ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/377054563

Q-learning based vegetation evolution for numerical optimization and wireless sensor network coverage optimization

Article *in* Alexandria Engineering Journal · January 2024

DOI: 10.1016/j.aej.2023.12.028

CITATIONS
2

READS
54

4 authors, including:

Rui Zhong
Hokkaido University
17 PUBLICATIONS **32** CITATIONS
SEE PROFILE

Jun Yu
Niigata University
20 PUBLICATIONS **16** CITATIONS
SEE PROFILE

Masaharu Munetomo
Hokkaido University
143 PUBLICATIONS **1,289** CITATIONS
SEE PROFILE

All content following this page was uploaded by Rui Zhong on 02 January 2024.

The user has requested enhancement of the downloaded file.

Alexandria Engineering Journal 87 (2024) 148–163

Contents lists available at ScienceDirect

Alexandria Engineering Journal

journal homepage: www.elsevier.com/locate/aej

Original Article

Q-learning based vegetation evolution for numerical optimization and wireless sensor network coverage optimization

Rui Zhong [a,*], Fei Peng [b], Jun Yu [c], Masaharu Munetomo [d]

[a] Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan
[b] Graduate School of Science and Technology, Niigata University, Niigata, Japan
[c] Institute of Science and Technology, Niigata University, Niigata, Japan
[d] Information Initiative Center, Hokkaido University, Sapporo, Japan

ARTICLE INFO

Dataset link: https://github.com/RuiZhong961230/QVEGE

Keywords:
Meta-heuristic algorithm
Vegetation evolution
Q-learning
Wireless sensor network coverage problems

ABSTRACT

Vegetation evolution (VEGE) is a newly proposed meta-heuristic algorithm (MA) with excellent exploitation but relatively weak exploration capacity. We thus focus on further balancing the exploitation and the exploration of VEGE well to improve the overall optimization performance. This paper proposes an improved Q-learning based VEGE, and we design an exploitation archive and an exploration archive to provide a variety of search strategies, each archive contains four efficient and easy-implemented search strategies. In addition, online Q-Learning, as well as ε-greedy scheme, are employed as the decision-maker role to learn the knowledge from the past optimization process and determine the search strategy for each individual automatically and intelligently. In numerical experiments, we compare our proposed QVEGE with eight state-of-the-art MAs including the original VEGE on CEC2020 benchmark functions, twelve engineering optimization problems, and wireless sensor networks (WSN) coverage optimization problems. Experimental and statistical results confirm that the proposed QVEGE demonstrates significant enhancements and stands as a strong competitor among existing algorithms. The source code of QVEGE is publicly available at https://github.com/RuiZhong961230/QVEGE.

1. Introduction

As the EC community develops rapidly, hundreds and thousands of new MAs spring up like mushrooms. Thanks to their superior characteristics such as robustness, flexibility, efficiency, and easy implementation, MAs have achieved great success in feature selection [1–3], structural damage detection [4,5], neural architecture search [6,7], and other fields [8–10]. Here, we list some typical MAs and their inspiration in Table 1, and roughly classify them into five categories: bio-inspired, math-based, swarm-based, physics-based, and human-based.

In the meantime, the introduction of machine learning techniques to MAs contributes to the significant development: Hassan et al. [31] proposed a novel variant of the whale optimization algorithm called Q-learning based whale optimization algorithm with the Exponential Monte Carlo acceptance principle. The embedding of Q-learning allows the proposal to select the search operators based on historical knowledge automatically, and the Exponential Monte Carlo acceptance principle can further strengthen the capacity of escaping local optima. Hamad et al. [32] proposed a novel Q-learning embedded sine cosine algorithm. The proposed approach can intelligently control the parameter of SCA through the embedded Q-learning technique. Dong et al. [33] proposed an efficient surrogate-assisted grey wolf optimization for high-dimensional computationally expensive black-box optimization problems. The radial basis function is employed as the surrogate model to mine the knowledge from the fitness landscape, which significantly contributes to the optimization convergence acceleration. Zhong et al. [34] proposed a surrogate ensemble assisted DE cooperating with the cooperative coevolutionary framework to deal with high-dimensional and computationally expensive optimization problems. The global surrogate model constructed by the generalized regression neural network estimates globally high-quality solutions and the local surrogate model built by the Gaussian process regression model contributes to exploitative search. These successful applications practically confirm the potential of hybridizing the machine learning approach in metaheuristics.

VEGE [35] is a new member of the bio-inspired meta-heuristic algorithm, and repeatedly imitates the growth and seeding behaviors of real

* Corresponding author.
E-mail addresses: rui.zhong.u5@elms.hokudai.ac.jp (R. Zhong), f22c128f@mail.cc.niigata-u.ac.jp (F. Peng), yujun@ie.niigata-u.ac.jp (J. Yu), munetomo@iic.hokudai.ac.jp (M. Munetomo).

https://doi.org/10.1016/j.aej.2023.12.028
Received 4 September 2023; Received in revised form 25 November 2023; Accepted 15 December 2023
Available online 22 December 2023
1110-0168/© 2023 THE AUTHORS. Published by Elsevier BV on behalf of Faculty of Engineering, Alexandria University. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Nomenclature

EC	Evolutionary Computation
MA	Meta-heuristic Algorithm
VEGE	Vegetation Evolution
CEC	IEEE Congress on Evolutionary Computation
QVEGE	Q-learning based Vegetation Evolution
LHS	Latin Hypercube Sampling
WSN	Wireless Sensor Network
LB	Lower bound of search space
UB	Upper bound of search space
GR	Growth radius
GD	Growth direction
GC	Growth cycle
MS	Moving scale
FEs	Fitness evaluations
DE	Differential Evolution
PSO	Particle Swarm Optimization
SSA	Sparrow Search Algorithm
BES	Bald Eagle Search
AEO	Artificial Ecosystem-based Optimization
BRO	Battle Royale Optimization
HGS	Hunger Games Search
HBA	Honey Badger Algorithm
OOA	Osprey Optimization Algorithm

Table 1
Representative MAs in literature.

Category	Algorithms	Inspiration	Year
Bio-inspired	Grey Wolf Optimizer [11]	social behavior of grey wolves	2014
	Sailfish Optimizer [12]	hunting sailfish behaviors	2019
	Slime Mould Algorithm [13]	oscillation mode of slime mould	2020
	Dingo Optimizer [14]	collaborative and social behavior of dingoes	2021
Math-based	Cross-Entropy Method [15]	variance minimization principle	2005
	Sine Cosine Algorithm [16]	proprieties of sine and cosine functions	2016
	Gradient-Based Optimizer [17]	gradient-based Newton's method	2020
	Chaos Game Optimization [18]	some principles of chaos theory	2021
Swarm-based	Cat Swarm Optimization [19]	behaviors of cats	2006
	Moth-Flame Optimization [20]	navigation method of moths in nature	2015
	Salp Swarm Algorithm [21]	swarming behaviors of salps	2017
	Greylag Goose Optimization [22]	seasonal migrations of the greylag goose	2023
Physics-based	Multi-Verse Optimizer [23]	white hole, black hole, and wormhole	2016
	Nuclear Reaction Optimization [24]	nuclear reaction process	2019
	Atom Search Optimization [25]	basic molecular dynamics	2019
	Equilibrium Optimizer [26]	control volume mass balance models	2020
Human-based	Culture Algorithm [27]	the behavior of species in nature	1994
	Brain Storm Optimization [28]	the creativity of human being	2011
	Queuing Search Algorithm [29]	human activities in queuing process	2018
	Battle Royale Optimization [30]	social behavior of some animals	2021

plants to realize the exploration and exploitation for finding the global optimum. Experimental results on CEC2013 benchmark functions verified that the performance of VEGE is competitive with several classic MAs, such as DE, PSO, and fireworks algorithm. However, as a young member, many possible improvements of VEGE are mentioned in [35], and we also notice that the original VEGE has powerful exploitation ability but is weak in exploration. In complex optimization problems, VEGE may easily get trapped in local optima and become premature, which makes it difficult for VEGE to achieve an acceptable solution within reasonable CPU time. Although some published works [36–39] attempted to strengthen the diversity of the population during the optimization by introducing various mutation strategies and multiple search operators, the above-mentioned shortages are still present in VEGE.

The objective of this paper focuses on improving the overall performance of the original VEGE by introducing an intelligent methodology and proposing QVEGE. We notice that the exploitation and exploration strategies of VEGE are too simple and thus limit the diversity of the population, and we believe that the problem stems, in part, from insufficient exploration capacity. Therefore, we design an exploitation archive and an exploration archive to address the above-mentioned problem, each archive contains several easy-implemented but efficient search strategies. In addition, the online Q-learning, as well as the ε-greedy scheme, is employed to drive the proposed QVEGE to select the search strategy automatically. We also introduce the LHS method for population initialization to generate a more uniform population in the high-dimensional space. Specifically, the contributions of this paper can be summarized as follows:

(1). We notice that the shortage of the original VEGE is the exploration, which limits the performance in complex optimization problems, and we owe this disadvantage to the single-search strategy partially. Thus, an exploration archive and an exploitation archive are designed to enhance the diversity of search strategies in our proposed QVEGE.

(2). Online Q-learning cooperated with the ε-greedy scheme is employed to determine the search strategy for each individual during the optimization, which is hoped to learn the experience from the past optimization process and make the decision intelligently.

(3). We adopt the LHS method instead of random initialization to generate the initial population in QVEGE. Population initialization is one of the most crucial steps of MAs because it will directly affect the final solution's quality. In high-dimensional space, conventional simple random cannot ensure that each interval is placed with a sample, and the solutions may form clusters or gaps, which can be alleviated by LHS.

(4). We implement comprehensive numerical experiments with eight state-of-the-art competitor MAs on CEC2020 benchmark functions, twelve engineering optimization problems, and WSN coverage optimization problems. Experimental and statistical results show that the proposed QVEGE has broad prospects for solving complex optimization problems.

The remainder of this paper is organized as follows. We introduce the original VEGE in section 2. Section 3 provides a detailed introduction to our proposed QVEGE. Section 4 covers numerical experiments

to evaluate the performance of QVEGE. Section 5 discusses the performance of QVEGE and lists some open topics for future research. Finally, Section 6 concludes the paper.

2. Related works

2.1. VEGE

2.1.1. Initialization period

VEGE adopts the random initialization technique as the initial population generator. Random initialization is the simplest and the most common method since it is unnecessary to provide any information about the solution or fitness landscape. Each initial solution can be described by Eq. (1)

$$X = LB + rand(0,1) * (UB - LB) \tag{1}$$

where X is an n-dimensional vector representing a candidate solution, that is, an individual. LB and UB are the lower and upper bound of the search space, respectively, and $rand(0,1)$ is a random number generated from 0 to 1.

In the low-dimensional space, random initialization is an efficient method both in solution quality and time consumption. Still, in high-dimensional space, it cannot ensure complete interval representation or evenly spaced sample points, leading to clusters or gaps in the sample space, which in turn affects the distribution of solutions in the fitness landscape.

2.1.2. Exploitation: growth period

Although the growth mechanism of real plants is hard to describe by the mathematical model, VEGE simplifies this operator and applies two parameters to mimic the growth behavior of real plants: a constant growth radius GR and a random number growth direction GD. Eq. (2) can describe this search strategy.

$$X_{offspring} = X_{parent} + GR * GD \tag{2}$$

The perturbation generated by GR and GD is employed to describe the growth increment. The growth cycle GC controls the maximum iteration of the exploitation phase. Once the offspring has better fitness, it will replace the parent solution and survive to the next growth cycle, which is also consistent with the natural behaviors that the plants prefer to live in better environments. Note that each growth operation here will only generate one offspring individual.

2.1.3. Exploration: maturity period

Real plants in nature can produce many seeds rather than a few to ensure the number of seeds surviving in new environments. Inspired by this phenomenon, VEGE adopts a one-to-many principle to produce multiple seed solutions, and the difference among plants is ignored, which means each plant has an identical ability to generate the same number of seeds. Besides, VEGE adopts a cur/1-like strategy of DE to generate the seeds. The benefit of this strategy is that any differential vector between two individuals can form a share of the population distribution, and differential vectors contain a part of this information. Furthermore, a random parameter moving scale MS is introduced to scale dispersed distance. In natural environments, the movement of seeds will be affected by many factors, such as wind, water flow, animals, and so on. MS is employed to depict this uncertainty. Eq. (3) defines this search strategy.

$$X_{seed} = X_i + MS * (X_{r1} - X_{r2}) \tag{3}$$

X_i is the i^{th} solution, X_{r1}, X_{r2} are two randomly selected and mutually exclusive solution. Since VEGE adopts the one-to-many strategy to generate the offspring, the one-to-one selection scheme in the growth period cannot be employed in the maturity period. To guarantee the quality of survived solutions, VEGE mixes the plants and seeds, and only the top-n solutions can survive to the next generation.

