Information Sciences 586 (2022) 424–449

Contents lists available at ScienceDirect

Information Sciences

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

Integrated optimization algorithm: A metaheuristic approach for complicated optimization

Chen Li [a], Guo Chen [a,*], Gaoqi Liang [b], Fengji Luo [c], Junhua Zhao [d,e], Zhao Yang Dong [a]

[a] School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney 2052, Australia
[b] School of Electrical and Electronic Engineering, Nanyang Technological University, 639798, Singapore
[c] School of Civil Engineering, The University of Sydney, Sydney 2006, Australia
[d] School of Science and Engineering, Chinese University of Hong Kong, Shenzhen, Shenzhen 518172, China
[e] Shenzhen Institute of Artificial Intelligence and Robotics for Society, Shenzhen 518172, China

ARTICLE INFO

Article history:
Received 22 May 2021
Received in revised form 28 September 2021
Accepted 14 November 2021
Available online 6 December 2021

Keywords:
Non-convex optimization
Non-differentiable optimization
Non-continuous optimization
Metaheuristic algorithm
Unit commitment
Deep learning

ABSTRACT

This paper proposes an integrated optimization algorithm (IOA) designed for solving complicated optimization problems that are non-convex, non-differentiable, non-continuous, or computationally intensive. IOA is synthesized from 5 sub-algorithms: follower search, leader search, wanderer search, crossover search, and role learning. The follower search finds better solutions by tracing the leaders. The leader search refines current optimal solutions by approaching or deviating from the central point of the population and then executes a single-round coordinate descent. The wanderer search carries out comprehensive search space expansion. The crossover search generates offspring using solutions from superior parents. Role learning automates the process in which a search agent decides whether to become a follower or a wanderer. A global optima estimation framework (GOEF) is proposed to offer guidelines for designing an efficient optimization algorithm, and IOA is proved to attain global optima. A differentiable integrated optimization algorithm (DIOA) that extends gradient descent is put forward to train deep learning models. Empirical case studies conclude that IOA shows a much faster convergence speed and finds better solutions than the other 8 comparative algorithms based on 27 benchmark functions. IOA has also been applied to solve unit commitment problems in the power system and shows satisfactory results. A power line sub-image classification model based on a convolutional neural network (CNN) is optimized by DIOA. Compared with the pure gradient descent approach, DIOA converges significantly faster and obtains a high test set accuracy with much fewer training epochs.

1. Introduction

Numerous problems in scientific branches and technology fields should be tackled by highly effective optimization algorithms. Traditional optimization techniques, *e.g.,* linear programming, quadratic programming, and dynamic programming, can solve optimization problems with special forms. Some complex models suffer from accuracy loss problems because linearization is required before using conventional optimization algorithms. Metaheuristic algorithms, which have no restric-

* Corresponding author.
E-mail addresses: chen.li6@student.unsw.edu.au (C. Li), guo.chen@unsw.edu.au (G. Chen), gaoqi.liang@ntu.edu.sg (G. Liang), fengji.luo@sydney.edu.au (F. Luo), zhaojunhua@cuhk.edu.cn (J. Zhao), joe.dong@unsw.edu.au (Z.Y. Dong).

