is not assigned a scheduling task and does not need to work. The fourth employee starts from the dispatch center to transfer the vehicle to the No. 3 station and then to the No. 4 station to transfer the vehicle to the No. 2 station, complete the dispatched task and return to the dispatch center. The fifth employee starts from the dispatch center to call out the vehicle at No. 4 station, then drives the car to No. 2 station to transfer in the vehicle, completes all the dispatch tasks, and returns to the dispatch center. Since then, the decoding has been completed.

3.2. DHHO algorithm

The basic HHO is a continuous optimization algorithm, and the coding method of the shared electric vehicle scheduling model considering charging scheduling is binary coding. Therefore, the original HHO cannot be directly applied to practical problems, which brings challenges to our research. Consequently, it is necessary to find an efficient transfer function to convert the continuous optimization algorithm into binary form. In [66], a binary version of the Harris Eagle optimization algorithm is proposed, in which the specific description of the transfer function is shown in Table 3. The transfer function image is shown in Figures 2–9. The pseudo-code definitions of the detailed steps of the algorithm are shown in Algorithms 1–3. The detailed flow chart of the DHHO approach for solving the SEVS model is shown in Figure 10.

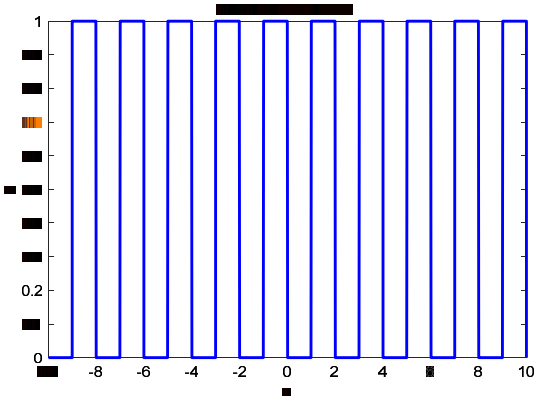

Figure 2. Transfer function T1.

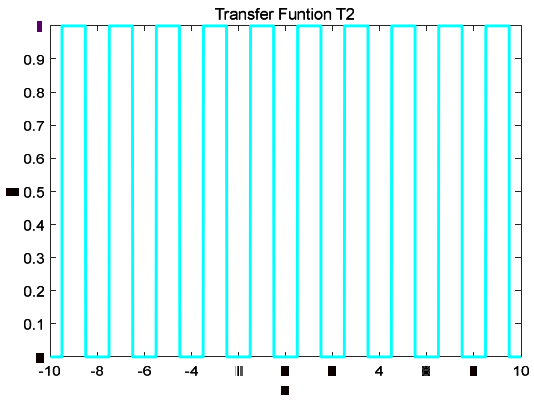

Figure 3. Transfer function T2.

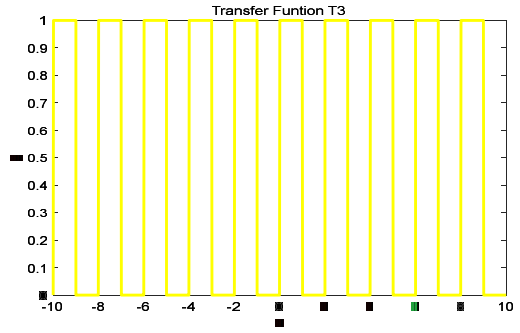

Figure 4. Transfer function T3.

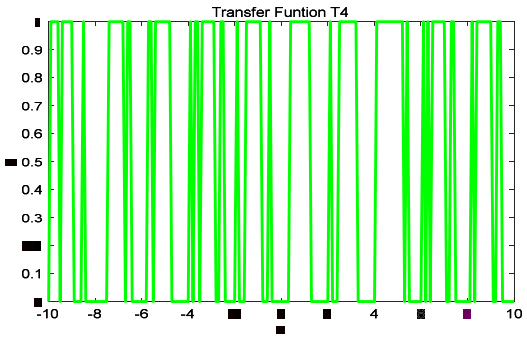

Figure 5. Transfer function T4.

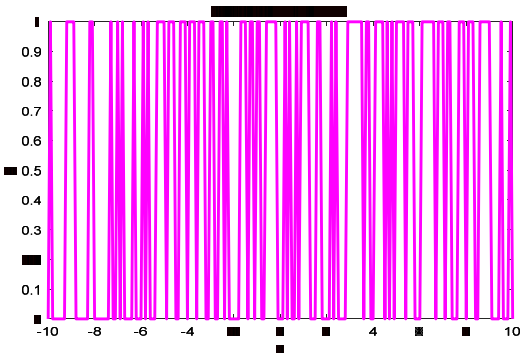

Figure 6. Transfer function T5.

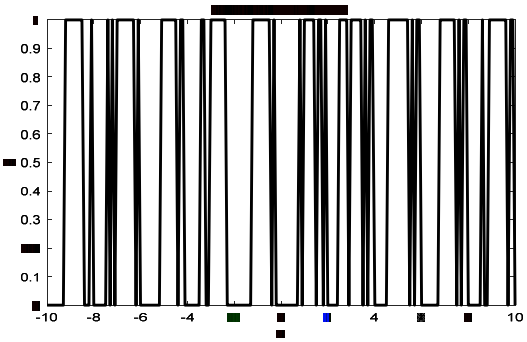

Figure 7. Transfer function T6.

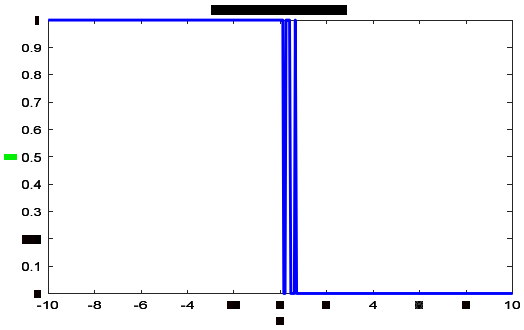

Figure 8. Transfer function T7.

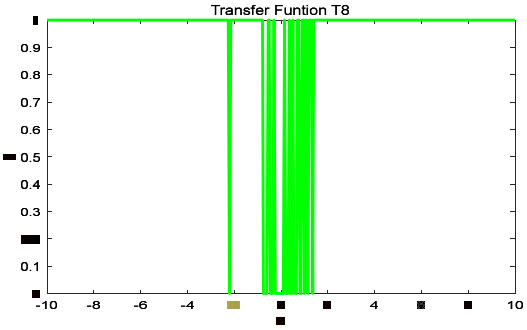

Figure 9. Transfer function T8.

Table 3. Transfer functions.