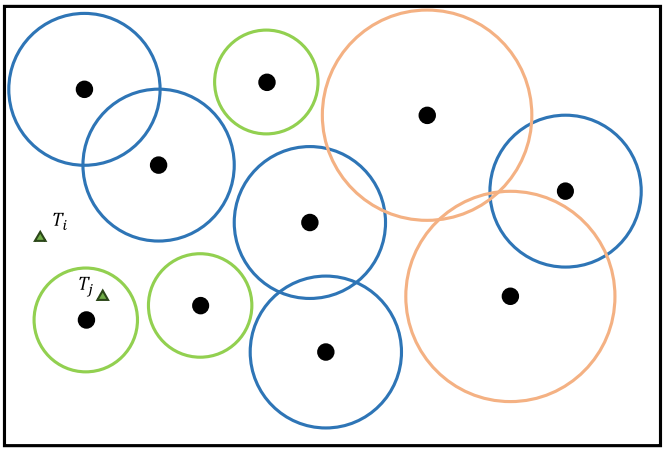

Fig. 1. A visual demonstration of the WSN coverage optimization problem.

2.2. WSN coverage optimization problem

The WSN coverage optimization problem [40] is a real-world challenge in the Internet of Things field, which requires the deployment of a large number of small, low-cost devices equipped with sensors. These sensors are distributed over an area to monitor and collect data from the environment. One of the fundamental goals in designing WSNs is to achieve efficient coverage of the target area to ensure that the entire region is adequately monitored by the sensors. A visual demonstration is shown in Fig. 1. The black-filled points represent the sensors and the spaces within the relatively large circles are covered. T_i and T_j are two example node positions where T_i is beyond any sensing range of sensor and T_j is within the signal range. In this paper, we focus on the Boolean model of wireless networks [41], such that, if the node position is within the signal range, the coverage state is 1 and vice versa. Simply, Eq. (4) can describe this model.

$$p_{cov}(s_i, T_k) = \begin{cases} 1, & if\ Dis(s_i, T_k) \le R_i \\ 0, & otherwise \end{cases} \tag{4}$$

where s_i is the i^{th} sensor, R_i is the coverage radius of i^{th} sensor, T_k is the k^{th} node position, and $Dis(s_i, T_k)$ denotes the Euclidean distance between s_i and T_k. The coverage rate of nodes is a critical metric that we are concerned about during the optimization, and the joint sensing probability of all sensor nodes for a given target node T_k is shown in Eq. (5).

$$P_{cov}(S, T_k) = 1 - \prod_{s_i \in S} (1 - p_{cov}(s_i, T_k)) \tag{5}$$

where S denotes the set of all sensors in the monitoring area, and the coverage rate can be computed by Eq. (6)

$$R_{cov}(S, T) = \frac{\sum P_{cov}(S, T_k)}{HW} \tag{6}$$

where HW represents the rectangular area of the monitoring area, and $R_{cov}(S, T)$ is the sum of target point probabilities for all sensor nodes covered [42]. However, the calculation of the joint area of the sensing area is difficult and computationally complex, and for the sake of simplicity, the Monte Carlo method [43,44] is employed in this research to estimate the coverage rate.

3. Our proposal: QVEGE

This section introduces our proposal in detail. We first provide the overview of QVEGE, and each component is introduced subsequently.

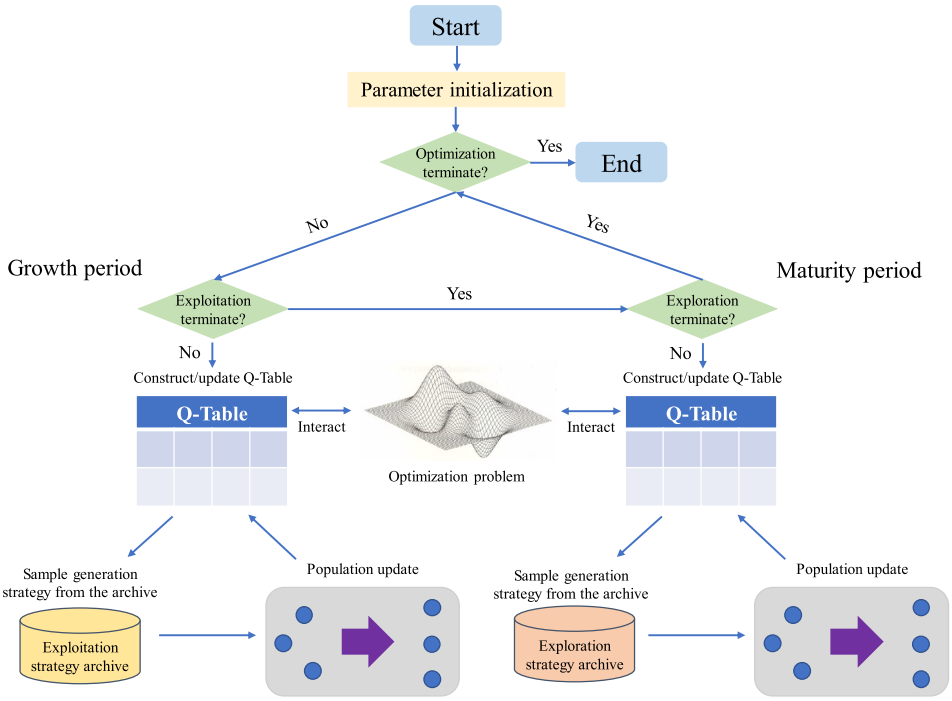

Fig. 2. The flowchart of QVEGE.

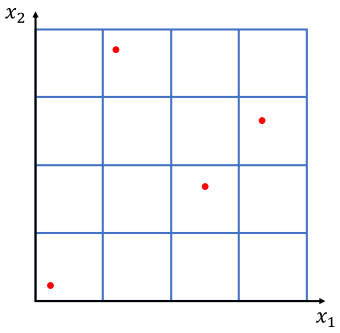

Fig. 3. A demonstration of LHS in 2-D space, where the sampling size is four.

3.1. The overview of QVEGE

Fig. 2 visualizes the main skeleton of our proposed QVEGE. After the population initialization by LHS, QVEGE repeats the growth period and maturity period until the optimum is found or FEs are exhausted. Meanwhile, we design multiple exploitation and exploration strategies in the growth and maturity period to construct the exploitation and the exploration archive, respectively. Online Q-learning learns the experience from past optimization and updates the Q-table. ε-greedy scheme cooperates with Q-table to select the search strategies to balance certainty and uncertainty, which is expected to decide the optimization sequence intelligently. In the following subsections, we will introduce five components: LHS for initialization, Exploitation archive design, Exploration archive design, Q-learning determination, and constraint handling method.

3.2. LHS for initialization

LHS [45] is a statistical method for generating approximate-random samples from a multi-dimensional distribution, which is a popular technique to generate the initial population instead of the random initialization in MAs [46,47]. A demonstration of LHS in 2-D search space can be visualized in Fig. 3.

LHS first divides the search space into equal intervals, and then the interval for every dimension is paired randomly to form the sampling areas. Each sample is randomly placed within the corresponding area. This technique can be extended to high-dimensional space easily. Compared to random initialization, LHS ensures that each interval is represented in the sample so that the sample points are more uniformly distributed at the cost of the computational budget.

3.3. Exploitation archive design

The essence of exploitation behavior is to generate a perturbation to each dimension of solutions, and the determination of this perturbation is a challenge. As we mentioned before the original VEGE adopts the most common local search strategy as the exploitation operator, here we introduce an exploitation archive to further enhance the diverse characteristics of the population. In the designed exploitation archive, four local search strategies are employed: uniform search, normal search, Lévy flight search, and chaotic search.

Uniform search: Uniform search is the same as the local search in the original VEGE, which can be described by Eq. (7)

$$X_{offspring} = X_{parent} + \delta \tag{7}$$

Each dimension of the solution X_i adds the perturbation δ, which is randomly sampled from the uniform distribution. We use the parameter setting of the original VEGE thus the $\delta \sim GR \cdot U(-1,1)$, where $GR = 2$ corresponds to the definition of growth radius in the original VEGE.

Normal search: Similar to the uniform search, the formulation of the normal search can also be expressed by Eq. (7), and only the difference is the distribution of δ follows the normal distribution which is $GR \cdot N(0,1)$.

Lévy flight search: Lévy flight is a random walk with heavy-tailed distribution [48] which has been widely combined with many metaheuristic algorithms to improve their performance [49–51]. Here, we adopt the Lévy flight as a basic exploitation operator, the perturbation δ of the Lévy flight can be formulated by Eq. (8).

$$\delta \sim \frac{u}{|v|^{\frac{1}{\beta}}}$$
$$\text{where } u \sim N(0,\sigma^2), v \sim N(0,1) \tag{8}$$
$$\sigma = \{\frac{\Gamma(1+\beta)\sin(\frac{\pi\beta}{2})}{\beta\Gamma(\frac{1+\beta}{2})2^{\frac{\beta-1}{2}}}\}^{\frac{1}{\beta}}$$

Chaotic search: Chaos is a kind of deterministic random method that is originally derived from non-linear dynamic systems [52]. The first application of chaotic sequence replacing random sequence to construct the initial population is proposed in [53]. Here, we adopt the Chebyshev map, one of the chaotic sequence construction techniques, as the basic perturbation generator. Eq. (9) shows the formulation of the Chebyshev map.

$$chaos_{i+1} = \cos(\frac{n}{cos(chaos_i)}) \tag{9}$$

Once the initial sample x_0 is determined, the chaotic sequence constructed by the Chebyshev map can be determined. And the perturbation can be described by Eq. (10).

$$\delta = GR \cdot \text{Chebyshev}(x_0) \tag{10}$$

where x_0 is the beginning of the chaotic sequence randomly sampled from 0 to 1. The benefit of the Chebyshev map is that the original formulation ranges from [-1, 1], and it is unnecessary to adjust the scale anymore.

3.4. Exploration archive design

We notice that the exploration strategy in the original VEGE simulates the cur/1 mutation in DE. To further strengthen the diversity of the exploration operator, four common mutation strategies close to cur/1 are employed in our proposed exploration archive: cur/1, cur to rand/1, cur to best/1, and cur to pbest/1.

cur/1: The cur/1 search strategy is applied in the original VEGE to generate the seeds. Similarly, we adopt this as one of the strategies in our exploration archive.

cur to rand/1: The adapted cur to rand/1 mutation strategy in QVEGE can be expressed by Eq. (11).

$$X_{seed} = X_i + MS_1 * (X_{r1} - X_i) + MS_2 * (X_{r2} - X_{r3}) \tag{11}$$

cur to best/1: The cur to best/1 replaces the X_{r1} with X_{best}.

$$X_{seed} = X_i + MS_1 * (X_{best} - X_i) + MS_2 * (X_{r2} - X_{r3}) \tag{12}$$

The differential vector $X_{best} - X_i$ greedily move from X_i to X_{best}, and $X_{r2} - X_{r3}$ adds the uncertainty to the X_{seed}.

cur to pbest/1: A more general operator is cur to pbest/1, which is proposed in [54] and use the mean of best-p solutions to construct the X_{rp}.

$$X_{seed} = X_i + MS_1 * (X_{rp} - X_i) + MS_2 * (X_{r2} - X_{r3}) \tag{13}$$

These four strategies have three mutually different expectations, which can strengthen the diversity of the population during optimization.

3.5. Q-learning determination

Q-learning [55] is a values-based reinforcement learning methodology, which contains four basic elements: environment (E), state (s), action (a), and the reward r depends on the reward function $R(s,a)$. For a specific task, we can observe a complete state-action-rewards trajectory $T = (s_1, a_1, r_1, \ldots)$, and this process can be described by Markov decision process [56]. Simply, the updated principle of Q-value can be formulated by Eq. (14).

$$Q^{new}(s_t,a_t) = Q(s_t,a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)) \tag{14}$$

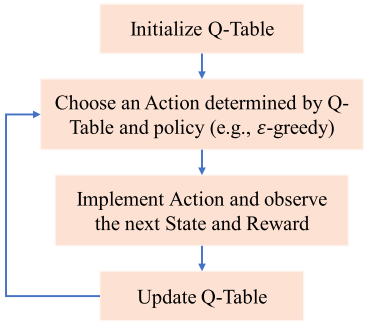

Fig. 4. The flowchart of Q-learning.

where α is the learning rate to control the increment. r_t is the reward for state s_t after taking the action a_t. Discount factor γ determines the importance of future rewards, and $\max_a Q(s_{t+1},a)$ is the maximum reward that can be achieved from state s_{t+1}. Fig. 4 demonstrates the flowchart of Q-learning.

In QVEGE, we adopt the online Q-learning method to guide the configuration of the optimization sequence, and the advantage of the online method is that the learning process and optimization of the instance take place simultaneously, which can save the computational budget. As for the correspondence between VEGE and Q-learning, we relate the involved concepts: the environment in Q-learning corresponds to the fitness landscape of problems, agents are the individuals or solutions in QVEGE, the state of the agent indicates the fitness is improved or degenerated through the search, actions are the search strategies that we designed in the archive, and the reward based on an individual can be calculated by the following Eq. (15).

$$R_t = F_{t-1}(S_{t-1},A_{t-1}) - F_t(S_t,A_t) \tag{15}$$

where F_t and F_{t-1} are fitness at the generation t and $t-1$ respectively. Since all benchmark functions in our experiments are minimization problems, once the better solution is obtained, the reward would be positive and vice versa. And for a search strategy A, the total reward can be expressed by Eq. (16)

$$\bar{R}_t^A = \frac{\sum_{i=1}^{n_A} R_t^A}{n_A} \tag{16}$$

n_A is the time of the strategy A to be adopted, and $\bar{R}_t^A$ represents the averaging improvement or reward by the search strategy A.