https://doi.org/10.1016/j.ins.2021.11.043
0020-0255/

tions for the objective function in terms of convexity, differentiability, and continuity, perform a stochastic search in pursuit of global optima inspired by natural phenomena or animals' behaviors. Many metaheuristic algorithms have been proposed: Harris hawks optimization (HHO) [7] is inspired by the cooperative chasing actions of Harris' hawks. Based on the seven-kills strategy, the solution converges quickly as the hawks attack the prey from different directions. In bald eagle search (BES) [1], the solution is obtained by space selection, searching, and swooping procedures. The Salp swarm algorithm (SSA) [20] inspired by the swarming behaviors of salps is effective in finding the optimal solutions. Natural aggregation algorithm (NAA) [18,19] is motivated by the self-regulated activities of group-living animals during the competition and resource sharing process. This dynamic mechanism enables NAA to find global or near-global solutions to non-convex optimization problems. Grey wolf optimizer (GWO) [21] mimics the hierarchy and hunting steps of grey wolves and attains competitive results compared to many metaheuristic algorithms. Bat-inspired algorithm (BA) [28] imitates the echolocation of bats. It strikes a balance between particle swarm optimizer (PSO) [15] and local search controlled by pulse rate and loudness. Variants of prestigious swarm-based optimization algorithms, *e.g.,* differential evolution (DE) [4,24], are also popular in recent years. To avoid cumbersome hyperparameter tuning, a deep neural network optimized by policy gradient is used as a parameter controller for DE [26]. A dual-surrogate-assisted cooperative PSO [11] is introduced to find several optimal solutions using a cost-efficient modal detection strategy. An adaptive region search based on the region encoding scheme proposed in [12] maintains population diversity and accelerates convergence in social learning PSO. However, [29] points out that many swarm-based algorithms are more suitable for optimization in static environments. Different algorithms can be combined to adapt to environmental changes. Therefore, a robust optimization solver should synthesize the advantages of different optimization algorithms.

In this paper, we propose an integrated optimization algorithm that solves complicated optimization problems. A complicated optimization problem has at least one of the following properties: non-convex, non-differentiable, non-continuous, and computationally intensive. IOA is synthesized from five sub-algorithms: follower search, leader search, wanderer search, crossover search, and role learning. The follower search, wanderer search, and role learning are inspired by the aggregation behaviors of group-living animals discussed in NAA [18]. The leader search refines the current optimal solutions by approaching or deviating from the central point of the population and then executes a single round coordinate descent. The genetic algorithm (GA) [8,23] inspired crossover search expands the search space and discovers better solutions. The follower search serves as the most dominant component in IOA because it demonstrates great effectiveness in finding a better solution especially when the objective function is monotonic, convex, or concave within a local region. The other sub-algorithms in IOA either improve the solutions or expand the search space.

Although plenty of metaheuristic algorithms can solve complicated optimization problems effectively, it is challenging to tell whether they can obtain the global optimal solutions from a mathematic perspective. Therefore, we propose a global optima estimation framework (GOEF) that provides guidelines for designing a robust optimization algorithm. IOA is proved to attain global optima according to GOEF given that the search space is sufficiently large.

As deep learning gains popularity in the recent decade, gradient descent has become the de-facto method for neural network training. However, the gradient descent algorithm cannot escape from poor local minima. Vanishing gradient issues make some deep learning models difficult to train, *e.g.,* generative adversarial networks [5]. Consequently, a simplified version of IOA called a differentiable integrated optimization algorithm is proposed to solve differentiable and computationally intensive optimization problems. In DIOA, coordinate descent is replaced by gradient descent in the leader search. The follower search is retained while the wanderer search, role learning, and crossover search are disabled.

To improve the performance of IOA and DIOA, multithreading techniques are applied to make full use of a multi-core CPU and reduce the algorithm execution time. In DIOA, GPU hardware acceleration is enabled for gradient-based calculations.

This paper makes the following 5 main contributions:

(1) We propose a new metaheuristic algorithm IOA, which is synthesized from five sub-algorithms: follower search, leader search, wanderer search, crossover search, and role learning, to swiftly find global optima for complicated optimization problems.
(2) The GOEF is proposed as a guideline for designing an efficient optimization algorithm. Based on the GOEF, mathematical proof shows that IOA can find global optima when the search space is sufficiently large.
(3) IOA wins 22 out of 27 benchmark function contests compared to the other 8 algorithms: HHO, BES, SSA, NAA, GWO, BA, DE, and PSO. IOA also shows a much faster convergence speed and obtains better solutions than the other algorithms, especially in high-dimensional cases.
(4) IOA and the other 5 algorithms: HHO, NAA, GWO, DE, and PSO are applied for solving the unit commitment problem, *i.e.*, a high-dimensional, non-continuous, and non-convex problem in the power system. IOA converges remarkably faster than the other algorithms and creates the optimal power generation schedule on average.
(5) DIOA is proposed for training deep neural networks. Compared with the gradient descent approach, DIOA converges significantly faster and obtains a high test set accuracy with much fewer training epochs.