Name	Transfer function	Name	Transfer function

T1(DHHO1)	binaryVector = mod(floor(continuous Vector),2);	T5(DHHO5)	if rand < 0.5
			binaryVector = mod(ceil(continuousVector),2);
			else
			binaryVector = mod(floor(continuousVector),2);
			end

T2(DHHO2)	binaryVector = mod(round(mod(continuousVector,2)),2);
	;
		T6(DHHO6)	if rand < 0.5
			binaryVector = mod(round(continuousVector),2);
			else
			binaryVector = mod(floor(continuousVector),2);
			end

T3(DHHO3)	binaryVector = mod(ceil(continuousVector),2);	T7(DHHO7)	$MSig(x)=\frac{1}{1+e^{-10(x(t+1)-0.5)}}$

T4(DHHO4)	if rand < 0.5
	binaryVector = mod(ceil(continuousVector),2);
	else
	binaryVector = mod(round(continuousVector),2)
	end
		T8(DHHO8)	$\tanh(x)=\frac{\left(e^{2x(t+1)}-1\right)}{\left(e^{2x(t+1)}+1\right)}$

Algorithm 1 Pseudo-code of DHHO algorithm

1. **Inputs:** $N, Iter, d, m, n, s$
2. **Outputs:** Location, path vector and Minimum cost
3. **Initialize:** $X_i, i = 1,2,3,4, \ldots, N$
4. **While** (stopping condition is not met) **do**
5. Binarization using $TF_{8(X_i)}$
6. Calculate the cost values of hawks with $Fitness(X_i)$
7. Set X_{best} as the location of best
8. **for** (each hawk (X_i)) **do**
9. Update the initial energy E_0 and jump strength J
10. Update the E using Eq (20)

Continued on next page

```
        if (|E| ≥ 1) then
            Update the location vector using Eq (18)
        if (|E| < 1) then
            if (r ≥ 0.5 and |E| ≥ 0.5) then
                Update the location vector using Eq (21)
            else if (r ≥ 0.5 and |E| < 0.5) then
                Update the location vector using Eq (23)
            else if (r < 0.5 and |E| ≥ 0.5) then
                With progressive rapid dives update the location vector using Eq (27)
            else if (r < 0.5 and |E| < 0.5) then
                With progressive rapid dives update the location vector using Eq (28)
            End if
        End if
    End for
End While
Return X_best
```

Algorithm 2 Pseudo-code of $TF_{8(X_I)}$ algorithm

```
Inputs: X_i
Outputs: X_i after discrete
While (length of X_i) do
    Binarization using transfer function
End While
Return X_i
```

Algorithm 3 Pseudo-code of $\mathrm{Fitness}(X_I)$ algorithm

```
Inputs: X_I, K, S_1, S_2, S_3, S_4, L, α, β, γ, δ, v_C, v_B, T, MM, SS
Outputs: X_I, X_route_I, COST_i
If (the number of 1 is an odd number in X_i)
    Select one of them j randomly at X_i
    If (X_j equals 1)
        X_j equals 0
    Else
        X_j equals 1
    End
End
For (length of X_i as ii) do
    For (length of X_i as jj) do
```

Continued on next page

If ($X_i^{ii} == 1 \& X_i^{jj} == 1 \& ii \neq jj$) then
  If ($ii \leq S_1 \& jj > S_1 \& jj \leq S\_station$) then
    For (number of car) do
      If ($SS(tt) \leq S_1 \& MM(tt) \geq \alpha \& \sim flag(tt)$) then
        If ($DIS(ii+1, jj+1) \leq (MM(tt)\text{-}\beta) \times L$) then
          Modify the station where the vehicle is located;
          Modify site redundancy;
          Mark that the vehicle has been dispatched;
          Record transfer out and transfer in sites;
          Calculate cycling distance and driving distance;
        End if
      End if
    End for
  End if
  If ($ii > S_1 \& ii \leq (S_1 + S_2)$)
    Nothing to do;
  End if
  If ($ii > (S_1 + S_2) \& ii \leq (S_1 + S_2 + S_3)$)
If ($\left(\left(jj > S_1 \& jj \leq (S_1 + S_2)\right) | \left(jj > (S_1 + S_2 + S_3)\right) \& jj \leq S_{station}\right)$)
    For (number of car) do
If ($\left(SS(tt) > (S_1 + S_2) \& SS(tt) \leq (S_1 + S_2 + S_3) \& MM(tt) \geq \alpha \& \sim flag(tt)\right)$)
      If ($DIS(ii+1, jj+1) \leq (MM(tt)\text{-}\beta) \times L$) then
        Modify the station where the vehicle is located;
        Modify site redundancy;
        Mark that the vehicle has been dispatched;
        Record transfer out and transfer in sites;
        Calculate cycling distance and driving distance;
      End if
    End if
    End for
    Else If ($jj \leq S_1 | \left(jj > S_1 \& jj \leq (S_1 + S_2)\right)$) then
    For (number of car) **do**
  If
($SS(tt) > (S_1 + S_2) \& SS(tt) \leq (S_1 + S_2 + S_3) \& MM(tt) < \alpha \& \sim flag(tt)$
      If ($DIS(ii+1, jj+1) \leq MM(tt) \times L$) then
        Modify the station where the vehicle is located;
        Modify site redundancy;
        Mark that the vehicle has been dispatched;

Continued on next page

Record transfer out and transfer in sites;
Calculate cycling distance and driving distance;
End if
End if
End for
End if
End if
If

$$((ii > (S_1 + S_2 + S_3)\&ii \leq S\_station)\&\left(jj \leq S_1|\left(jj > S_1\&jj \leq (S_1 + S_2)\right)\right))$$

For (number of car)
If

$(SS(tt) > (S_1 + S_2 + S_3)\&SS(tt) \leq S\_station\&MM(tt) < \alpha\&\sim flag(tt))$

If $(DIS(ii + 1, jj + 1) \leq MM(tt) \times L)$ then
Modify the station where the vehicle is located;
Modify site redundancy;
Mark that the vehicle has been dispatched;
Record transfer out and transfer in sites;
Calculate cycling distance and driving distance;
End if
End if
End for
End if
End if
End for
End for
Calculate time if timeout penalty;
Calculate $COST_i$;
Return $X_I, X_{route_I}, COST_I$;