In the action selection, we adopt a common ε-greedy scheme. Simply, if a random value is smaller than the threshold ε, QVEGE will choose the search strategy with the highest previous reward, otherwise, QVEGE will choose a random operator. In summary, the pseudocode of complete QVEGE is shown in Algorithm 1.

3.6. Constraint handling for engineering problems

Since the original VEGE and QVEGE cannot deal with optimization problems with constraints, it is necessary to introduce a constraint-handling technique to QVEGE to solve engineering optimization problems. A common methodology is to attach the penalty to infeasible solutions, which is named the penalty function. Coello et al. [57] reviewed various penalty functions including static, dynamic, simulated annealing, adaptive, and death penalty. As one of the simplest methods, the static penalty function assigns a violation-related penalty to the individual, which can be formulated in Eq. (17).

$$F(R_i) = f(R_i) + w \cdot \sum_{i=1}^{m}(\max(0, g_i(R_i))) \tag{17}$$

Algorithm 1: QVEGE.

Input: Population size: N, Dimension: D, Maximum iteration: T_{max}, Threshold: ε, Exploitation archive: E_1; Exploration archive: E_2
Output: Optimum
Function QVEGE (N, D, T_{max}):
  $P \leftarrow$ **LHS**(D, S) # Population initialization with LHS
  $X_{best} \leftarrow$ **best**(P)
  Initialize Q-table Q_1, Q_2 for growth and maturity period.
  $t = 1$
  while $t \leq T_{max}$ **do**
    while *Growth period is not terminated* **do**
      Select search strategy from E_1 by ε-greedy scheme
      Implement the search and calculate the reward
    end
    Update Q_1 by Eq. (14)
    while *Maturity period is not terminated* **do**
      Select search strategy from E_2 by ε-greedy scheme
      Implement the search and calculate the reward
    end
    Update Q_2 by Eq. (14)
    $t \leftarrow t + 1$
  end
  return the optimum found by QVEGE

$F(\cdot)$ is the fitness function, while $f(\cdot)$ and $g_i(\cdot)$ are the objective function and constraint function, respectively. w is a constant set to $10e^7$ by default.

4. Numerical experiments

The details of the experiment are covered in this section. Section 4.1 introduces the experiment settings: experimental environments, benchmark functions, and compared methods with their parameters. Section 4.2 shows the experimental results.

4.1. Experiment settings

4.1.1. Experimental environments and implementation

All optimization methods are programmed by Python 3.11 and implemented in Lenovo Legion R9000P, which is equipped with Windows 11, AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz, and 16GB RAM. Except for VEGE and QVEGE, all algorithms are provided by the MEALPY library [58], CEC2020 benchmark functions are provided by OpFuNu [59], and engineering optimization problems can be downloaded from ENOPPY library [60].

4.1.2. Benchmark functions

To evaluate our proposed QVEGE comprehensively and fairly, we implement optimization experiments on three types of optimization problems:

(1). 10-D, 30-D, 50-D, and 100-D CEC2020 benchmark functions [61] in Table 2.

(2). Twelve real-world engineering optimization problems [62] in Table 3.

(3). WSN coverage optimization problem in Table 4.

4.1.3. Compared methods and parameters

We compare QVEGE with eight advanced MAs including SSA [63], BES [64], AEO [65], BRO [30], HGS [66], HBA [67], the original VEGE [35], and OOA [68]. Each algorithm for each optimization task is independently implemented 30 times to avoid randomness.

For the CEC2020 and engineering optimization problems, the population size of VEGE and QVEGE is 10, while the rest of the competitor algorithms are set to 100. Maximum FEs for CEC functions and engineering problems are 1,000*D (D = dimension) and 20,000, respectively, and the detailed parameter settings of compared methods are listed in Table 5. The experimental setting follows the suggested parameter from the original papers. For the WSN coverage optimization problem, the population size of VEGE and QVEGE is 10, while the rest of the algorithms are set to 30. Maximum FEs are set to 3,000, which is consistent with the recommended experimental settings in [42].

Table 2
Summary of the CEC2020 benchmark functions: Uni. = Unimodal function, Multi. = Multimodal function, Hybrid. = Hybrid function, Comp. = Composition function.

Fun.	Description	Feature	Optimum
f_1	Shifted and Rotated Bent Cigar Function	Uni.	100
f_2	Shifted and Rotated Schwefel's function	Multi.	1100
f_3	Shifted and Rotated Lunacek bi-Rastrigin function		700
f_4	Expanded Rosenbrock's plus Griewangk's function		1900
f_5	Hybrid function 1 (N = 3)	Hybrid.	1700
f_6	Hybrid function 2 (N = 4)		1600
f_7	Hybrid function 3 (N = 5)		2100
f_8	Composition function 1 (N = 3)	Comp.	2200
f_9	Composition function 2 (N = 4)		2400
f_{10}	Composition function 3 (N = 5)		2500
search space: [-100, 100]			

Table 3
Summary of 12 engineering optimization problems.

Name	Abbr.	Dim.	# of constraints
Welded Beam Problem	WBP	4	7
Pressure Vessel Problem	PVP	4	4
Compression Spring Problem	CSP	4	4
Speed Reducer Problem	SRD	7	11
Three Bar Truss Problem	TBTD	2	3
Gear Train Problem	GTD	4	0
Cantilever Beam Problem	CBD	5	1
I Beam Problem	IBD	4	2
Tubular Column Problem	TCD	2	6
Piston Lever Problem	PLD	4	4
Corrugated Bulkhead Problem	CBHD	4	6
Reinforced Concrete Beam Problem	RCB	3	2

Table 4
Experimental settings of WSN coverage optimization problem.

Parameter	Value
Search range lb and ub	0 and 50
Sensing radius	5
# of sensors Num	32, 42 and 54
Monte Carlo sampling size	2500

4.2. Experimental results

This section shows the experimental and statistical results of optimization. All techniques listed in Table 5 are independently implemented 30 times for every single optimization task. Holm multiple comparison test [69] is employed to determine the statistical significance between every pair of algorithms. +, ≈, and − are applied to represent that our proposed QVEGE is significantly better, with no significance, and significantly worse with the compared method, and the best value is in bold.

4.2.1. Case study 1: optimization on CEC2020 benchmark functions

This section provides the experimental and statistical results on CEC2020 benchmark functions, Table 6, 7, 8, and 9 summarize the results, and the convergence curve of representative functions (i.e., f_1: unimodal; f_2: multimodal; f_5: hybrid; f_9: composite.) are visualized in Fig. 5.

Table 5
The parameters of competitor algorithms.

MAs	Parameters	Value
SSA (2020)	Safety threshold α	0.8
	# of producers	0.2
	# of sparrows perceiving the danger	0.1
BES (2020)	Change controller a	2
	Acceleration coefficients c_1 and c_2	2
	Corner determination a	10
	Search cycle R	1.5
AEO (2020)	Parameter-free	
BRO (2021)	Dead threshold	3
HGS (2021)	Position update probability L	0.08
	Largest hunger	10,000
HBA (2022)	The ability of a honey badger to get food β	6
	Constant C	2
VEGE (2022)	Growth cycle GC	6
	Growth radius GR	2
	Total seed size	60
	Moving scaling MS	a random number in [-2,2]
OOA (2023)	Parameter-free	
QVEGE	Threshold ε	0.5
	Discount factor γ	0.9

Table 6
Experimental and statistical results on 10-D CEC2020 benchmark functions. f_1: Unimodal function; $f_2 - f_4$: Multimodal functions; $f_5 - f_7$: Hybrid functions; $f_8 - f_{10}$: Composition functions; mean and std: the mean and the standard deviation of 30 trial runs.

Func.		SSA	BES	AEO	BRO	HGS	HBA	VEGE	OOA	QVEGE
f_1	mean	1.06e+07 +	1.16e+09 +	1.19e+08 +	2.19e+09 +	2.38e+08 +	1.48e+06 +	1.47e+06 +	2.04e+08 +	**3.90e+05**
	std	1.61e+07	9.13e+08	1.41e+08	8.83e+08	2.34e+08	9.37e+05	7.35e+05	2.44e+08	2.35e+05
f_2	mean	6.15e+07 ≈	7.81e+10 +	9.04e+09 +	1.84e+11 +	1.75e+10 +	5.74e+08 +	1.28e+08 +	1.05e+10 +	**3.84e+07**
	std	1.02e+08	5.50e+10	8.35e+09	7.95e+10	2.61e+10	2.71e+09	6.41e+07	9.18e+09	2.18e+07
f_3	mean	1.37e+08 +	3.37e+10 +	3.59e+09 +	6.12e+10 +	5.43e+09 +	3.13e+07 ≈	4.55e+07 +	5.12e+09 +	**1.58e+07**
	std	4.39e+08	2.92e+10	4.42e+09	3.03e+10	9.06e+09	2.00e+07	2.28e+07	6.20e+09	8.37e+06
f_4	mean	1.98e+03 +	1.92e+03 +	1.91e+03 +	2.10e+03 +	2.11e+03 +	1.91e+03 +	1.91e+03 +	1.91e+03 +	**1.90e+03**
	std	1.00e+02	3.18e+01	6.97e+00	5.46e+02	4.39e+02	3.69e+00	2.18e+00	1.53e+01	1.77e+00
f_5	mean	1.91e+04 +	4.88e+04 +	1.68e+04 +	8.20e+05 +	7.44e+04 +	1.43e+04 ≈	5.57e+04 +	2.12e+04 +	**9.52e+03**
	std	1.21e+04	9.93e+04	7.92e+03	5.53e+05	2.49e+05	7.97e+03	5.94e+04	1.27e+04	5.97e+03
f_6	mean	1.36e+04 +	3.31e+03 +	2.38e+03 +	1.35e+04 +	1.04e+04 +	2.46e+03 +	5.39e+03 +	2.11e+03 +	**1.91e+03**
	std	8.94e+03	1.83e+03	5.16e+02	8.72e+03	7.17e+03	1.35e+03	4.62e+03	3.06e+02	3.38e+02
f_7	mean	2.79e+04 +	1.73e+04 +	1.28e+04 +	4.84e+05 +	3.60e+04 +	1.72e+04 +	1.89e+04 +	1.39e+04 +	**6.91e+03**
	std	1.79e+04	1.23e+04	7.75e+03	4.45e+05	2.87e+04	9.78e+03	1.24e+04	9.18e+03	4.33e+03
f_8	mean	2.33e+03 +	2.31e+03 +	2.31e+03 +	2.32e+03 +	2.32e+03 +	2.31e+03 +	2.32e+03 +	2.31e+03 +	**2.30e+03**
	std	1.11e+01	3.33e+00	3.39e+00	4.45e+00	6.90e+00	2.81e+00	1.64e+01	3.38e+00	1.90e+00
f_9	mean	2.72e+03 ≈	4.05e+03 +	3.04e+03 +	4.52e+03 +	3.24e+03 +	2.65e+03 ≈	2.66e+03 +	3.11e+03 +	**2.62e+03**
	std	2.80e+02	7.49e+02	1.99e+02	5.01e+02	3.15e+02	9.46e+01	7.49e+01	3.28e+02	5.56e+01
f_{10}	mean	3.16e+03 +	3.13e+03 +	3.07e+03 +	3.18e+03 +	3.13e+03 +	3.00e+03 ≈	3.01e+03 +	3.06e+03 +	**2.99e+03**
	std	1.11e+02	5.78e+01	5.37e+01	6.47e+01	6.59e+01	2.91e+01	3.77e+01	5.33e+01	2.92e+01
+/≈/−:		8/2/0	10/0/0	10/0/0	10/0/0	10/0/0	6/4/0	10/0/0	10/0/0	

4.2.2. Case study 2: optimization on engineering problems

This section provides the experimental and statistical results of competitor algorithms on twelve engineering optimization problems. As real-world applications, we also focus on the stability of algorithms. Therefore, Tables 10 and 11 provide the mean, the std, the best, and the worst value of 30 trial runs. Fig. 6 visualizes the convergence curves in engineering problems.

4.2.3. Case study 3: optimization on WSN coverage problems

Table 12 summarizes the mean and std of nine MAs on WSN coverage problems, Fig. 7 shows the corresponding convergence curves. In addition, we visualize the optimal solution for each MA among 30 trial runs in Figs. 8 and 9.

5. Discussion

This section first provides the computational complexity analysis theoretically and the performance analysis of QVEGE based on the experimental and statistical results. To further improve the performance of QVEGE and promote the development of the EC community, we list some open topics at the end of this section.

5.1. Computational complexity analysis

QVEGE mainly consists of three phases: initialization, growth period, and maturity period. Q-learning and corresponding operations are embedded in the growth period and the maturity period. Assuming the dimension of the problem is D, the population size is N, the growth cycle is GC, and the maximum iteration is T. The computational com-

Table 7
Experimental and statistical results on 30-D CEC2020 benchmark functions.