This paper is organized as follows: Section 2 offers an overview of IOA and explains the five sub-algorithms in detail. Section 3 proposes GOEF and evaluates the follower search, leader search, and wanderer search sub-algorithms using this framework. Section 4 puts forward DIOA which is capable of training a deep neural network. Section 5 compares the performance of IOA, HHO, BES, SSA, NAA, GWO, BA, DE, and PSO algorithms based on 27 benchmark functions. Convergence prop-

erties and execution duration are also evaluated for each procedure of IOA. Section 6 applies IOA, HHO, NAA, GWO, DE, and PSO to solve unit commitment problems in the power system. Section 7 carries out a power line sub-image classification experiment based on DIOA and CNN. Section 8 summarizes the paper.

2. Integrated optimization algorithm

IOA is a combination of 5 sub-algorithms: follower search, leader search, wanderer search, crossover search, and role learning. An overview of IOA is offered and each sub-algorithm is explained in detail.

2.1. Integrated optimization algorithm overview

IOA finds global optima through the collaborative work of many search agents. In this paper, a **search agent** is an independent optimization solver that travels through the search space in pursuit of a better solution. Each search agent has an associated **solution**, **fitness value**, and search space. The solution is a n dimensional input vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, the fitness value is the value of an objective function $f(\boldsymbol{x})$, and the search space denoted by $\mathbb{C}$ is the subspace of real vector space $\Re^n$. Smaller fitness values are better than larger ones. A search agent has three roles to choose from: **leader**, **follower**, and **wanderer**.

IOA consists of 5 sub-algorithms: follower search, leader search, wanderer search, crossover search, and role learning. In the **follower search**, a follower carries out a local search by tracing its leader in pursuit of a better solution. In the **leader search**, a leader that has a superior fitness value performs a local search by approaching or deviating from the central point of the population. Subsequently, the leader executes a single round coordinate descent [27]. In the **wanderer search**, a wanderer accomplishes a global search by randomly picking several search agents as its movement anchors. If a better solution is not found, the wanderer encircles and evades from the best search agent to obtain a better fitness value. In the **crossover search**, superior search agents are selected as parents according to the spinning wheel approach [23] to reproduce high-quality offspring. In **role learning**, the role learner of a search agent decides whether it becomes a follower or a wanderer based on the past cumulative experience gained from the environment. Fig. 1 shows the entire architecture of IOA.

The overall procedures of IOA are as follows. At initialization time, the solution of each search agent is randomly generated, search agents with small fitness values are assigned as initial leaders, and the remaining search agents become followers. As the algorithm proceeds, search agents with outstanding fitness values are elected as new leaders to perform local search operations. A follower or a wanderer selects a new solution using either the follower search or wanderer search algorithm. If the new solution has a better fitness value, the previous solution of the follower or wanderer is replaced by the new solution. Meanwhile, the role learning task is performed by the role learner. Crossover operations are accomplished subsequently to generate superior offspring. Algorithm 1 explains the overall procedures of IOA.

Algorithm 1: Integrated Optimization Algorithm

Input 1: lc = dimension, lsp = 1, v = 0.9, **Input 2**: Random solutions for search agents, **Output**: Solution of the best search agent
for each iteration **do**:
  1) Each leader compares itself to the central point and makes a movement if a better solution is found.
  2) Single round coordinate descent operations are carried out by the leaders.
  3) **for** each follower or wanderer **do** using CPU multithreading:
    if search agent is follower:
      Next solution is obtained by the follower search.
    else if search agent is wanderer:
      Next solution is obtained by the wanderer search.
    end if
    Calculate the fitness value given the new solution.
    if *fitness value*(next solution) < *fitness value*(current solution):
      Current solution ← next solution
      Role learner makes a reward for the search agent.
    else:
      Role learner makes a punishment for the search agent.
    end if
      Role learner transits the role for the search agent.
    end for
  4) High-quality offspring are reproduced by crossover operations.
  5) Search agents are ranked by their fitness values from small to large and leaders are reelected.
end for
Note: lc: leader count; lsp: leader coorindate descent selection probability, v: role learner discounting factor

Fig. 1. Architecture of integrated optimization algorithm.