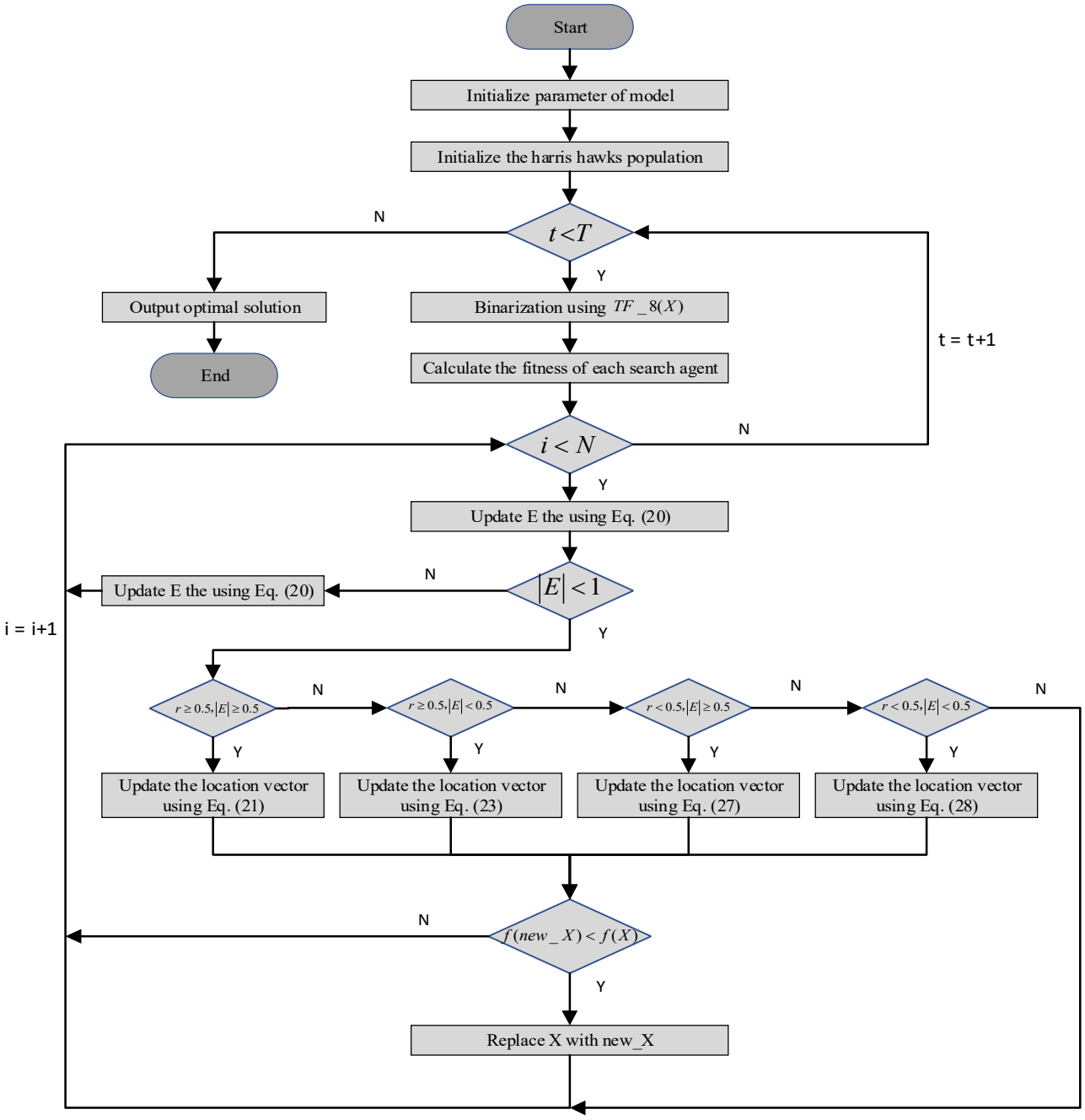

Figure 10. The flow chart of DHHO.

3.3. Constrain handling

The constraint processing of the model is generally divided into three methods, namely: directly processing the constraints, that is, adding constraints in the encoding process; checking the constraints during the calculation process; and processing constraints through the penalty function [66]. According to the characteristics of the model, this paper adopts a mixed constraint processing method. Constraints (4)–(8) and (14) adopt the penalty function method, constraints (9)–(13) are processed in the encoding process, and constraints (15) and (16) are checked during the calculation process.

3.4. DHHO algorithm complexity analysis

As can be seen from Algorithm 1, the complexity of the DHHO algorithm consists of three parts: initialization of model and algorithm parameters, fitness calculation, and Harris Hawk population update. The complexity of the initialization phase mainly depends on the number of Harris Hawk, so the complexity is $O(N)$. From Algorithms 2 and 3, the complexity of fitness calculation and Harris Hawk population update is $O(T \times N \times D \times D \times C) + O(T \times N \times D \times D)$, where C is the number of cars, D is the vector dimension of Harris Hawk positions, and T is the maximum number of iterations. Overall, the computational complexity of DHHO is $O(N \times (T \times D \times D \times (C + 1) + 1))$.

4. Experimental results and discussion

4.1. Parameter setting

The experiments were used Matlab 2019b and were run on an Intel Core i5 machine, with a 1.80 GHz and 8 GB of RAM. In this work, the number of stations of the platform in the area usually belongs to the interval $[4, 12]$. Randomly generate 15 groups of examples, with 5 cases in each of the small, medium, and large groups. The main parameter settings of the experiment are shown in Table 4, and the specific information of the number of stations of each type and the number of vehicles of each type in each calculation example is shown in Table 5.

Table 4. Parameter setting.

Parameter	Initial value
N	30
$Iter$	500, 3000, 5000
L	150 km
v_C	25 km/h
v_B	15 km/h
γ	1.5 $/km
δ	0.5 $/km
T	5 h
K	5
α	0.7
β	0.7
m_i	[0.5, 1] randomly generated
n_j	small [-2, 2], medium [-4, 4], large [-8, 8] randomly generated
S	[4, 12]
d_{ij}	[1, 3] km

Table 5. Specific information about the number of stations of each type and the number of vehicles of each type.

Case scale	Case	Number of vehicles		Number of stations			
		|C1|	|C2|	|S1|	|S2|	|S3|	|S4|
small	1	7	3	1	1	1	1
	2	8	2	0	1	2	1
	3	7	3	1	1	1	1
	4	4	6	1	1	1	1
	5	6	4	1	2	1	0
medium	6	26	14	2	3	1	2
	7	27	13	2	1	1	4
	8	28	12	2	1	3	2
	9	26	14	3	2	1	2
	10	19	21	3	2	1	2
large	11	65	55	4	1	2	5
	12	80	40	3	4	3	2
	13	71	49	2	3	2	5
	14	76	44	3	3	4	2
	15	82	38	2	5	3	2

4.2. Performance measures

To evaluate the performance of our proposed algorithm on the model, we use the best value, mean, and standard deviation to analyze the results.

Best value: It is the best obtained values over several runs, its definitions as follows:

$$B\text{est} = \min\{F_1, F_2, ..., F_N\} \tag{31}$$

where N is the number of algorithm runs, and $\mathrm{F_i}$ is the best value.

Mean value: It is an average of the best obtained values over several runs, it is calculated as:

$$AVG = \frac{1}{N}\sum_{i=1}^{N} F_i \tag{32}$$

Standard deviation: It is used to indicate the stability of the algorithm to produce the best value over different runs:

$$STD = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(F_i - AVG)^2} \tag{33}$$

4.3. Small scale example experiments

In the experiments, we adopt eight transfer functions mentioned in the literature, the T1, T2, T3, T4, T5, and T6 transfer functions are proposed by [67,68], and the T7 and T8 transfer functions are proposed by [69]. We combined the transfer function with HHO and named DHHOT1, DHHOT2, DHHOT3, DHHOT4, DHHOT5, DHHOT6, DHHOT7, and DHHOT8, respectively. The number of small-scale data iterations is 500 times, and the experimental results are shown in Table 6. In Table 6, the small-scale data is set to verify the algorithm's effectiveness. As the experiment shows, each calculation example can find the optimal value on the algorithm, and each algorithm has a good performance on each calculation example, and the variance is 0.

Table 6. Experimental results of small-scale data with 500 iterations and a population size of 30.