Func.		SSA	BES	AEO	BRO	HGS	HBA	VEGE	OOA	QVEGE
f_1	mean	1.00e+08 +	3.50e+10 +	7.20e+09 +	3.86e+10 +	1.10e+10 +	4.42e+09 +	3.26e+07 +	2.45e+10 +	**1.04e+07**
	std	9.09e+07	6.75e+09	2.16e+09	6.43e+09	4.44e+09	2.94e+09	7.89e+06	6.46e+09	3.42e+06
f_2	mean	1.55e+10 +	4.03e+12 +	8.91e+11 +	4.55e+12 +	1.04e+12 +	3.19e+11 +	3.48e+09 +	2.74e+12 +	**9.99e+08**
	std	9.37e+09	9.16e+11	3.63e+11	8.00e+11	3.35e+11	2.79e+11	7.10e+08	6.88e+11	3.49e+08
f_3	mean	3.63e+09 +	1.21e+12 +	2.15e+11 +	1.42e+12 +	3.41e+11 +	1.27e+11 +	1.10e+09 +	8.55e+11 +	**3.13e+08**
	std	2.24e+09	2.56e+11	6.58e+10	2.62e+11	1.15e+11	8.21e+10	2.03e+08	2.14e+11	7.98e+07
f_4	mean	2.75e+03 +	2.48e+05 +	8.85e+03 +	1.29e+05 +	1.07e+04 +	2.48e+03 +	1.93e+03 ≈	9.91e+04 +	**1.93e+03**
	std	5.39e+02	2.54e+05	1.00e+04	8.13e+04	9.95e+03	6.71e+02	5.88e+00	9.20e+04	7.00e+00
f_5	mean	1.31e+06 ≈	1.66e+07 +	5.06e+06 +	3.26e+07 +	4.24e+06 +	6.06e+05 ≈	8.33e+05 +	8.48e+06 +	**5.37e+05**
	std	9.74e+05	1.29e+07	5.57e+06	1.84e+07	6.91e+06	4.05e+05	5.51e+05	8.94e+06	2.97e+05
f_6	mean	3.44e+04 ≈	1.71e+07 +	9.62e+04 +	4.23e+06 +	8.79e+04 +	**1.64e+04** ≈	4.13e+04 +	1.63e+05 +	2.01e+04
	std	2.63e+04	3.55e+07	1.16e+05	5.68e+06	1.25e+05	1.33e+04	1.92e+04	3.71e+05	1.18e+04
f_7	mean	2.47e+06 +	5.28e+07 +	1.36e+07 +	9.19e+07 +	9.06e+06 +	7.92e+05 +	1.09e+06 +	2.86e+07 +	**5.22e+05**
	std	2.23e+06	4.56e+07	1.17e+07	4.36e+07	1.08e+07	4.52e+05	5.58e+05	2.98e+07	3.20e+05
f_8	mean	4.08e+03 +	3.13e+03 +	2.91e+03 +	3.04e+03 +	3.45e+03 +	2.55e+03 ≈	3.69e+03 +	3.00e+03 +	**2.54e+03**
	std	9.28e+02	4.92e+02	2.30e+02	2.04e+02	4.99e+02	9.02e+01	7.28e+02	2.50e+02	6.65e+01
f_9	mean	7.08e+03 +	2.66e+04 +	8.46e+03 +	2.26e+04 +	1.38e+04 +	7.20e+03 +	3.01e+03 +	2.05e+04 +	**2.84e+03**
	std	6.11e+03	4.83e+03	9.54e+02	3.76e+03	4.49e+03	2.41e+03	4.65e+01	3.75e+03	9.97e+01
f_{10}	mean	3.08e+03 +	5.30e+03 +	3.53e+03 +	4.83e+03 +	3.46e+03 +	3.07e+03 +	2.96e+03 ≈	4.44e+03 +	**2.95e+03**
	std	7.36e+01	5.82e+02	1.73e+02	5.74e+02	1.91e+02	8.66e+01	3.13e+01	4.77e+02	3.06e+01
+/≈/−:		8/2/0	10/0/0	10/0/0	10/0/0	10/0/0	7/3/0	8/2/0	10/0/0	

Table 8
Experimental and statistical results on 50-D CEC2020 benchmark functions.

Func.		SSA	BES	AEO	BRO	HGS	HBA	VEGE	OOA	QVEGE
f_1	mean	1.75e+08 +	9.00e+10 +	2.24e+10 +	1.03e+11 +	2.87e+10 +	1.78e+10 +	1.01e+08 +	7.19e+10 +	**2.72e+07**
	std	6.47e+07	1.21e+10	4.87e+09	1.45e+10	5.36e+09	7.13e+09	1.46e+07	9.46e+09	7.55e+06
f_2	mean	3.02e+10 +	9.67e+12 +	2.48e+12 +	1.12e+13 +	3.07e+12 +	1.57e+12 +	1.15e+10 +	8.13e+12 +	**3.64e+09**
	std	1.78e+10	1.73e+12	5.73e+11	1.63e+12	7.12e+11	6.10e+11	1.54e+09	9.81e+11	1.41e+09
f_3	mean	9.66e+09 +	3.10e+12 +	8.26e+11 +	3.87e+12 +	9.30e+11 +	4.95e+11 +	3.77e+09 +	2.69e+12 +	**1.08e+09**
	std	4.79e+09	6.26e+11	1.99e+11	4.91e+11	2.34e+11	2.13e+11	7.08e+08	4.35e+11	3.19e+08
f_4	mean	4.21e+03 +	1.28e+06 +	5.59e+04 +	1.39e+06 +	4.77e+04 +	6.31e+03 +	**1.96e+03** ≈	6.23e+05 +	1.96e+03
	std	9.42e+02	5.88e+05	5.80e+04	5.75e+05	3.78e+04	6.95e+03	9.46e+00	3.66e+05	1.48e+01
f_5	mean	1.39e+07 +	1.08e+08 +	2.18e+07 +	8.57e+07 +	2.21e+07 +	2.98e+06 +	3.31e+06 +	4.46e+07 +	**1.39e+06**
	std	9.48e+06	6.15e+07	1.13e+07	4.00e+07	1.56e+07	1.30e+06	1.03e+06	4.08e+07	5.41e+05
f_6	mean	9.70e+04 +	1.59e+09 +	7.34e+06 +	3.59e+08 +	2.43e+06 +	1.24e+05 +	5.96e+04 +	1.52e+08 +	**3.23e+04**
	std	1.37e+05	2.20e+09	8.34e+06	3.28e+08	3.61e+06	2.30e+05	2.54e+04	1.74e+08	7.44e+03
f_7	mean	2.32e+07 +	2.27e+09 +	1.46e+08 +	1.66e+09 +	9.23e+07 +	9.17e+06 +	5.21e+06 +	7.86e+08 +	**2.20e+06**
	std	2.39e+07	1.58e+09	1.10e+08	8.17e+08	8.71e+07	5.65e+06	2.03e+06	5.07e+08	1.13e+06
f_8	mean	1.16e+04 +	8.81e+03 +	6.30e+03 +	6.32e+03 +	9.26e+03 +	3.93e+03 ≈	8.03e+03 +	7.00e+03 +	**3.61e+03**
	std	2.79e+03	2.49e+03	2.33e+03	1.09e+03	2.10e+03	8.94e+02	1.98e+03	1.69e+03	6.46e+02
f_9	mean	2.26e+04 +	6.30e+04 +	1.85e+04 +	6.08e+04 +	4.06e+04 +	2.29e+04 +	3.43e+03 +	5.45e+04 +	**3.20e+03**
	std	1.36e+04	4.95e+03	5.36e+03	8.36e+03	9.77e+03	7.12e+03	1.10e+02	3.83e+03	2.29e+02
f_{10}	mean	4.07e+03 +	1.76e+04 +	7.03e+03 +	1.73e+04 +	6.83e+03 +	4.75e+03 +	3.59e+03 +	1.37e+04 +	**3.48e+03**
	std	2.40e+02	3.37e+03	7.70e+02	4.13e+03	1.32e+03	5.60e+02	1.24e+02	2.48e+03	1.50e+02
+/≈/−:		10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	9/1/0	9/1/0	10/0/0	

plexity of initialization is $O(N \times D)$. In the growth period, QVEGE repeats the exploitation operators within the growth cycle. Considering the number of search strategies in the archive is a constant. Thus the selection of search strategy is $O(1)$, and the computational complexity of the growth period is $O(GC \times N \times D)$. Simply, the computational complexity of the Q-table update which is executed after the growth period is also $O(1)$. In the maturity period, although each plant can generate multiple seeds, we can still denote the computational complexity of these exploration operators by $O(N \times D)$. Another process is to select the best-N individual to survive. We sort the individuals with the corresponding fitness and execute the selection, where the computational complexity is $O(N \log N + N) := O(N \log N)$. In summary, the total computational complexity of QVEGE can be expressed by Eq. (18).

$$\begin{aligned} &O(N \times D + T \times (GC \times N \times D + N \times D + N \log N)) \\ =&O(N \times D + T \times N \times (GC \times D + D + \log N)) \end{aligned} \tag{18}$$

Although the computational complexity of QVEGE is identical to the conventional VEGE theoretically [35,39], QVEGE consumes higher CPU times in practice since the Q-Table update and ε-greedy scheme follow the computational complexity of $O(1)$.

Additionally, the principle of computational complexity analysis between QVEGE and other MAs is distinct, we must standardize this principle to compare QVEGE with other MAs such as DE, GA, and PSO fairly. Here, given the GC in Eq. (18) is a constant (i.e., $GC = 6$ in our numerical experiments), we further simplify the computational complexity of QVEGE to Eq. (19).

$$\begin{aligned} &O(N \times D + T \times N \times (D + \log N)) \\ =&O(T \times N \times (D + \log N)) \end{aligned} \tag{19}$$

The previous term of $O(N \times D)$ presents the computational complexity of the population initialization, and the iteration part dominates the computational complexity of QVEGE, thus, Eq. (19) formulates the simplified computational complexity analysis of QVEGE. In the meantime,

Table 9
Experimental and statistical results on 100-D CEC2020 benchmark functions.

Func.		SSA	BES	AEO	BRO	HGS	HBA	VEGE	OOA	QVEGE
f_1	mean	1.03e+09 +	2.47e+11 +	6.96e+10 +	3.07e+11 +	8.70e+10 +	6.78e+10 +	4.19e+08 +	2.14e+11 +	**1.22e+08**
	std	3.31e+08	1.94e+10	1.35e+10	2.62e+10	1.30e+10	1.44e+10	3.86e+07	1.50e+10	2.80e+07
f_2	mean	1.41e+11 +	2.74e+13 +	6.93e+12 +	3.18e+13 +	1.01e+13 +	7.52e+12 +	4.24e+10 +	2.47e+13 +	**1.19e+10**
	std	5.27e+10	2.33e+12	1.36e+12	2.17e+12	1.76e+12	1.61e+12	5.55e+09	1.86e+12	2.74e+09
f_3	mean	3.37e+10 +	9.69e+12 +	2.36e+12 +	1.13e+13 +	3.21e+12 +	2.62e+12 +	1.54e+10 +	8.36e+12 +	**4.83e+09**
	std	1.08e+10	7.00e+11	4.02e+11	7.03e+11	4.74e+11	4.53e+11	1.57e+09	5.51e+11	1.09e+09
f_4	mean	9.23e+03 +	5.47e+06 +	1.41e+05 +	8.26e+06 +	1.81e+05 +	3.29e+04 +	**2.06e+03** −	2.78e+06 +	2.08e+03
	std	2.17e+03	1.84e+06	6.70e+04	3.40e+06	1.11e+05	1.89e+04	1.24e+01	8.17e+05	3.05e+01
f_5	mean	4.73e+07 +	7.57e+08 +	1.76e+08 +	5.22e+08 +	1.07e+08 +	3.39e+07 +	1.96e+07 +	4.12e+08 +	**8.08e+06**
	std	1.41e+07	2.39e+08	4.42e+07	1.13e+08	3.39e+07	1.38e+07	4.61e+06	1.54e+08	2.32e+06
f_6	mean	1.90e+07 +	1.88e+10 +	8.17e+07 +	1.15e+10 +	1.70e+08 +	3.76e+07 +	1.81e+05 +	1.13e+10 +	**1.21e+05**
	std	9.79e+07	8.68e+09	8.80e+07	4.72e+09	2.21e+08	8.65e+07	4.27e+04	5.26e+09	3.47e+04
f_7	mean	4.59e+07 +	2.01e+10 +	8.66e+08 +	1.04e+10 +	4.92e+08 +	5.65e+07 +	1.29e+07 +	1.10e+10 +	**7.05e+06**
	std	3.22e+07	6.26e+09	3.07e+08	3.91e+09	1.97e+08	3.30e+07	3.77e+06	3.53e+09	3.00e+06
f_8	mean	2.48e+04 +	2.58e+04 +	1.86e+04 +	1.62e+04 +	2.11e+04 +	1.10e+04 ≈	1.86e+04 +	2.08e+04 +	**8.89e+03**
	std	5.76e+03	2.50e+03	5.17e+03	2.41e+03	1.77e+03	4.15e+03	2.69e+03	3.74e+03	3.32e+03
f_9	mean	4.33e+04 +	1.78e+05 +	9.41e+04 +	2.42e+05 +	1.25e+05 +	9.19e+04 +	5.80e+03 +	1.65e+05 +	**4.77e+03**
	std	3.69e+04	7.05e+03	1.98e+04	2.57e+04	1.50e+04	2.36e+04	8.33e+02	6.58e+03	4.31e+02
f_{10}	mean	4.14e+03 +	3.22e+04 +	9.26e+03 +	3.95e+04 +	8.77e+03 +	6.78e+03 +	3.75e+03 +	2.52e+04 +	**3.63e+03**
	std	1.79e+02	4.94e+03	1.05e+03	5.96e+03	1.04e+03	8.55e+02	1.10e+02	3.04e+03	8.60e+01
+/≈/−:		10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	9/1/0	9/0/1	10/0/0	