2.2. Follower search

The follower search has a high probability of obtaining a better solution. Define $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ as the solution of the leader; $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ as the solution of the follower, $\boldsymbol{x} \neq \boldsymbol{y}$; P as a uniform random variable, $P \sim U(0,1)$. The fitness value of the leader is always no greater than the one for the follower, $f(\boldsymbol{y}) \leqslant f(\boldsymbol{x})$. Define the movement vector for $\boldsymbol{x}$ as:

$$\Delta \boldsymbol{x} = 2P(\boldsymbol{y} - \boldsymbol{x}) \tag{1}$$

When $P = 0$, the solution of the follower does not change; when $P = 1$ the follower moves to $\boldsymbol{x}' = 2\boldsymbol{y} - \boldsymbol{x}$ which is on the opposite side of the leader; when $P = 0.5$, the follower lands exactly on the leader. Fig. 2 shows the migration trace for the follower.

The efficiency for finding a better solution is high in the follower search when f is either monotonic, convex, or concave within a local region. Define the equation for the straight line that passes through $\boldsymbol{x}$ and $\boldsymbol{y}$ as:

$$(z_1 - y_1)/(x_1 - y_1) = (z_2 - y_2)/(x_2 - y_2) = \cdots = (z_n - y_n)/(x_n - y_n) \tag{2}$$

Define Λ as a uniform random variable $\Lambda \sim U(-1,1)$; $\boldsymbol{z} = (z_1, z_2, \ldots, z_n)$ as any point on the straight line; $\mathbb{Z}$ as a search subspace, $\mathbb{Z} = \{\boldsymbol{z} | \boldsymbol{z} = \Lambda \boldsymbol{x} + (1 - \Lambda)\boldsymbol{y}, \Lambda \in [-1, 1]\}$; A and B as events $A = \Lambda \in [0,1], B = \Lambda \in [-1,0]$. The probability that two events happen are $\Pr(A) = \Pr(\Lambda \in [0,1]) = 1/2, \Pr(B) = \Pr(\Lambda \in [-1,0]) = 1/2$.

When f is a monotonic function within $\mathbb{Z}$, since $f(2\boldsymbol{y} - \boldsymbol{x}) \leqslant f(\boldsymbol{y}) \leqslant f(\boldsymbol{x})$, compare $f(\boldsymbol{z})$ to $f(\boldsymbol{x})$:

$$\Pr(f(\boldsymbol{z}) \leqslant f(\boldsymbol{x})) = \Pr(f(\boldsymbol{z}) \leqslant f(\boldsymbol{x})|A)\Pr(A) + \Pr(f(\boldsymbol{z}) \leqslant f(\boldsymbol{x})|B)\Pr(B) = 1 \cdot 1/2 + 1 \cdot 1/2 = 1 \tag{3}$$

Compare $f(\boldsymbol{z})$ to $f(\boldsymbol{y})$:

$$\Pr(f(\boldsymbol{z}) \leqslant f(\boldsymbol{y})) = \Pr(f(\boldsymbol{z}) \leqslant f(\boldsymbol{y})|A)\Pr(A) + \Pr(f(\boldsymbol{z}) \leqslant f(\boldsymbol{y})|B)\Pr(B) = 0 \cdot 1/2 + 1 \cdot 1/2 = 1/2 \tag{4}$$

This implies when f is monotonic, the follower obtains a better solution with probability (*w.p.*) of 1 and outperforms its leader *w.p.* of 0.5. Fig. 3 shows the migration trace for the follower when f is monotonic and $\boldsymbol{z} \in \mathbb{Z}$.