Case	Standard	DHHO1	DHHO2	DHHO3	DHHO4	DHHO5	DHHO6	DHHO7	DHHO8
	Best	25.22	25.22	25.22	25.22	25.22	25.22	25.22	25.22
1	Avg	25.22	25.22	25.22	25.22	25.22	25.22	25.22	25.22
	Std	0	0	0	0	0	0	0	0
	Best	12.25	12.25	12.25	12.25	12.25	12.25	12.25	12.25
2	Avg	12.25	12.25	12.25	12.25	12.25	12.25	12.25	12.25
	Std	0	0	0	0	0	0	0	0
	Best	15.38	15.38	15.38	15.38	15.38	15.38	15.38	15.38
3	Avg	15.38	15.38	15.38	15.38	15.38	15.38	15.38	15.38
	Std	0	0	0	0	0	0	0	0
	Best	12.51	12.51	12.51	12.51	12.51	12.51	12.51	12.51
4	Avg	12.51	12.51	12.51	12.51	12.51	12.51	12.51	12.51
	Std	0	0	0	0	0	0	0	0
	Best	12.53	12.53	12.53	12.53	12.53	12.53	12.53	12.53
5	Avg	12.53	12.53	12.53	12.53	12.53	12.53	12.53	12.53
	Std	0	0	0	0	0	0	0	0

Note: Bold data is the best performing data.

4.4. Medium scale example experiments

In Table 7, for the medium-sized data set, each algorithm of Examples 8 and 10 calculated the optimal value. The individual algorithms of Examples 6, 7, and 9 calculated the optimal value. Algorithms DHHO4, DHHO5, DHHO6, and DHHO8 have calculated the optimal values in Example 6, and the algorithm DHHO8 has the lowest mean and variance. Algorithms DHHO1, DHHO3, DHOO5, DHHO6, and DHHO8 have calculated the optimal values in Example 7. The algorithm DHHO8 is also the lowest in terms of mean and variance. Algorithms DHHO2, DHHO4, DHHO5, DHOO6, DHHO7, and DHHO8 have calculated the optimal values in Example 9. The algorithm DHHO8 has the lowest average value, and the variance is slightly higher than DHHO4. Overall, DHHO8 performed more prominently.

Table 7 Experimental results of medium-scale data with 3000 iterations and a population size of 30.

Case	Standard	DHHO1	DHHO2	DHHO3	DHHO4	DHHO5	DHHO6	DHHO7	DHHO8
	Best	34.99	35.57	34.99	**34.95**	**34.95**	**34.95**	34.99	**34.95**
6	Avg	37.56	38.02	38.99	35.76	35.71	35.80	36.26	**35.39**
	Std	2.20	1.83	2.65	0.75	0.60	1.10	1.12	**0.43**
	Best	**25.93**	26.20	**25.93**	26.20	**25.93**	**25.93**	26.20	**25.93**
7	Avg	28.60	27.22	27.88	26.65	26.42	26.41	27.02	**26.16**
	Std	1.95	1.03	1.61	0.61	0.53	0.45	0.88	**0.10**
	Best	**36.46**	**36.46**	**36.46**	**36.46**	**36.46**	**36.46**	**36.46**	**36.46**
8	Avg	41.61	41.04	41.58	39.67	39.66	**39.19**	**39.19**	39.27
	Std	2.93	2.62	3.45	1.55	1.74	1.83	1.75	1.68
	Best	32.56	**32.11**	32.56	**32.11**	**32.11**	**32.11**	**32.11**	**32.11**
9	Avg	35.94	34.78	36.30	32.69	32.68	32.82	34.23	**32.45**
	Std	2.26	1.71	3.82	**0.31**	0.55	0.96	4.36	0.33
	Best	**30.92**	**30.92**	**30.92**	**30.92**	**30.92**	**30.92**	**30.92**	**30.92**
10	Avg	35.01	35.04	34.15	31.97	31.55	32.20	32.18	**30.99**
	Std	2.60	2.27	2.68	1.82	0.75	1.25	1.09	**0.15**

Note: Bold data is the best performing data.

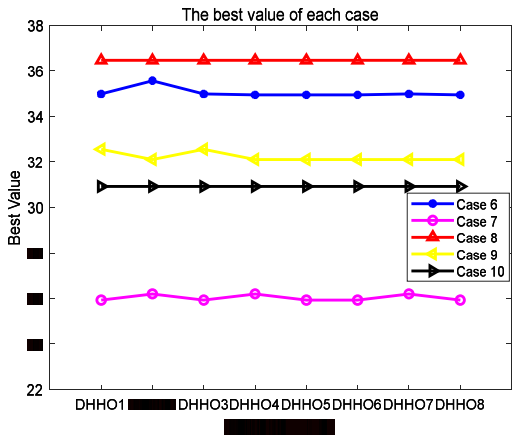

Figure 11. Best value for medium-scale studies.

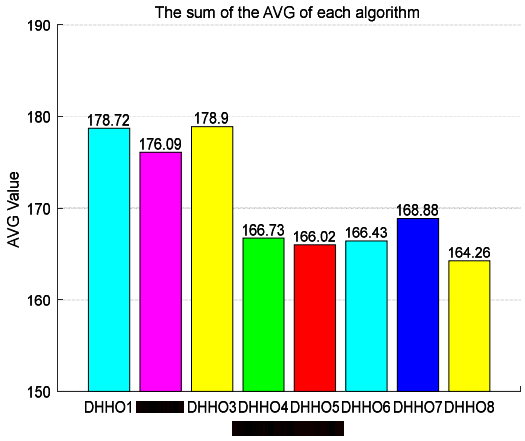

Figure 12. Mean value for medium-scale studies.

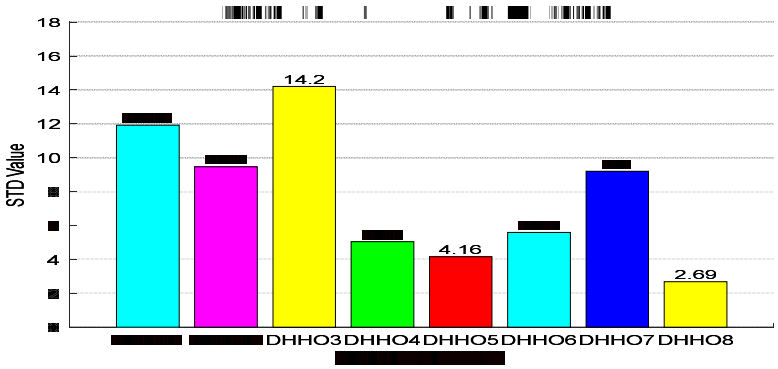

Figure 13. STD value for medium-scale studies.