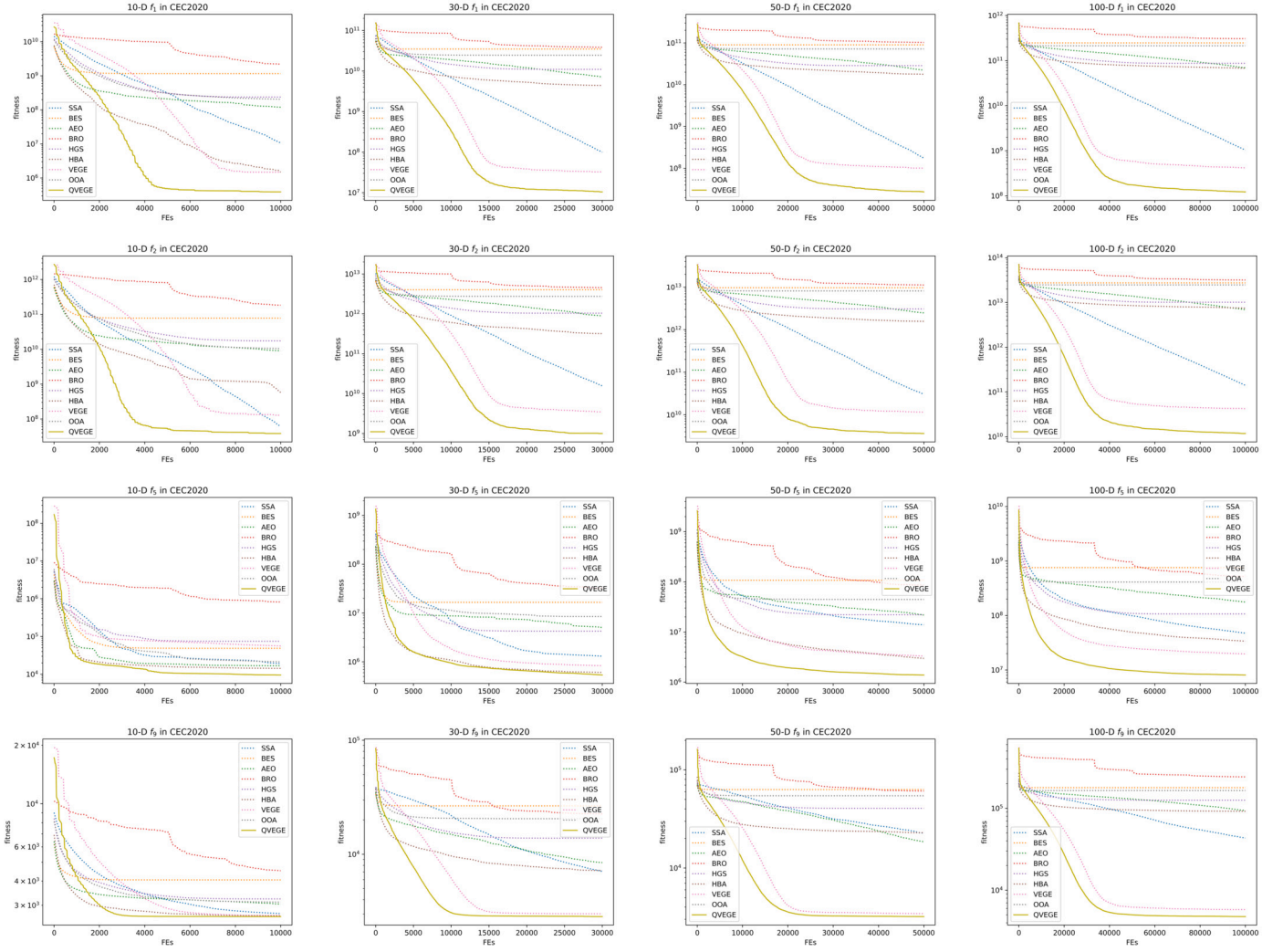

Fig. 5. Convergence curves of representative functions on CEC2020 among nine MAs.

Table 10
Experimental and statistical results on twelve engineering optimization problems.

Func.		SSA	BES	AEO	BRO	HGS	HBA	VEGE	OOA	QVEGE
WBP	mean	3.294e+00 +	1.724e+00 ≈	2.009e+00 +	2.143e+00 +	2.996e+00 +	1.734e+00 +	2.009e+00 +	2.267e+00 +	**1.696e+00**
	std	7.942e-01	8.867e-02	3.708e-01	2.046e-01	6.593e-01	1.666e-01	1.591e-01	3.544e-01	1.094e-02
	best	1.810e+00	1.683e+00	1.685e+00	1.832e+00	1.838e+00	1.683e+00	1.758e+00	1.720e+00	1.684e+00
	worst	5.085e+00	2.137e+00	3.049e+00	2.751e+00	4.535e+00	2.445e+00	2.478e+00	3.377e+00	1.727e+00
PVP	mean	7.449e+04 +	7.865e+03 ≈	7.317e+03 ≈	2.195e+04 +	3.553e+04 +	7.064e+03 ≈	9.901e+03 +	**6.377e+03** ≈	7.989e+03
	std	1.136e+05	2.102e+03	3.408e+03	8.395e+03	5.553e+04	2.301e+03	2.173e+03	3.469e+03	1.317e+03
	best	2.892e+03	2.892e+03	2.892e+03	1.117e+04	2.892e+03	2.892e+03	3.737e+03	2.892e+03	2.909e+03
	worst	5.133e+05	1.178e+04	1.417e+04	4.727e+04	2.044e+05	8.334e+03	1.630e+04	1.145e+04	8.392e+03
CSP	mean	1.592e+06 +	6.076e-03 −	6.076e-03 −	1.051e-02 +	6.622e-03 ≈	**6.076e-03** −	7.548e-03 +	6.076e-03 −	6.078e-03
	std	8.573e+06	1.735e-18	1.735e-18	2.974e-03	8.801e-04	1.735e-18	1.245e-03	1.919e-10	4.180e-06
	best	6.076e-03	6.076e-03	6.076e-03	6.857e-03	6.076e-03	6.076e-03	6.206e-03	6.076e-03	6.076e-03
	worst	4.776e+07	6.076e-03	6.076e-03	2.102e-02	9.204e-03	6.076e-03	1.071e-02	6.076e-03	6.093e-03
SRD	mean	4.169e+03 +	2.999e+03 −	3.175e+03 ≈	3.629e+03 +	3.773e+03 +	**2.988e+03** −	3.412e+03 +	3.262e+03 +	3.034e+03
	std	7.693e+02	9.151e+00	5.103e+02	5.467e+02	8.056e+02	5.232e-01	4.093e+02	4.029e+02	3.263e+01
	best	3.011e+03	2.987e+03	2.988e+03	3.072e+03	2.989e+03	2.987e+03	3.038e+03	2.995e+03	3.000e+03
	worst	5.541e+03	3.031e+03	5.101e+03	5.060e+03	5.855e+03	2.989e+03	4.642e+03	4.449e+03	3.130e+03
TBTD	mean	2.664e+02 +	**2.639e+02** ≈	2.639e+02 ≈	2.641e+02 +	2.660e+02 +	2.639e+02 ≈	2.640e+02 +	2.639e+02 ≈	2.639e+02
	std	2.278e+00	1.468e-14	1.071e-03	1.276e-01	2.455e+00	9.258e-04	1.898e-01	4.630e-03	3.223e-12
	best	2.639e+02	2.639e+02	2.639e+02	2.639e+02	2.639e+02	2.639e+02	2.639e+02	2.639e+02	2.639e+02
	worst	2.707e+02	2.639e+02	2.639e+02	2.644e+02	2.734e+02	2.639e+02	2.646e+02	2.639e+02	2.639e+02
GTD	mean	**0.000e+00** ≈	2.585e-16 +	**0.000e+00** ≈	1.562e-09 +	**0.000e+00** ≈	2.516e-14 +	1.339e-13 +	**0.000e+00** ≈	**0.000e+00**
	std	0.000e+00	1.128e-15	0.000e+00	2.134e-09	0.000e+00	9.476e-14	7.210e-13	0.000e+00	0.000e+00
	best	0.000e+00	6.520e-30	0.000e+00	2.052e-12	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
	worst	0.000e+00	6.267e-15	0.000e+00	9.269e-09	0.000e+00	5.113e-13	4.017e-12	0.000e+00	0.000e+00

Table 11
Experimental and statistical results on twelve engineering optimization problems (continued).

Func.		SSA	BES	AEO	BRO	HGS	HBA	VEGE	OOA	QVEGE
CBD	mean	4.606e+00 +	1.346e+00 ≈	1.440e+00 +	2.812e+00 +	1.567e+00 +	**1.340e+00** −	1.357e+00 +	1.383e+00 +	1.344e+00
	std	1.506e+00	1.045e-02	7.880e-02	7.032e-01	1.595e-01	1.375e-04	7.452e-03	3.421e-02	3.444e-03
	best	1.558e+00	1.340e+00	1.343e+00	1.688e+00	1.384e+00	1.340e+00	1.343e+00	1.342e+00	1.340e+00
	worst	7.487e+00	1.380e+00	1.618e+00	4.871e+00	2.159e+00	1.341e+00	1.374e+00	1.474e+00	1.351e+00
IBD	mean	1.765e-04 +	**1.746e-04** −	**1.746e-04** −	1.749e-04 ≈	1.751e-04 ≈	1.746e-04 −	1.765e-04 +	1.746e-04 −	1.750e-04
	std	7.723e-06	5.421e-20	5.421e-20	1.601e-06	1.862e-06	1.583e-16	7.078e-07	3.281e-16	3.260e-07
	best	1.746e-04	1.746e-04	1.746e-04	1.746e-04	1.746e-04	1.746e-04	1.751e-04	1.746e-04	1.746e-04
	worst	2.136e-04	1.746e-04	1.746e-04	1.835e-04	1.836e-04	1.746e-04	1.779e-04	1.746e-04	1.756e-04
TCD	mean	3.133e+01 +	**3.015e+01** ≈	3.015e+01 ≈	3.042e+01 +	3.114e+01 +	3.015e+01 ≈	3.024e+01 +	3.015e+01 ≈	3.015e+01
	std	1.125e+00	3.553e-15	6.061e-12	1.553e-01	1.078e+00	3.534e-11	9.404e-02	3.095e-06	2.332e-12
	best	3.017e+01	3.015e+01	3.015e+01	3.022e+01	3.015e+01	3.015e+01	3.015e+01	3.015e+01	3.015e+01
	worst	3.455e+01	3.015e+01	3.015e+01	3.078e+01	3.454e+01	3.015e+01	3.050e+01	3.015e+01	3.015e+01
PLD	mean	3.797e+02 +	3.686e+01 +	1.070e+02 +	1.899e+02 +	3.480e+02 +	5.098e+01 +	1.665e+02 +	7.321e+01 +	**1.118e+00**
	std	3.231e+02	6.842e+01	1.131e+02	1.206e+02	2.695e+02	7.626e+01	1.709e+02	1.060e+02	5.514e-02
	best	1.934e+00	1.057e+00	1.058e+00	1.970e+01	1.722e+00	1.057e+00	1.154e+00	1.057e+00	1.058e+00
	worst	1.386e+03	1.846e+02	3.188e+02	4.130e+02	1.107e+03	1.675e+02	7.297e+02	3.248e+02	1.245e+00
CBHD	mean	7.773e+00 +	6.862e+00 −	6.985e+00 +	7.422e+00 +	9.419e+00 +	**6.846e+00** −	7.053e+00 +	7.086e+00 +	6.876e+00
	std	7.375e-01	3.262e-02	9.048e-02	2.837e-01	1.081e+00	7.163e-03	1.692e-01	2.052e-01	3.086e-02
	best	6.964e+00	6.843e+00	6.846e+00	6.977e+00	7.745e+00	6.843e+00	6.889e+00	6.849e+00	6.844e+00
	worst	9.981e+00	6.982e+00	7.137e+00	8.398e+00	1.218e+01	6.878e+00	7.538e+00	7.623e+00	6.956e+00
RCB	mean	1.682e+02 +	1.665e+02 +	1.664e+02 +	1.646e+02 ≈	1.688e+02 +	**1.599e+02** −	1.650e+02 ≈	1.668e+02 +	1.632e+02
	std	2.558e+00	9.138e-01	1.267e+00	2.578e+00	2.868e+00	1.850e+00	1.514e-01	1.168e+00	2.392e+00
	best	1.636e+02	1.628e+02	1.606e+02	1.595e+02	1.636e+02	1.594e+02	1.648e+02	1.636e+02	1.600e+02
	worst	1.722e+02	1.668e+02	1.668e+02	1.671e+02	1.750e+02	1.668e+02	1.653e+02	1.722e+02	1.668e+02
+/≈/−:		11/1/0	3/5/4	5/5/2	10/2/0	9/3/0	3/3/6	11/1/0	6/4/2	
Ave. rank:		8.25	2.66	3.91	6.75	7.58	**2.58**	5.83	4.66	2.75

we list some representative MAs and their computational complexity in Table 13.