When f is a non-monotonic convex function within $\mathbb{Z}$, compare $f(\boldsymbol{z})$ to $f(\boldsymbol{x})$:

$$\Pr(f(\boldsymbol{z}) \leqslant f(\boldsymbol{x})) = \Pr(f(\boldsymbol{z}) \leqslant f(\boldsymbol{x})|A)\Pr(A) + \Pr(f(\boldsymbol{z}) \leqslant f(\boldsymbol{x})|B)\Pr(B) \geqslant 1 \cdot 1/2 + 0 \cdot 1/2 = 1/2 \tag{5}$$

This implies that the follower obtains a better solution *w.p.* of more than 0.5. Fig. 4 shows the migration trace for the follower when f is non-monotonic convex and $\boldsymbol{z} \in \mathbb{Z}$.

When f is a non-monotonic concave function within $\mathbb{Z}$, compare $f(\boldsymbol{z})$ to $f(\boldsymbol{x})$:

$$\Pr(f(\boldsymbol{z}) \leqslant f(\boldsymbol{x})) = \Pr(f(\boldsymbol{z}) \leqslant f(\boldsymbol{x})|A)\Pr(A) + \Pr(f(\boldsymbol{z}) \leqslant f(\boldsymbol{x})|B)\Pr(B) \geqslant 0 \cdot 1/2 + 1 \cdot 1/2 = 1/2 \tag{6}$$

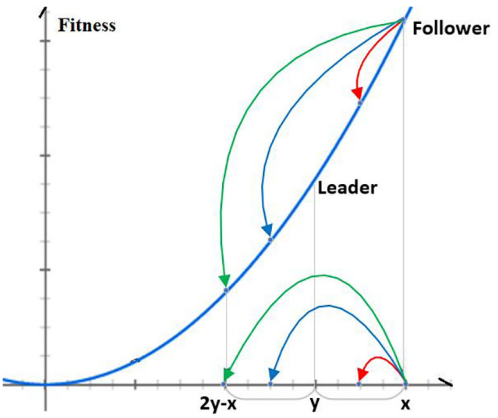

Fig. 2. Follwer search.

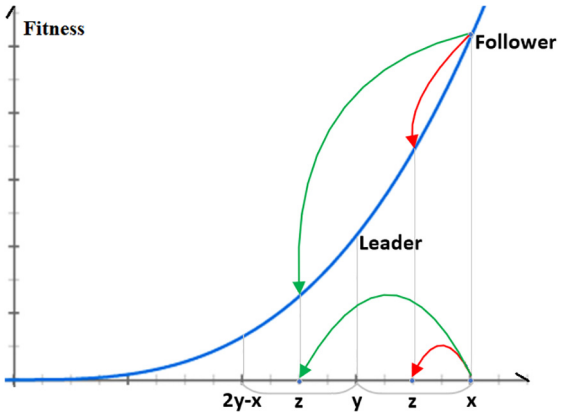

Fig. 3. Follwer search for monotonic function.

Compare $f(\boldsymbol{z})$ to $f(\boldsymbol{y})$:

$$\Pr(f(\boldsymbol{z}) \leqslant f(\boldsymbol{y})) = \Pr(f(\boldsymbol{z}) \leqslant f(\boldsymbol{y})|A)\Pr(A) + \Pr(f(\boldsymbol{z}) \leqslant f(\boldsymbol{y})|B)\Pr(B) = 0 \cdot 1/2 + 1 \cdot 1/2 = 1/2 \tag{7}$$

This implies that the follower obtains a better solution *w.p.* of more than 0.5 and outperforms its leader *w.p.* of 0.5. Fig. 5 shows the migration trace for the follower when f is non-monotonic concave and $\boldsymbol{z} \in \mathbb{Z}$.