4.5. Large scale example experiments

In Table 8, for large-scale data sets, not every algorithm can find the optimal value for every calculation example. Algorithms DHHO3, DHHO5, DHHO7, and DHHO8 can all find the optimal value in Example 11, and the average value of algorithm DHHO5 is also the smallest. Algorithm DHHO5 is the only algorithm that can calculate the optimal value in Example 12, and the average value is slightly higher than that of Algorithm DHHO7. Among them, the average value calculated by Algorithm DHHO7 in Example 12 is the smallest. Algorithm DHHO5 and algorithm DHHO7 found the optimal value in example 13, and algorithm DHHO6 found the lowest average value in example 13. Algorithm DHHO5 performed well in Example 14. Not only the optimal value was obtained, but the average value was also the lowest among the eight algorithms. The algorithm DHHO7 calculated the optimal value in Example 15, and the algorithm DHHO5 achieved the lowest mean value and the lowest variance. In general, the algorithm DHHO5 performs better.

Table 8. Experimental results of large-scale data with 5000 iterations and a population size of 30.

Case	Standard	DHHO1	DHHO2	DHHO3	DHHO4	DHHO5	DHHO6	DHHO7	DHHO8
	Best	110.56	107.63	**102.19**	102.59	**102.19**	103.41	**102.19**	**102.19**
11	Avg	123.85	119.12	127.95	112.89	**110.13**	114.79	110.59	115.91
	Std	10.20	8.95	31.39	**6.94**	7.17	9.49	7.07	9.80
	Best	115.20	125.03	125.00	112.86	**108.26**	118.37	116.40	121.06
12	Avg	144.14	142.95	141.20	132.49	130.81	137.23	**130.10**	135.67
	Std	14.60	11.69	11.76	11.24	10.97	12.22	9.59	**9.07**
	Best	140.76	131.83	133.12	130.13	**124.58**	130.32	**124.58**	132.82
13	Avg	168.49	180.06	166.95	156.16	157.91	**153.85**	154.64	160.00
	Std	25.53	31.00	25.36	**14.42**	18.05	16.30	14.66	15.49
	Best	168.13	159.34	153.99	151.62	**145.83**	151.51	153.38	154.20
14	Avg	186.90	189.78	189.14	179.92	**170.29**	181.48	177.14	177.70
	Std	**13.34**	17.76	20.63	19.76	16.64	23.76	17.33	17.58
	Best	166.02	177.01	152.29	152.36	155.03	152.50	**151.31**	151.75
15	Avg	214.61	215.83	213.03	195.26	**175.45**	189.67	190.44	187.27
	Std	25.01	27.87	33.75	34.97	**19.90**	20.62	33.68	23.04

Note: Bold data is the best performing data.

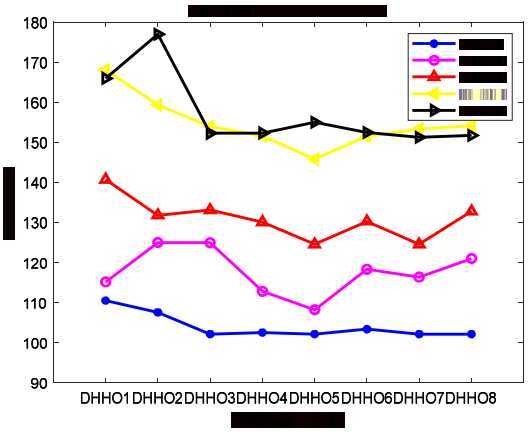

Figure 14. Best value for large-scale studies.

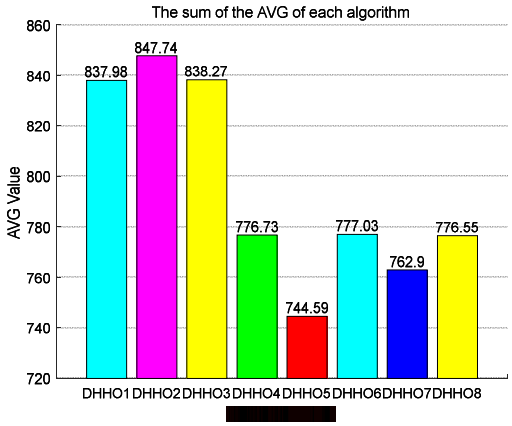

Figure 15. Mean value for large-scale studies.

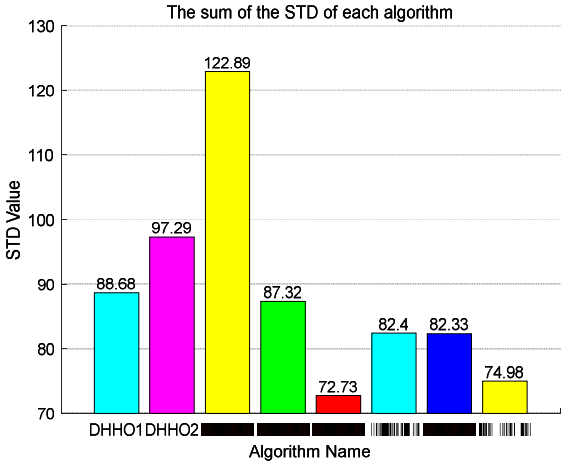

Figure 16. STD value for large-scale studies.

4.6. Friedman test

The Friedman test is based on the ranking of the algorithms. We used this method to rank the performance of the eight algorithms and finally calculated the average order. In Table 9, we can easily see that DHHO8 achieved the average of the minimum values and ranked first. In Table 10, DHHO5 performs well on the large-scale dataset, outperforming other algorithms, achieving the minimum mean, and ranking first.

Table 9. Ranking produced by Friedman test comparing DHHO1, DHHO2, DHHO3, DHHO4, DHHO5, DHHO6, DHHO7, DHHO8 over medium data sets.

	DHHO1	DHHO2	DHHO3	DHHO4	DHHO5	DHHO6	DHHO7	DHHO8
Friedman's mean rank	7.20	6.60	7.20	3.60	2.60	3.20	4.20	**1.40**
Rank	7	6	7	4	2	3	5	**1**

Note: Bold indicates a difference.

Table 10. Ranking produced by Friedman test comparing DHHO1, DHHO2, DHHO3, DHHO4, DHHO5, DHHO6, DHHO7, DHHO8 over large data sets.

	DHHO1	DHHO2	DHHO3	DHHO4	DHHO5	DHHO6	DHHO7	DHHO8
Friedman's mean rank	7.00	7.40	6.60	3.60	**1.80**	3.60	2.20	3.80
Rank	6	7	5	3	**1**	3	2	4

Note: Bold indicates a difference.

4.7. Wilcoxon rank-sum test

To evaluate the performance of the algorithm, a Wilcoxon rank-sum test was performed to obtain a statistical test. We use statistical methods to analyze whether the results of different discrete Harris Eagle algorithms differ. The significance level for nonparametric tests was 5%. Table 9 presents a statistical analysis using the Wilcoxon rank-sum test to compare the results obtained for each discrete version of the Harris Eagle algorithm. In general, *p*-values greater than 0.05 can be considered as the alternative hypothesis is accepted; otherwise, the null hypothesis.