Through the listed computational complexity of representative MAs, we can conclude that when a specific MA involves the sort operator in the iteration phase, it has an identical computational complexity with QVEGE; otherwise, it has a lighter computational complexity than QVEGE. Given the superior performance and significant improvement of QVEGE, the higher computational complexity is still acceptable.

5.2. Performance analysis on CEC2020

The CEC2020 benchmark functions exhibit a diverse range of characteristics, including rotation, shifting, and composition, allowing for a comprehensive evaluation of optimization algorithms from various perspectives. For instance, f_1 is an unimodal function with a single optimum, which makes it possible to evaluate the exploitation capacity of algorithms. As demonstrated in Tables 6, 7, 8, and 9, QVEGE

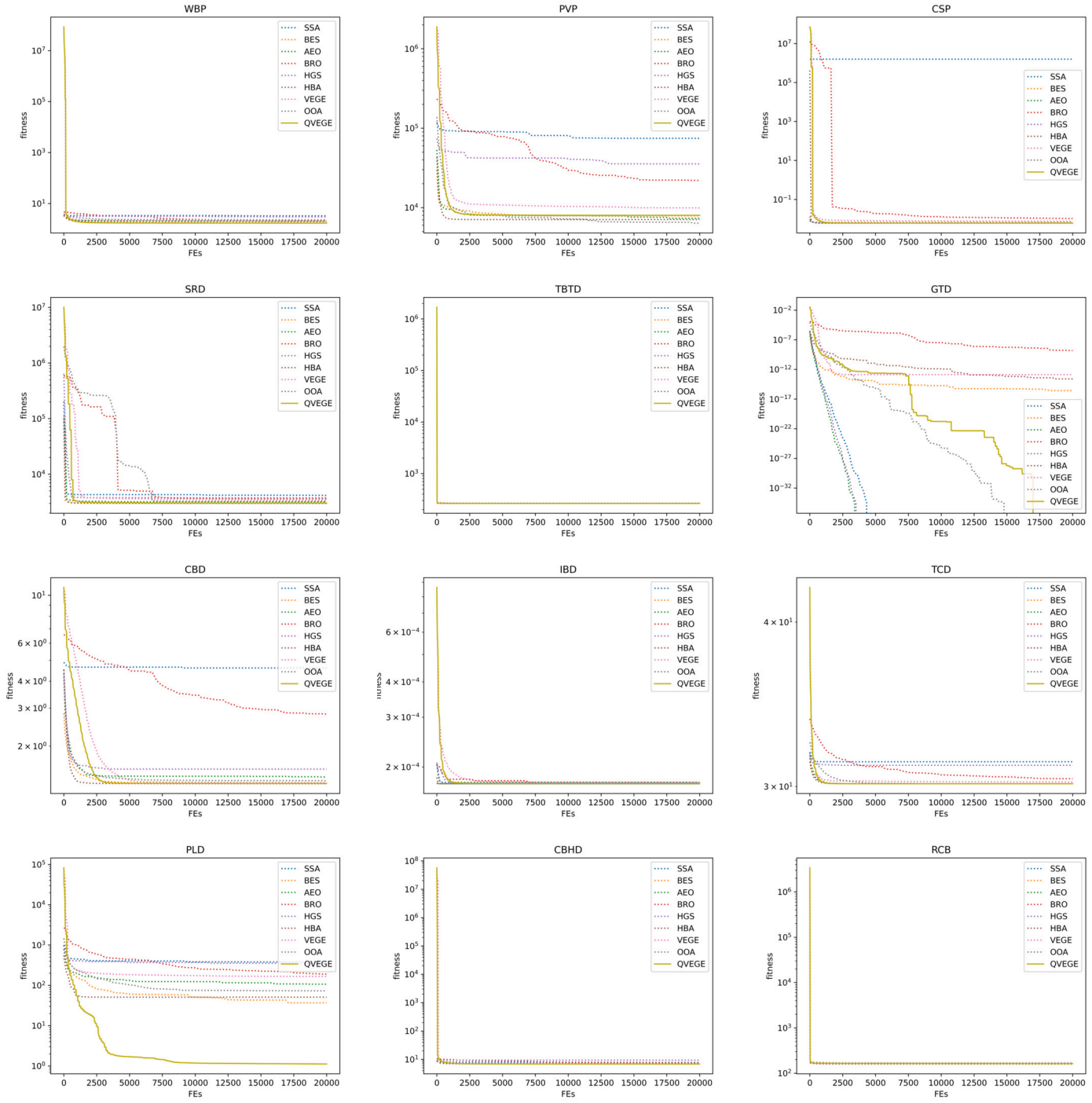

Fig. 6. Convergence curves of engineering problems among nine MAs.

Table 12
Experimental results on WSN coverage optimization problems.

Prob.		SSA	BES	AEO	BRO	HGS	HBA	VEGE	OOA	QVEGE
# of sensors = 32	mean	70.8% +	73.3% +	72.5% +	70.1% +	70.7% +	76.5% +	80.6% +	72.5% +	**82.0%**
	std	2.3%	1.7%	1.7%	1.0%	2.1%	2.5%	1.3%	2.0%	1.9%
# of sensors = 42	mean	79.5% +	82.4% +	81.6% +	79.2% +	81.0% +	83.9% +	89.7% +	81.4% +	**91.4%**
	std	2.0%	1.7%	1.7%	1.1%	1.7%	2.1%	0.9%	1.7%	1.6%
# of sensors = 54	mean	87.9% +	89.1% +	89.7% +	87.2% +	88.4% +	90.3% +	95.0% +	89.2% +	**96.1%**
	std	1.6%	1.6%	1.9%	1.3%	1.6%	1.9%	1.0%	1.7%	1.0%

consistently outperforms the competitor algorithms, and we owe this success to that QVEGE inherits the robust structure of VEGE and further achieves the improvements through intelligent search operator selection by Q-learning.

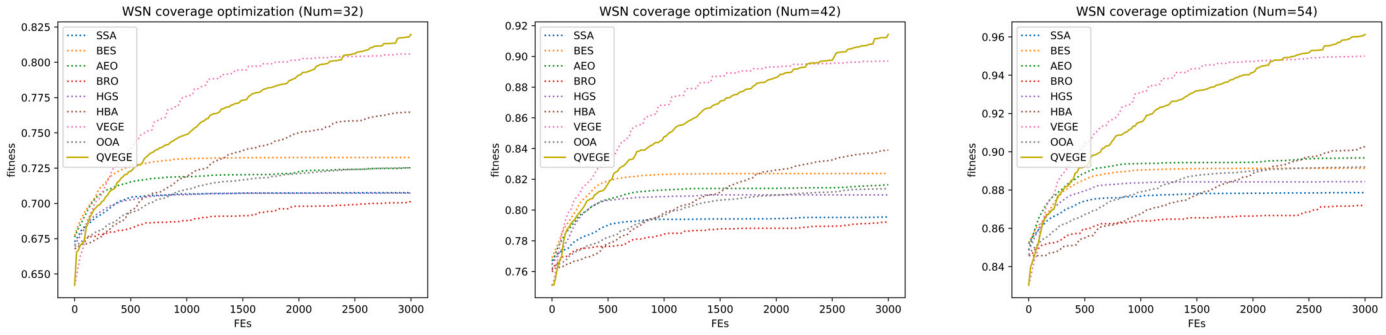

Fig. 7. Convergence curves of WSN coverage problems among nine MAs.

Table 13
Computational complexity of representative MAs.

MAs	Computational complexity
GA [70]	$O(T \times N \times (D + \log N))$
DE [71]	$O(T \times N \times D)$
PSO [72]	$O(T \times N \times D)$
GWO [11]	$O(T \times N \times (D + \log N))$

Beyond f_1, the rest functions have various categories, such as multimodal functions (f_2 to f_4), hybrid functions (f_5 to f_7), and composite functions (f_8 to f_{10}). These functions pose considerable challenges due to their complex fitness landscapes and multiple local optima. In most instances, QVEGE demonstrates superior performance, with the exception of 100-D f_4, where it is significantly worse than the original VEGE. However, these results collectively and practically prove the efficiency and effectiveness of our proposed QVEGE.

It's worth noting that VEGE and QVEGE both start with an initial population size of 10, while the other competitor algorithms are set to 100. Consequently, the convergence curves in Fig. 5 demonstrate that the initial fitness of QVEGE is comparatively worse. Nonetheless, the rapid convergence rate enables QVEGE to swiftly outperform competitor algorithms, which also illustrates the superiority of QVEGE.

5.3. Performance analysis on engineering problems

Tables 10 and 11 provide a comprehensive summary of the experimental and statistical outcomes from nine MAs across twelve distinct engineering optimization challenges. Visual representation of the convergence trajectories is presented in Fig. 6. While our proposed QVEGE algorithm is not the best algorithm observed from the performance indicators of average rank and statistical summary, a notable performance improvement can be observed between VEGE and Q-VEGE.

In the meantime, it is worth noting that QVEGE performs best on the welded beam problem (WBP), gear train problem (GTP), and piston lever problem (PLD). Furthermore, the inferiority of the average rank between the best optimizer HBA and the second-best optimizer BES is slight. Consequently, QVEGE is a promising and successful approach for addressing real-world engineering optimization problems.

Nonetheless, we also notice that QVEGE exhibits poor performance in the compression spring problem (CSP) and the I-beam problem (IBD). This inferiority can be attributed to the inherent design of VEGE which conduces it is not good at dealing with these types of problems, and we prefer to use the No Free Lunch Theory to explain this phenomenon: No Free Lunch Theory posits that if one algorithm performs exceptionally well in a specific class of problems, it must inherently exhibit a trade-off by performing less effectively in the rest classes of problems. This theory implies that any pair of algorithms will demonstrate identical averaging performance across all conceivable problem domains.

5.4. Performance analysis on WSN coverage optimization problem

Our research extends to the evaluation of QVEGE in real-world applications, specifically the WSN coverage optimization problem. We conduct a comprehensive simulation experiment including three distinct experimental settings, with the number of sensors set at 32, 42, and 54, respectively.

The results from both experimental and statistical analyses practically prove the excellent performance of our proposed QVEGE. In all instances, it significantly outperforms the eight competitor algorithms employed in this study. Moreover, the standard deviation observed across multiple trial runs is also acceptable. This not only underscores the stability of QVEGE but also confirms its superiority when dealing with complex real-world applications.

5.5. Open topics for further improvements

In summary, the improvement from the conventional VEGE and QVEGE can be observed from the above numerical experiments, and the competitiveness of QVEGE compared with other state-of-the-art MA approaches is also obvious. We attribute this significant improvement to the incorporation of three key elements: the Latin sampling technique, the well-designed exploitation and exploration archive, and the integration of the Q-learning technique. While the Latin sampling technique performs similarly to simple random initialization in low-dimensional search spaces, it exhibits notable advantages in high-dimensional search spaces, which ensures a more uniform distribution of the initial population and effectively mitigates premature optimization. The exploitation and exploration archive design greatly augments the diversity of search operators to further alleviate potential search structural bias during optimization [73,74]. Additionally, the Q-learning ensemble leverages experiential knowledge to intelligently and efficiently determine subsequent search operators. Comprehensive numerical experiments confirm the efficiency and superior performance of our proposed QVEGE across a spectrum of optimization tasks.

Despite the success of QVEGE as outlined above, there are still unsolved issues that warrant attention. The utilization of the Q-learning technique does introduce a higher practical computational complexity compared to conventional VEGE. Besides, as a stochastic optimization approach, QVEGE still suffers from the No Free Lunch Theory, meaning its superior performance in unknown optimization tasks is not guaranteed. Consequently, we provide several open topics to further advance both VEGE and QVEGE.