When f is a continuous function within $\mathbb{Z}$, if $f(\boldsymbol{y}) < f(\boldsymbol{x})$, $\exists \boldsymbol{z} \in U^{\circ}(\boldsymbol{y}, \boldsymbol{\varepsilon}), \| \boldsymbol{\varepsilon} \| > 0$, such that $f(\boldsymbol{z}) < f(\boldsymbol{x})$. Define $Z = \{\boldsymbol{z} | \boldsymbol{z} \in U^{\circ}(\boldsymbol{y}, \boldsymbol{\varepsilon}) \cap \mathbb{Z}\}$; $\| Z \| = \sup_{\boldsymbol{z}_a, \boldsymbol{z}_b \in Z} \| \boldsymbol{z}_a - \boldsymbol{z}_b \| I(Z \neq \varnothing)$. The probability that $f(\boldsymbol{z}) < f(\boldsymbol{x})$ given $\boldsymbol{z} \in \mathbb{Z}$ satisfies:

$$\Pr(f(\boldsymbol{z}) < f(\boldsymbol{x})) \geqslant \frac{\| Z \|}{2 \| \boldsymbol{y} - \boldsymbol{x} \|} \tag{8}$$

The follower search has a high probability of obtaining a better solution because the follower pursues its leader that has a better fitness value. As the dimension and objective function type varies, IOA offers a follower the option to follow either the best leader, a portion of leaders (great leaders), or all leaders to minimize its fitness value. When the follower decides to follow great leaders, let l be the number of leaders and suppose all leaders are ranked by their fitness values, a random leader with rank index G is selected as the follower's pursuit target, where G is generated from the exponential distribution $G \sim Exp(\log_2 n)$ and clipped by $G \in [1, l]$. By default, a follower chooses each option with equal probability. The user can also customize the likelihood for each option based on the actual circumstances. In practice, when solving non-continuous optimization problems that involve integer programming, IOA converges faster when a higher probability is assigned to the option of following all leaders.

2.3. Leader search

The leader search improves the solutions of search agents that have outstanding fitness values. The leader search process is divided into two procedures: leader central point search (LCPS) and leader coordinate descent search (LCDS). In the first procedure, each leader compares its fitness value to that of the central point of the population. The solution of the central point is determined by $\bar{\boldsymbol{x}} = (\boldsymbol{x}_1 + \boldsymbol{x}_2 + \ldots + \boldsymbol{x}_\Omega)/\Omega$, where Ω is the population size. If the fitness value of the central point is better than that of the leader, the leader approaches the center point equivalently to the follower search according to (1). Otherwise, the leader deviates from the central point according to (9), where $\boldsymbol{y}$ is the solution of the leader, P is a uniform random variable, $P \sim U(0, 1)$. Either way, the leader is assigned a new solution only when $f(\boldsymbol{y} + \Delta\boldsymbol{y}) < f(\boldsymbol{y})$. In practice, LCPS may help leaders avoid bad local minima because approaching the central point voted by all the search agents is essentially a global search process. Fig. 6 shows how the leader deviates from the central point.

$$\Delta\boldsymbol{y} = P(\boldsymbol{y} - \bar{\boldsymbol{x}}) \tag{9}$$

In the second step, a single-round coordinate descent is carried out by each leader. Unlike standard coordinate descent [27] where the axis-wise linear search is carried out several times, the linear search is executed only once for the single-round coordinate descent. Define $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ as the solution of a leader; ξ_i as the alteration factor for the i^{th} axis, $\xi_i \in [-r, r], r > 0$; $\boldsymbol{y}' = (y_1, \ldots, y_i + \xi_i, \ldots, y_n)$ as the new solution after altering y_i. When $f(\boldsymbol{y}') < f(\boldsymbol{y})$, $\boldsymbol{y} \leftarrow \boldsymbol{y}'$, otherwise $\boldsymbol{y}$ remains unchanged. This process is carried out from the 1st scalar to the n^{th} scalar only once. Suppose solutions of leaders are $\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(l)}$, and the fitness value of each leader satisfies $f(\boldsymbol{y}^{(1)}) \leqslant f(\boldsymbol{y}^{(2)}) \leqslant \ldots \leqslant f(\boldsymbol{y}^{(l)})$, the step size r for the coordinate descent is:

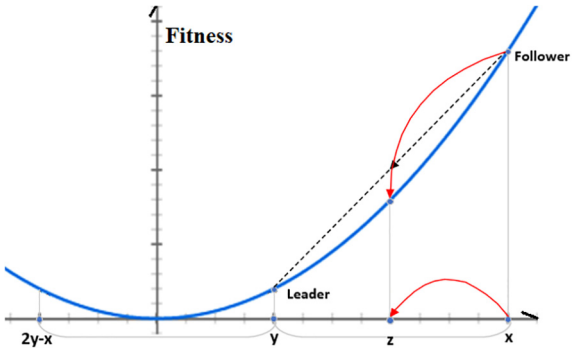

Fig. 4. Follwer search model for non-monotonic convex function.

$$r = \frac{1}{l-1}\sum_{i=2}^{l} \| \boldsymbol{y}^{(i)} - \boldsymbol{y}^{(1)} \| \tag{10}$$

When the algorithm has not yet converged, r is large because leaders tend to be distant from each other, and this allows leaders to search within a large local region. When the algorithm is about to converge, r is small since leaders are close to each other. Thus, the local region shrinks, and leaders can precisely discover better solutions.

Although the single round coordinate descent for a leader has a higher time complexity $O(n)$ compared to the follower search with time complexity $O(1)$, the computation is worthwhile because it is reasonable to precisely improve the current optimal solutions so that followers can follow high-quality leaders. Our high-performance multi-threading implementation for IOA facilitates LCDS by having all leaders perform single-round coordinate descents simultaneously. However, when the dimension n is large, a selection probability can be applied such that coordinate descent operations are partially carried out for some coordinates to save time. Moreover, the maximum number of leaders can be specified according to the computation budget.

2.4. Wanderer search

Global search operations should be performed by search agents to avoid local optima. In the wanderer search, n search agents are randomly selected as movement anchors for the wanderer, where n is the dimension. Define $\boldsymbol{x}$ as the solution of the wanderer; $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n$ as solutions of the anchor search agents; $P_i, i \in [1, n]$ as weight probabilities that satisfy $\sum_{i=1}^{n} P_i = 1$; R as a radius random variable that follows a uniform distribution $R \sim U(0, 1)$. The movement vector $\Delta\boldsymbol{x}$ for the wanderer is:

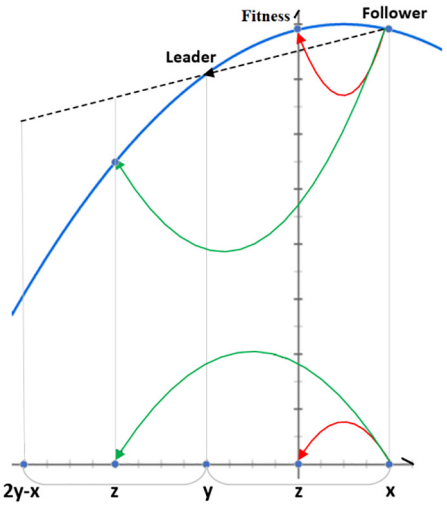

Fig. 5. Follwer search model for non-monotonic concave function.

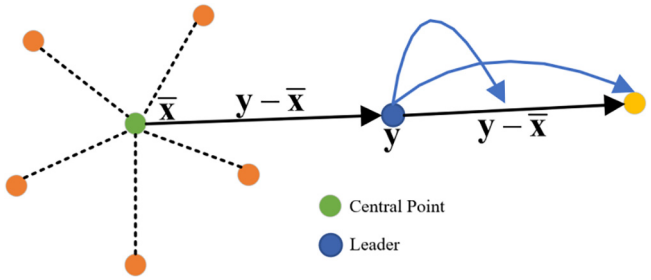

Fig. 6. Leader deviates from the central point.

$$\Delta \boldsymbol{x} = R\sum_{i=1}^{n} P_i(\boldsymbol{a}_i - \boldsymbol{x}) \tag{11}$$

If $f(\boldsymbol{x}+\Delta\boldsymbol{x}) < f(\boldsymbol{x})$, the wanderer moves from $\boldsymbol{x}$ to $\boldsymbol{x}+\Delta\boldsymbol{x}$, otherwise the original solution is retained. Although there is no guarantee that the wanderer will find a better solution in its next movement, the purpose of the wanderer search is to expand the search space. Interestingly, if the global optimum is already enclosed by the wanderer and the anchor search agents, the wanderer search tends to have a faster convergence speed because the next movement pushes the wanderer closer to the global optimum. Fig. 7 shows how the wanderer search works in a 3-dimensional space.