As shown in Table 11, DHHO8 performs better than other algorithms in the medium-scale examples, Examples 6 and 10. In Example 7, except for DHHO6, DHHO8 outperforms other algorithms. In Example 8, DHHO8 is better than DHHO1, DHHO2, and DHHO3. Excellent performance in example 9, except for DHHO5 and DHHO6. As shown in Table 12, among the large-scale examples, DHHO5 outperforms other algorithms in Example 11, except for DHHO6 and DHHO7. In Example 12, the performance of DHHO5 is better than that of DHHO1, DHHO2, DHHO3, and DHHO6. In Examples 13 and 14, the performance of the algorithm DHHO5 is better than that of DHHO1, DHHO2, DHHO3, and DHHO8. Example 15 outperforms other algorithms except for DHHO7. In general, DHHO5 is valid for other algorithms.

Table 11. *P*-values produced by Wilcoxon's rank-sum test comparing DHHO8 VS DHHO1, DHHO2, DHHO3, DHHO4, DHHO5, DHHO6, DHHO7 over medium data sets.

DHHO8 vs	DHHO1	DHHO2	DHHO3	DHHO4	DHHO5	DHHO6	DHHO7
6	1.12e-05	4.91e-08	5.49e-06	1.60e-03	1.97e-02	1.50e-03	3.17e-04
7	4.98e-07	6.39e-06	6.96e-05	5.42e-04	6.40e-03	**6.82e-02**	2.27e-06
8	5.11e-04	1.70e-03	1.12e-04	**3.68e-01**	**3.60e-01**	**1.20e-01**	**4.14e-01**
9	3.13e-05	1.43e-06	1.05e-04	9.00e-03	**5.06e-02**	**6.26e-02**	8.45e-04
10	5.00e-07	9.58e-07	6.15e-08	5.10e-03	2.41e-02	1.94e-04	5.21e-04

Note: Bold indicates a difference.

Table 12. *P*-values produced by Wilcoxon's rank-sum test comparing DHHO5 VS DHHO1, DHHO2, DHHO3, DHHO4, DHHO6, DHHO7, DHHO8 over large data sets.

DHHO5 vs	DHHO1	DHHO2	DHHO3	DHHO4	DHHO6	DHHO7	DHHO8
11	1.30e-03	2.35e-06	4.54e-06	5.08e-04	**9.62e-02**	**6.01e-01**	1.55e-02
12	3.05e-04	1.30e-03	2.75e-02	**3.94e-01**	4.11e-02	**9.35e-02**	**2.29e-01**
13	1.30e-02	1.14e-02	4.53e-02	**7.87e-01**	**2.39e-01**	**2.28e-01**	1.79e-02
14	4.68e-05	1.79e-04	1.81e-05	**8.83e-02**	**9.62e-02**	**6.17e-01**	2.94e-02
15	8.29e-05	1.20e-03	1.29e-04	6.00e-03	8.40e-03	**9.35e-02**	3.37e-02

Note: Bold indicates a difference.

4.8. Results discussion

During the experiment, we use 8 transfer functions to convert the continuous HHO into the discrete HHO, which is used to solve the shared electric vehicle dispatching model considering the charging dispatch. We analyze the performance of eight transfer functions on three datasets of varying sizes. On a small-scale data set, eight discrete HHOs can obtain the optimal solution, which proves the effectiveness of the algorithm. On medium-sized datasets, some algorithms perform well, such as DHHO8. Notably, in Examples 8 and 10, each algorithm can calculate the optimal value. The reason for the analysis may be that the complexity of the randomly generated datasets in Examples 8 and 10 is not high, and it is relatively easy to get the optimal value. On large-scale datasets, the DHHO5 algorithm performs well, and four out of five examples calculate the optimal value. Consider that if the number of iterations of the algorithm is reduced, then the performance of the DHHO5 algorithm may be more obvious compared to other algorithms. It is clear from the experimental results that the use of different transfer functions by the algorithm in datasets of different sizes impacts the model performance. It may well be that other transfer functions lead to different population distributions, and a good transfer function can lead to a more diverse population distribution.

5. Conclusions and future works

In this paper, a discrete DHHO algorithm is proposed to solve the shared vehicle scheduling model considering the charging schedule, and the influence of the transfer function on the algorithm is analyzed. The DHHO algorithm finds the optimal solution through coding and decoding and discovers the scheduling scheme that minimizes the objective function, which is the cost of all employees to complete the scheduling task. In the experiment, three data sets of different scales are used for verification. In the small-scale data set, there is no significant difference in the experimental results of the eight transfer functions. DHHO8 performs well in medium-sized datasets. DHHO5 outperforms the other seven transfer functions in large-scale datasets. Experiments show that the transfer function has a certain influence on the algorithm in datasets of different scales. In future work, we plan to modify the shared vehicle dispatching model considering charging scheduling into a multi-objective model. Since the constraints (4)–(8) in the model are in an ideal situation, the quality of the results depends on the speed of the randomly generated data set. This study adopts the penalty function to deal with these constraints. In future research, the constraints (4)–(8) will also be considered as the

objective function. At the same time, the two objectives of optimization, the minimum cost and the balance of the vehicle at each station, will be optimized. This goal makes the model more general.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (U21A20464, 62066005), Project of the Guangxi Science and Technology (AD21196006).

Conflict of interest

The authors declare no conflict of interest.

References

1. R. Mounce, J. D. Nelson, On the potential for one-way electric vehicle car-sharing in future mobility systems, *Trans. Res. Part A: Policy Pract.*, **120** (2019), 17–30. https://doi.org/10.1016/j.tra.2018.12.003
2. F. Ferrero, G. Perboli, M. Rosano, A. Vesco, Car-sharing services: an annotated review, *Sustainable Cities Soc.*, **37** (2018), 501–518. https://doi.org/10.1016/j.scs.2017.09.020
3. M. Nourinejad, M. J. Roorda, Carsharing operations policies: a comparison between one-way and two-way systems, *Transportation*, **42** (2015), 497–518. https://doi.org/10.1007/s11116-015-9604-3
4. J. Firnkorn, M. Müller, What will be the environmental effects of new free-floating car-sharing systems? The case of car2 go in Ulm, *Ecol. Econ.*, **70** (2011), 1519–1528. https://doi.org/10.1016/j.ecolecon.2011.03.014
5. J. H. Holland, Genetic algorithms, *Sci. Am.*, 1992. https://doi.org/10.1038/scientificamerican0792-66
6. R. Storn, Differential evolution research–trends and open questions, in *Advances in Differential Evolution*, **143** (2008),1–31. https://doi.org/10.1007/978-3-540-68830-3_1
7. I. Rechenberg, Evolutionary strategy, *Comput. Intell.: Imitating Life*, 1994.
8. G. B. Fogel, Evolutionary programming, in *Handbook of Natural Computing*, Springer, Berlin, 2011.
9. A. V. Sebald, L. J. Fogel, Evolutionary programming, *Evol. Program.*, (1994), 1–386. https://doi.org/10.1142/9789814534116
10. O. K. Erol, I. Eksin, A new optimization method: Big Bang–Big Crunch, *Adv. Eng. Software.*, **37** (2006), 106–111. https://doi.org/10.1016/j.advengsoft.2005.04.005
11. E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, *Inf. Sci.*, **179** (2009), 2232–2248. https://doi.org/10.1016/j.ins.2009.03.004
12. A. Kaveh, S. Talatahari, A novel heuristic optimization method: charged system search, *Acta Mech.*, **213** (2010):267–289. https://doi.org/10.1007/s00707-009-0270-4
13. M. H. Tayarani-N, M. R. Akbarzadeh-T, Magnetic Optimization Algorithms a new synthesis, in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, (2008), 2659–2664. https://doi.org/10.1109/CEC.2008.4631155
14. R. A. Formato, Central force optimization: a new metaheuristic with applications in applied electromagnetics, *Prog. Electromagn. Res.*, **77** (2007), 425–491. https://doi.org/10.2528/PIER07082403