5.5.1. Dynamic parameter adaptation of VEGE and QVEGE

The original VEGE and QVEGE have fixed hyper-parameters: The growth cycle GC for each plant is identical as well as the seeding ability. However, in nature, plants have different growth statuses even if they are the same species, e.g., the plants in the fertilized area have faster growth speed and longer longevity, while the plants in the poor area are on the contrary. Furthermore, adequate nutrition also allows

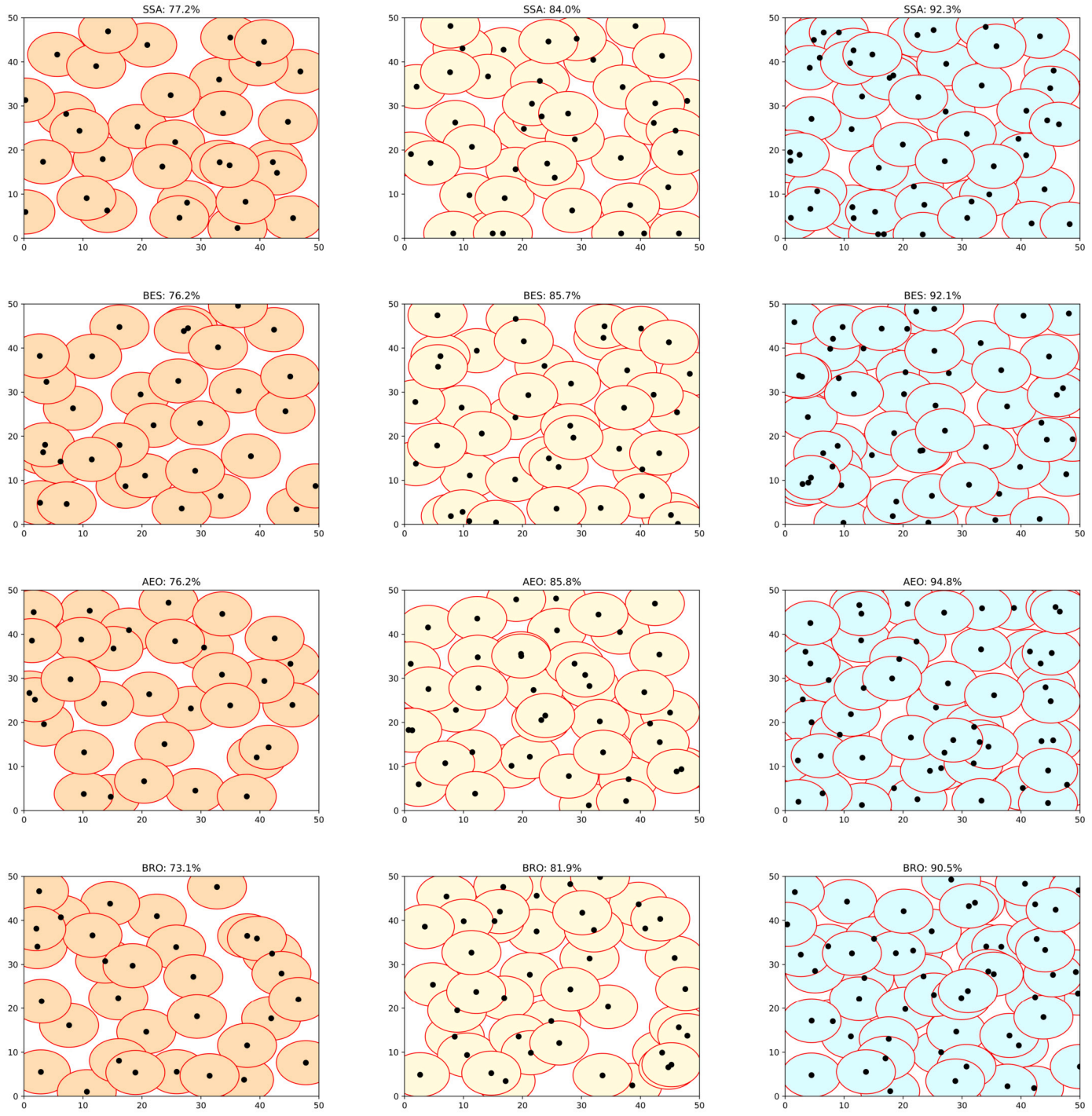

Fig. 8. The visualized optimal solutions found by nine MAs among 30 trial runs.

them to generate more seeds. To simulate the growth of vegetation more precisely, we suggest developing a dynamic version of VEGE and QVEGE, which can automatically determine the parameters based on the fitness value of individuals, iteration times, and other factors.

5.5.2. *Hybridization with other algorithms*

Another potential topic to improve the performance of VEGE and QVEGE is to hybridize with other efficient MAs. Commonly, hybridization is an effective methodology to enhance the optimization performance: Bi et al. [75] and Nadim et al. [76] developed the hybrid versions of the whale optimization algorithm with DE for single- and multi-objective optimization. Chen et al. [77] suggested combining the DE with the Henry gas solubility optimizer to solve pressure retarded osmosis optimization problems. Raamesh et al. [78] hybridized the battle royale optimization with the remora optimization algorithm to solve the challenge in test case selection and prioritization problems. Many effective hybrid techniques have been proposed to tackle various optimization problems, and we recommend hybridizing the VEGE or QVEGE with other optimization approaches to strengthen the global optimization ability.

5.5.3. *Real-world applications*

Many MA approaches have been adopted to solve real-world problems successfully: Pierre et al. [79] adopted the genetic algorithm (GA)

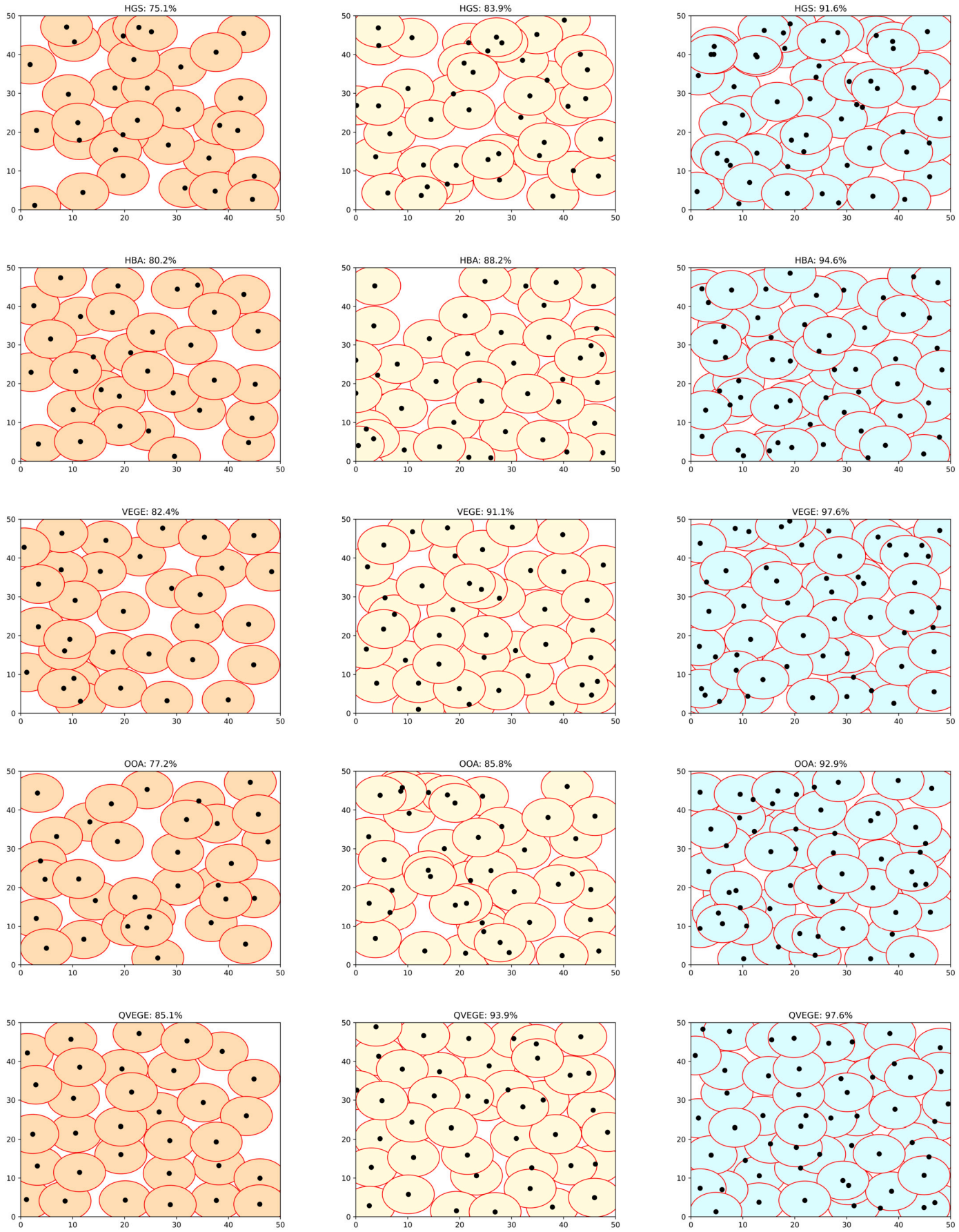

Fig. 9. The visualized optimal solutions found by nine MAs among 30 trial runs (continued).

to optimize the base station placement of mobile communication considering the human exposure to electromagnetic waves, received signal strength measured from users' smartphones, and the location of base station antennas or their radiation pattern. Pierezan et al. [80] developed a cultural coyote optimization algorithm to deal with the operation optimization of a heavy-duty gas turbine placed in Brazil and used in power generation. The ultimate goal is to find the best valve setup to reduce fuel consumption subject to cope with environmental and physical constraints from its operation. Xiao et al. [81] first constructed a multi-component energy model based on the energy characteristics analysis of dry gear hobbing machines, and then a modified multi-objective imperialist competitive algorithm was employed to optimize this mathematical model. Massive simulation experiments reveal the robustness and effectiveness of MAs. In future research, we will focus on extending the QVEGE to real-world applications and attempting to solve various optimization problems.

6. Conclusion

In this paper, we focus on further improving the performance of the original VEGE and propose an improved version named QVEGE. We design an exploitation search strategy archive and an exploration search strategy archive. Q-learning cooperating with the ε-greedy scheme is adopted to select the search strategy automatically during the optimization. To evaluate the performance of the proposed QVEGE, we implement comprehensive experiments on CEC2020, twelve engineering optimization problems, and WSN coverage optimization problems with eight state-of-the-art MAs to investigate the performance of our proposed QVEGE. The experimental results and statistical analysis show that QVEGE achieves great improvements compared with VEGE and is competitive with the other MAs.

In future research, we will hybridize the VEGE or QVEGE with some advanced MAs to further improve competitiveness and extend QVEGE to other application scenarios.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The source code of this research can be downloaded from https://github.com/RuiZhong961230/QVEGE.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Numbers JP20K11967 and 21A402 and JST SPRING, Grant Number JPMJSP2119.

References

[1] Majdi Mafarja, Seyedali Mirjalili, Whale optimization approaches for wrapper feature selection, Appl. Soft Comput. 62 (2018) 441–453.
[2] Mohammad H. Nadimi-Shahraki, Hoda Zamani, Seyedali Mirjalili, Enhanced whale optimization algorithm for medical feature selection: a Covid-19 case study, Comput. Biol. Med. 148 (2022) 105858.
[3] Jayashree Piri, Puspanjali Mohapatra, Raghunath Dey, Biswaranjan Acharya, Vassilis C. Gerogiannis, Andreas Kanavos, Literature review on hybrid evolutionary approaches for feature selection, Algorithms 16 (3) (2023).
[4] Nizar Alkayem, Maosen Cao, Yufeng Zhang, Mahmoud Bayat, Zhongqing Su, Structural damage detection using finite element model updating with evolutionary algorithms: a survey, Neural Comput. Appl. 30 (2018) 389.
[5] Hamed Fathnejat, Behrouz Ahmadi-Nedushan, An efficient two-stage approach for structural damage detection using meta-heuristic algorithms and group method of data handling surrogate model, Front. Struct. Civ. Eng. 05 (2020).
[6] Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G. Yen, Kay Chen Tan, A survey on evolutionary neural architecture search, IEEE Trans. Neural Netw. Learn. Syst. 34 (2) (2023) 550–570.
[7] Zeki Kuş, Musa Aydın, Berna Kiraz, Burhanettin Can, Neural architecture search using metaheuristics for automated cell segmentation, in: Metaheuristics, Springer International Publishing, Cham, 2023, pp. 158–171.
[8] Jeng-Shyang Pan, Ru-Yu Wang, Shu-Chuan Chu, Kuo-Kun Tseng, Fang Fan, A quasi-affine transformation evolutionary algorithm enhanced by hybrid Taguchi strategy and its application in fault detection of wireless sensor network, Symmetry 15 (4) (2023).
[9] Wu Deng, Hongcheng Ni, Yi Liu, Huiling Chen, Huimin Zhao, An adaptive differential evolution algorithm based on belief space and generalized opposition-based learning for resource allocation, Appl. Soft Comput. 127 (2022) 109419.
[10] Rui Zhong, Enzhi Zhang, Masaharu Munetomo, Cooperative coevolutionary differential evolution with linkage measurement minimization for large-scale optimization problems in noisy environments, Complex Intell. Syst. 9 (2023) 4439–4456.
[11] Seyedali Mirjalili, Seyed Mohammad Mirjalili, Andrew Lewis, Grey Wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[12] S. Shadravan, H.R. Naji, V.K. Bardsiri, The sailfish optimizer: a novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems, Eng. Appl. Artif. Intell. 80 (2019) 20–34.
[13] Shimin Li, Huiling Chen, Mingjing Wang, Ali Asghar Heidari, Seyedali Mirjalili, Slime mould algorithm: a new method for stochastic optimization, Future Gener. Comput. Syst. 111 (2020) 300–323.
[14] Amit K. Bairwa, Sandeep Joshi, Dilbag Singh, Dingo optimizer: a nature-inspired metaheuristic approach for engineering problems, Math. Probl. Eng. (2021) 2021.
[15] Pieter-Tjerk Boer, Dirk Kroese, Shie Mannor, Reuven Rubinstein, A tutorial on the cross-entropy method, Ann. Oper. Res. 134 (19–67) (2005) 02.
[16] Seyedali Mirjalili Sca, A sine cosine algorithm for solving optimization problems, Knowl.-Based Syst. 96 (2016) 120–133.
[17] Iman Ahmadianfar, Omid Bozorg-Haddad, Xuefeng Chu, Gradient-based optimizer: a new metaheuristic optimization algorithm, Inf. Sci. 540 (2020) 131–159.
[18] Siamak Talatahari, Mahdi Azizi, Chaos game optimization: a novel metaheuristic algorithm, Artif. Intell. Rev. 54 (917–1004) (2021) 02.
[19] Shu-Chuan Chu, Pei-wei Tsai, Jeng-Shyang Pan, Cat swarm optimization, in: Qiang Yang, Geoff Webb (Eds.), PRICAI 2006: Trends in Artificial Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 854–858.
[20] Seyedali Mirjalili, Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm, Knowl.-Based Syst. 89 (2015) 228–249.
[21] Seyedali Mirjalili, Amir H. Gandomi, Seyedeh Zahra Mirjalili, Shahrzad Saremi, Hossam Faris, Seyed Mohammad Mirjalili, Salp swarm algorithm: a bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (2017) 163–191.
[22] El-Sayed M. El-kenawy, Nima Khodadadi, Seyedali Mirjalili, Abdelaziz A. Abdelhamid, Marwa M. Eid, Abdelhameed Ibrahim, Greylag goose optimization: nature-inspired optimization algorithm, Expert Syst. Appl. 238 (2024) 122147.
[23] Seyedali Mirjalili, Seyed Mirjalili, Abdolreza Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization, Neural Comput. Appl. 27 (2016) 495.
[24] Zhenglei Wei, Changqiang Huang, Xiaofei Wang, Tong Han, Yintong Li, Nuclear reaction optimization: a novel and powerful physics-based algorithm for global optimization, IEEE Access 7 (2019) 66084–66109.
[25] Weiguo Zhao, Liying Wang, Zhenxing Zhang, Atom search optimization and its application to solve a hydrogeologic parameter estimation problem, Knowl.-Based Syst. 163 (2019) 283–304.
[26] Afshin Faramarzi, Mohammad Heidarinejad, Brent Stephens, Seyedali Mirjalili, Equilibrium optimizer: a novel optimization algorithm, Knowl.-Based Syst. 191 (2020) 105190.
[27] Robert Reynolds, An introduction to cultural algorithms, in: Evolutionary Programming — Proceedings of the Third Annual Conference, 1994, pp. 131–139, 01.
[28] Yuhui Shi, Brain storm optimization algorithm, in: Ying Tan, Yuhui Shi, Yi Chai, Guoyin Wang (Eds.), Advances in Swarm Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 303–309.
[29] Jinhao Zhang, Mi Xiao, Liang Gao, Quanke Pan, Queuing search algorithm: a novel metaheuristic algorithm for solving engineering optimization problems, Appl. Math. Model. 63 (2018) 464–490.
[30] Taymaz Akan-R. Farshi, Battle royale optimization algorithm, Neural Comput. Appl. (2021) 33:1139–1157, 02.
[31] Ali Hassan, Salwani Abdullah, Kamal Zamli, Rozilawati Razali, Q-learning whale optimization algorithm for test suite generation with constraints support, Neural Comput. Appl. 35 (2023) 1.
[32] Qusay Shihab Hamad, Hussein Samma, Shahrel Azmin Suandi, Junita Mohamad-Saleh, Q-learning embedded sine cosine algorithm (qlesca), Expert Syst. Appl. 193 (2022) 116417.
[33] Huachao Dong, Zuomin Dong, Surrogate-assisted grey wolf optimization for high-dimensional, computationally expensive black-box problems, Swarm Evol. Comput. 57 (2020) 100713.
[34] Rui Zhong, Enzhi Zhang, Masaharu Munetomo, Cooperative coevolutionary surrogate ensemble-assisted differential evolution with efficient dual differential grouping for large-scale expensive optimization problems, Complex Intell. Syst. (2023) 1–21.
[35] Jun Yu, Vegetation evolution: an optimization algorithm inspired by the life cycle of plants, Int. J. Comput. Intell. Appl. 21 (2022) 06.

[36] Jun Yu, Hideyuki Takagi, Accelerating vegetation evolution with mutation strategy and gbased growth strategy, in: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), 2019, pp. 3033–3039.
[37] Jun Yu, Hideyuki Takagi, Multi-species generation strategy-based vegetation evolution, in: 2020 IEEE Congress on Evolutionary Computation (CEC), 2020, pp. 1–6.
[38] Jun Yu, Hideyuki Takagi, Performance analysis of vegetation evolution, in: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), 2019, pp. 2214–2219.
[39] Rui Zhong, Fei Peng, Enzhi Zhang, Jun Yu, Masaharu Munetomo, Vegetation evolution with dynamic maturity strategy and diverse mutation strategy for solving optimization problems, Biomimetics 8 (6) (2023).
[40] Esmaeil Kaffashi, Mehdi Taghizade Shoorabi, Sahar Hemmatian Bojnourdi, Coverage optimization in wireless sensor networks, in: 2014 4th International Conference on Computer and Knowledge Engineering (ICCKE), 2014, pp. 322–327.
[41] Jianghao Yin, Na Deng, Jindan Zhang, Wireless sensor network coverage optimization based on Yin–Yang pigeon-inspired optimization algorithm for Internet of Things, Int. Things 19 (2022) 100546.
[42] Jinyan Liang, Min Tian, Yang Liu, Jie Zhou, Coverage optimization of soil moisture wireless sensor networks based on adaptive Cauchy variant butterfly optimization algorithm, Sci. Rep. 12 (2022) 07.
[43] Seok Kwon, Jin Kim, Coverage ratio in the wireless sensor networks using Monte Carlo simulation, 2008, pp. 235–238, 10.
[44] Suparna Chakraborty, N.K. Goyal, S. Mahapatra, Sieteng Soh, A Monte-Carlo Markov chain approach for coverage-area reliability of mobile wireless sensor networks with multistate nodes, Reliab. Eng. Syst. Saf. 193 (2020) 106662.
[45] Michael Stein, Large sample properties of simulations using Latin hypercube sampling, Technometrics 29 (2) (1987) 143–151.
[46] Zhiwei Zhao, Jingming Yang, Ziyu Hu, Haijun Che, A differential evolution algorithm with self-adaptive strategy and control parameters based on symmetric Latin hypercube design for unconstrained optimization problems, Eur. J. Oper. Res. 250 (1) (2016) 30–45.
[47] Siti Julia Rosli, Hasliza A. Rahim, Khairul Najmy Abdul Rani, Ruzelita Ngadiran, R. Badlishah Ahmad, Nor Zakiah Yahaya, Mohamedfareq Abdulmalek, Muzammil Jusoh, Mohd Najib Mohd Yasin, Thennarasan Sabapathy, Allan Melvin Andrew, A hybrid modified method of the sine cosine algorithm using latin hypercube sampling with the cuckoo search algorithm for optimization problems, Electronics 9 (11) (2020).
[48] Alexei V. Chechkin, Ralf Metzler, Joseph Klafter, Vsevolod Yu. Gonchar, Introduction to the Theory of Lévy Flights, chapter 5, John Wiley & Sons, Ltd, 2008, pp. 129–162.
[49] Essam Houssein, Mohamed Hosny, Salah Kamel, Kashif Hussain, Fatma A. Hashim, Modified Levy flight distribution algorithm for global optimization and parameters estimation of modified three-diode photovoltaic model, Appl. Intell. 09 (2022).
[50] Yong Gao, Hao Zhang, Yingying Duan, Huaifeng Zhang, A novel hybrid pso based on Levy flight and wavelet mutation for global optimization, PLoS ONE 18 (1) (2023) 1–27, 01.
[51] Changting Zhong, Gang Li, Zeng Meng, Wanxin He, Opposition-based learning equilibrium optimizer with Levy flight and evolutionary population dynamics for high-dimensional global optimization problems, Expert Syst. Appl. 215 (2023) 119303.
[52] Mehak Kohli, Sankalap Arora, Chaotic grey wolf optimization algorithm for constrained optimization problems, J. Comput. Des. Eng. 5 (4) (2018) 458–472.
[53] R. Caponetto, L. Fortuna, S. Fazzino, M.G. Xibilia, Chaotic sequences to improve the performance of evolutionary algorithms, IEEE Trans. Evol. Comput. 7 (3) (2003) 289–304.
[54] Jingqiao Zhang, A.C. Sanderson Jade, Adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (945–958) (2009) 11.
[55] Christopher JCH Watkins, Peter Dayan, Q-learning, Mach. Learn. 8 (3) (1992) 279–292.
[56] Eyal Even-Dar, Yishay Mansour, Learning rates for q-learning, in: David Helmbold, Bob Williamson (Eds.), Computational Learning Theory, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 589–604.
[57] Carlos A. Coello Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Comput. Methods Appl. Mech. Eng. 191 (11) (2002) 1245–1287.
[58] Nguyen Van Thieu, Seyedali Mirjalili Mealpy, An open-source library for latest meta-heuristic algorithms in python, J. Syst. Archit. 139 (2023) 102871.
[59] Thieu Nguyen, A framework of optimization functions using numpy (opfunu) for optimization problems, 2020.
[60] Nguyen Van Thieu, Enoppy: a python library for engineering optimization problems, May 2023.
[61] P.N. Suganthan, C.T. Yue, K.V. Price, Problem definitions and evaluation criteria for the cec 2020 special session and competition on single objective bound constrained numerical optimization, in: Technical Report Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2020.
[62] Absalom Ezugwu, Ovre Agushaka, Laith Abualigah, Seyedali Mirjalili, Amir Gandomi, Prairie dog optimization algorithm, Neural Comput. Appl. 34 (2022) 20017.
[63] Jiankai Xue, Bo Shen, A novel swarm intelligence optimization approach: sparrow search algorithm, Syst. Sci. Control Eng. 8 (1) (2020) 22–34.
[64] H. Alsattar, A. Zaidan, Bilal Bahaa, Novel meta-heuristic bald eagle search optimisation algorithm, Artif. Intell. Rev. 53 (2020) 2237.
[65] Weiguo Zhao, Liying Wang, Zhenxing Zhang, Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm, Neural Comput. Appl. 32 (2020) 1.
[66] Yutao Yang, Huiling Chen, Ali Asghar Heidari, Amir H. Gandomi, Hunger games search: visions, conception, implementation, deep analysis, perspectives, and towards performance shifts, Expert Syst. Appl. 177 (2021) 114864.
[67] Fatma A. Hashim, Essam H. Houssein, Kashif Hussain Mai, S. Mabrouk, Walid Al-Atabany, Honey badger algorithm: new metaheuristic algorithm for solving optimization problems, Math. Comput. Simul. 192 (2022) 84–110.
[68] Mohammad Dehghani, Pavel Trojovský, Osprey optimization algorithm: a new bio-inspired metaheuristic algorithm for solving engineering optimization problems, Front. Mech. Eng. 8 (2023).
[69] Sture Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. 6 (2) (1979) 65–70.
[70] John H. Holland, Genetic algorithms, Sci. Am. 267 (1) (1992) 66–73.
[71] Rainer Storn, Kenneth Price, Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (1997) 341.
[72] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.
[73] Anna V. Kononova, David W. Corne, Philippe De Wilde, Vsevolod Shneer, Fabio Caraffini, Structural bias in population-based algorithms, Inf. Sci. 298 (2015) 468–490.
[74] Fabio Caraffini, Anna V. Kononova, David Corne, Infeasibility and structural bias in differential evolution, Inf. Sci. 496 (2019) 161–179.
[75] Jing Bi, Wenduo Gu, Haitao Yuan, Hybrid whale optimization algorithm with differential evolution and chaotic map operations, in: 2021 IEEE International Conference on Networking, Sensing and Control (ICNSC), vol. 1, 2021, pp. 1–6.
[76] Nadim Rana, Muhammad Shafie Abd Latiff, Shafi'i Muhammad Abdulhamid, Sanjay Misra, A hybrid whale optimization algorithm with differential evolution optimization for multi-objective virtual machine scheduling in cloud computing, Eng. Optim. 54 (12) (2022) 1999–2016.
[77] Yingxue Chen, Linfeng Gou, Huihui Li, A differential evolution based Henry gas solubility optimizer for dynamic performance optimization problems of pro system, Appl. Soft Comput. 125 (2022) 109097.
[78] Lilly Raamesh, S. Radhika, S. Jothi, A cost-effective test case selection and prioritization using hybrid battle royale-based remora optimization, Neural Comput. Appl. 34 (2022) 09.
[79] Pierre Combeau, N. Noe, François Gaudaire, Steve Joumessi, Jean-Benoit Dufour, A numerical simulation system for mobile telephony base station emf exposure using smartphones as probes and a genetic algorithm to improve accuracy, Prog. Electromagn. Res. B 87 (111–129) (2020) 01.
[80] Juliano Pierezan, Gabriel Maidl, Eduardo Massashi Yamao Leandro dos Santos Coelho, Viviana Cocco Mariani, Cultural coyote optimization algorithm applied to a heavy duty gas turbine operation, Energy Convers. Manag. 199 (2019) 111932.
[81] Qinge Xiao, Congbo Li, Ying Tang, Jian Pan, Jun Yu, Xingzheng Chen, Multi-component energy modeling and optimization for sustainable dry gear hobbing, Energy 187 (2019) 115911.

View publication stats