15. B. Alatas, ACROA: artificial chemical reaction optimization algorithm for global optimization, *Expert Syst. Appl.*, **38** (2011), 13170–13180. https://doi.org/10.1016/j.eswa.2011.04.126
16. A. Hatamlou, Black hole: a new heuristic optimization approach for data clustering, *Inf. Sci.*, **222** (2013), 175–184. https://doi.org/10.1016/j.ins.2012.08.023
17. H. Du, X. Wu, J. Zhuang, Small-world optimization algorithm for function optimization, in *Advances in Natural Computation*, (2006), 264–273. https://doi.org/10.1007/11881223_33
18. H. Shah-Hosseini, Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimization, *Int. J. Comput. Sci. Eng.*, **6** (2011), 132–140. https://doi.org/10.1504/IJCSE.2011.041221
19. Y. T. Hsiao, C. L. Chuang, J. A. Jiang, C. C. Chien, A novel optimization algorithm: space gravitational optimization, in *2005 IEEE International Conference on Systems, Man and Cybernetics*, **3** (2005), 2323–2328. https://doi.org/10.1109/ICSMC.2005.1571495
20. F. A. Hashim, E. H. Houssein, M. S. Mabrouk, W. Al-Atabany, S. Mirjalili, Henry gas solubility optimization: a novel physics-based algorithm, *Future Gener. Comput. Syst.*, **101** (2019), 646–667. https://doi.org/10.1016/j.future.2019.07.015
21. L. Abualigah, A. Diabat, S. Mirjalili, M. A. Elaziz, A. H. Gandomi, The arithmetic optimization algorithm, *Comput. Methods Appl. Mech. Eng.*, **376 (**2021), 113609. https://doi.org/10.1016/j.cma.2020.113609
22. I. Ahmadianfar, A. A. Heidari, A. H. Gandomi, X. Chu, H. Chen, RUN beyond the metaphor: an efficient optimization algorithm based on Runge Kutta method, *Expert Syst. Appl.*, **181** (2021), 115079. https://doi.org/10.1016/j.eswa.2021.115079
23. J. Kennedy, R. Eberhart, Particle swarm optimization, in *Proceedings of ICNN'95-International Conference on Neural Networks*, **4** (1995), 1942–1948. https://doi.org/10.1109/ICNN.1995.488968
24. D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *J. Global Optim.*, **39** (2007), 459–471. https://doi.org/10.1007/s10898-007-9149-x
25. M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, *IEEE Trans. Syst., Man, Cybern., Part B*, **26** (1996), 29–41. https://doi.org/10.1109/3477.484436
26. X. S. Yang, Firefly algorithms for multimodal optimization, in *Stochastic Algorithms: Foundations and Applications*, Springer, (2009), 169–178. https://doi.org/10.1007/978-3-642-04944-6_14
27. X. S. Yang, A. H. Gandomi, Bat algorithm: a novel approach for global engineering optimization, *Eng. Comput.*, **29** (2012), 464–483. https://doi.org/10.1108/02644401211235834
28. E. Valian, E. Valian, A cuckoo search algorithm by Lévy flights for solving reliability redundancy allocation problems, *Eng. Optim.*, **45** (2013), 1273–1286. https://doi.org/10.1080/0305215X.2012.729055
29. S. A. Uymaz, G. Tezel, E. Yel, Artificial algae algorithm (AAA) for nonlinear global optimization, *Appl. Soft Comput.*, **31** (2015), 153–171. https://doi.org/10.1016/j.asoc.2015.03.003
30. M. S. Kiran, TSA: Tree-seed algorithm for continuous optimization, *Expert Syst. Appl.*, **42** (2015), 6686–6690. https://doi.org/10.1016/j.eswa.2015.04.055
31. S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Software*, **69** (2014), 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

32. J. James, V. O. Li, A social spider algorithm for global optimization, *Appl. Soft Comput.*, **30** (2015), 614–627. https://doi.org/10.1016/j.asoc.2015.02.014
33. S. Mirjalili, Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm, *Knowl.-Based Syst.*, **89** (2015), 228–249. https://doi.org/10.1016/j.knosys.2015.07.006
34. S. Mirjalili, A. Lewis, The whale optimization algorithm, *Adv. Eng. Software*, **95** (2016), 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008
35. A. Kaveh, N. Farhoudi, A new optimization method: dolphin echolocation, *Adv. Eng. Software*, **59** (2013), 53–70. https://doi.org/10.1016/j.advengsoft.2013.03.004
36. S. C. Chu, P. W. Tsai, J. S. Pan, Cat swarm optimization, in *PRICAI 2006: Trends in Artificial Intelligence*, Springer, (2006), 854–858. https://doi.org/10.1007/978-3-540-36668-3_94
37. M. Yazdani, F. Jolai, Lion optimization algorithm (LOA): a nature-inspired metaheuristic algorithm, *J. Comput. Des. Eng.*, **3** (2016), 24–36. https://doi.org/10.1016/j.jcde.2015.06.003
38. X. Bo, W. J. Gao, Fruit fly optimization algorithm, in *Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms*, 2014.
39. M. Khishe, M. R. Mosavi, Chimp optimization algorithm, *Expert Syst. Appl.*, **149** (2020), 113338. https://doi.org/10.1016/j.eswa.2020.113338
40. M. S. Braik, Chameleon swarm algorithm: a bio-inspired optimizer for solving engineering design problems, *Expert Syst. Appl.*, **174** (2021), 114685. https://doi.org/10.1016/j.eswa.2021.114685
41. S. Li, H. Chen, M. Wang, A. A. Heidari, S. Mirjalili, Slime mould algorithm: a new method for stochastic optimization, *Future Gener. Comput. Syst.*, **111** (2020), 300–323. https://doi.org/10.1016/j.future.2020.03.055
42. Y. Yang, H. Chen, A. A. Heidari, A. H. Gandomi, Hunger games search: visions, conception, implementation, deep analysis, perspectives, and towards performance shifts, *Expert Syst. Appl.*, **177** (2021), 114864. https://doi.org/10.1016/j.eswa.2021.114864
43. J. Tu, H. Chen, M. Wang, A. H. Gandomi, The colony predation algorithm, *J. Bionic Eng.*, **18** (2021), 674–710. https://doi.org/10.1007/s42235-021-0050-y
44. G. B. Dantzig, J. H. Ramser, The truck dispatching problem, *Manage. Sci.*, **6** (1959), 80–91. https://doi.org/10.1287/mnsc.6.1.80
45. J. Du, X. Li, L. Yu, R. Dan, J. Zhou, Multi-depot vehicle routing problem for hazardous materials transportation: a fuzzy bilevel programming, *Inf. Sci.*, **399** (2017), 201–218. https://doi.org/10.1016/j.ins.2017.02.011
46. A. García-Nájera, J. A. Bullinaria, M. A. Gutiérrez-Andrade, An evolutionary approach for multi-objective vehicle routing problems with backhauls, *Comput. Ind. Eng.*, **81** (2015), 90–108. https://doi.org/10.1016/j.cie.2014.12.029
47. E. Cao, M. Lai, H. Yang, Open vehicle routing problem with demand uncertainty and its robust strategies, *Expert Syst. Appl.*, **41** (2014), 3569–3575. https://doi.org/10.1016/j.eswa.2013.11.004
48. E. Jabir, V. V. Panicker, R. Sridharan, Design and development of a hybrid ant colony-variable neighbourhood search algorithm for a multi-depot green vehicle routing problem, *Trans. Res. Part D: Transp. Environ.*, **57** (2017), 422–457. https://doi.org/10.1016/j.trd.2017.09.003
49. M. Okulewicz, J. Mańdziuk, The impact of particular components of the PSO based algorithm solving the Dynamic Vehicle Routing Problem, *Appl. Soft Comput.*, **58** (2017), 586–604. https://doi.org/10.1016/j.asoc.2017.04.070

50. S. Iqbal, M. Kaykobad, M. S. Rahman, Solving the multi-objective Vehicle Routing Problem with Soft Time Windows with the help of bees, *Swarm Evol. Comput.*, **24** (2015), 50–64. https://doi.org/10.1016/j.swevo.2015.06.001
51. E. Teymourian, V. Kayvanfar, G. M. Komaki, M. Zandieh, Enhanced intelligent water drops and cuckoo search algorithms for solving the capacitated vehicle routing problem, *Inf. Sci.*, **334–335** (2016), 354–378. https://doi.org/10.1016/j.ins.2015.11.036
52. A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: algorithm and applications, *Future Gener. Comput. Syst.*, **97** (2019), 849–872. https://doi.org/10.1016/j.future.2019.02.028
53. C. Fan, Y. Zhou, Z. Tang, Neighborhood centroid opposite-based learning Harris Hawks optimization for training neural networks, *Evol. Intell.*, **14** (2021), 1847–1867. https://doi.org/10.1007/s12065-020-00465-x
54. H. Zhang, H. Nguyen, X. N. Bui, B. Pradhan, P. G. Asteris, R. Costache, et al., A generalized artificial intelligence model for estimating the friction angle of clays in evaluating slope stability using a deep neural network and Harris Hawks optimization algorithm, *Eng. Comput.*, 2021. https://doi.org/10.1007/s00366-020-01272-9
55. S. Mouassa, T. Bouktir, F. Jurado, Scheduling of smart home appliances for optimal energy management in smart grid using Harris-hawks optimization algorithm, *Optim. Eng.*, **22** (2021), 1625–1652. https://doi.org/10.1007/s11081-020-09572-1
56. P. Kumar, S. N. Singh, S. Dawra, Software component reusability prediction using extra tree classifier and enhanced Harris hawks optimization algorithm, *Int. J. Syst. Assur. Eng. Manage.*, **13** (2022), 892–903. https://doi.org/10.1007/s13198-021-01359-6
57. M. K. Naik, R. Panda, A. Wunnava, B. Jena, A. Abraham, A leader Harris hawks optimization for 2-D Masi entropy-based multilevel image thresholding, *Multimedia Tools Appl.*, **80** (2021), 35543–35583. https://doi.org/10.1007/s11042-020-10467-7
58. M. A. Mossa, O. M. Kamel, H. M. Sultan, A. A. Z. Diab, Parameter estimation of PEMFC model based on Harris Hawks' optimization and atom search optimization algorithms, *Neural Comput. Appl.*, **33** (2021), 5555–5570. https://doi.org/10.1007/s00521-020-05333-4
59. E. H. Houssein, M. E. Hosney, D. Oliva, W.M. Mohamed, M. Hassaballah, A novel hybrid Harris hawks optimization and support vector machines for drug design and discovery, *Comput. Chem. Eng.*, **133** (2020), 106656. https://doi.org/10.1016/j.compchemeng.2019.106656
60. I. N. Setiawan, R. Kurniawan, B. Yuniarto, R. E. Caraka, B. Pardamean, Parameter optimization of support vector regression using Harris Hawks optimization, *Procedia Comput. Sci.*, **179** (2021), 17–24. https://doi.org/10.1016/j.procs.2020.12.003
61. A. A. Dehkordi, A. S. Sadiq, S. Mirjalili, K. Z. Ghafoor, Nonlinear-based Chaotic Harris Hawks Optimizer: algorithm and internet of vehicles application, *Appl. Soft Comput.*, **109** (2021), 107574. https://doi.org/10.1016/j.asoc.2021.107574
62. H. M. Alabool, D. Alarabiat, L. Abualigah, A. A. Heidari, Harris Hawks optimization: a comprehensive review of recent variants and applications, *Neural Comput. Appl.*, **33** (2021), 8939–8980. https://doi.org/10.1007/s00521-021-05720-5
63. R. Y. Zhang, Z. M. Wang, D. C. Wang, Modeling and optimization of transportation problem for shared electric-cars with recharging scheduling, *Syst. Eng.-Theory Pract.*, **41** (2021), 370–377.
64. H. Haklı, H. Uğuz, A novel particle swarm optimization algorithm with Levy flight, *Appl. Soft Comput.*, **23** (2014), 333–345. https://doi.org/10.1016/j.asoc.2014.06.034

65. X. S. Yang, Nature-inspired Metaheuristic Algorithms, Luniver press, 2010.
66. N. Wang, W. J. Zhang, X. Liu, J. Zuo, Inter-Site-Vehicle artificial scheduling strategy design for electric vehicle sharing, *J. Tongji Univ. (Nat. Sci.)*, **46** (2018), 1064–1071. https://doi.org/10.11908/j.issn.0253-374x.2018.08.009
67. A. Beşkirli, İ. Dağ, A new binary variant with transfer functions of Harris Hawks optimization for binary wind turbine micrositing, *Energy Rep.*, **6** (2020), 668–673. https://doi.org/10.1016/j.egyr.2020.11.154
68. M. Beşkirli, İ. Koç, H. Haklı, H. Kodaz, A new optimization algorithm for solving wind turbine placement problem: binary artificial algae algorithm, *Renewable Energy*, **121** (2018), 301–308. https://doi.org/10.1016/j.renene.2017.12.087
69. R. M. Rizk-Allah, A. E. Hassanien, M. Elhoseny, M. Gunasekaran, A new binary salp swarm algorithm: development and application for optimization tasks, *Neural Comput. Appl.*, **31** (2019), 1641–1663. https://doi.org/10.1007/s00521-018-3613-z

AIMS Press