If the wanderer fails to find a better solution, it is given a second chance to improve its fitness value by encircling and deviating from the best search agent. Define ρ as the distance between the wanderer and the best search agent; $\boldsymbol{e}$ as a random unit vector; $\boldsymbol{y}^{(1)}$ as the solution of the best search agent; P as a uniform random variable, $P \sim U(1,3)$. The new solution $\boldsymbol{x}'$ of the wanderer is given by (12). Likewise, $\boldsymbol{x}'$ will become the next solution of the wanderer when $f(\boldsymbol{x}') < f(\boldsymbol{x})$. Fig. 8 depicts the movement of the wanderer.

$$\boldsymbol{x}' = \boldsymbol{y}^{(1)} + P\rho\boldsymbol{e} \tag{12}$$

2.5. Crossover search

In the crossover search, parent search agents are selected according to the spinning wheel selection rule [23] to produce children that have close vicinity to their parents. Define $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ as the solutions of any search agent couple; $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_n)$ as the solution of the child search agent. The crossover process is defined by (13), where P is a uniform random variable, $P \sim U(0,1)$, and I is the indicator function.

$$\gamma_i = a_i I(P \leqslant 0.5) + b_i I(P > 0.5), i \in [1, n] \tag{13}$$

The expected distance between the child and its parents is:

$$\begin{aligned}(E(\|\boldsymbol{\gamma}-\boldsymbol{a}\|))^2 &= E\left(\|\boldsymbol{\gamma}-\boldsymbol{a}\|^2\right) - Var(\|\boldsymbol{\gamma}-\boldsymbol{a}\|) \leqslant E\left(\|\boldsymbol{\gamma}-\boldsymbol{a}\|^2\right) = E\left(\sum_{i=1}^{n}(\gamma_i - a_i)^2\right) = \sum_{i=1}^{n} E(\gamma_i - a_i)^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}(b_i - a_i)^2 = \frac{1}{2}\|\boldsymbol{b}-\boldsymbol{a}\|^2 \Rightarrow E(\|\boldsymbol{\gamma}-\boldsymbol{a}\|) \leqslant \frac{\|\boldsymbol{b}-\boldsymbol{a}\|}{\sqrt{2}} \Rightarrow E(\|\boldsymbol{\gamma}-\boldsymbol{b}\|) \leqslant \frac{\|\boldsymbol{b}-\boldsymbol{a}\|}{\sqrt{2}}\end{aligned} \tag{14}$$

Therefore, the crossover search guarantees that the child search agent is close to its parent. Since two children are produced by a search agent couple, the population size is doubled. To keep the population constant due to the computation budget, all search agents are ranked by their fitness values, then inferior search agents are discarded.

2.6. Role learning

When solving complicated optimization problems, search space exploration should be well conducted to evade local optima. However, this task is poorly performed by some algorithms with a single search agent. In IOA, a search agent can play the role of a leader, follower, or wanderer to expand the search space. Specifically, the local search is carried out by leaders and followers while the global search is performed by wanderers. Exploitation and exploration are balanced by the role learner of each search agent. The local search performed by followers is the exploitation process, while the global search performed by wanderers is the exploration process.

Denote wanderer state as s_1, and follower state as s_2. Let $r_t(s_j|s_i), i,j \in \{1,2\}$ be the reward function at time t given that the search agent switches its role from s_i to s_j. If the initial state is not s_i or the target state is not s_j for the search agent at time t, $r_t = 0$. If the search agent obtains a better solution, $r_t = 1$, otherwise $r_t = -1$. Define $\boldsymbol{C}$ as a cumulative reward matrix with a discounting factor $v \in (0,1]$:

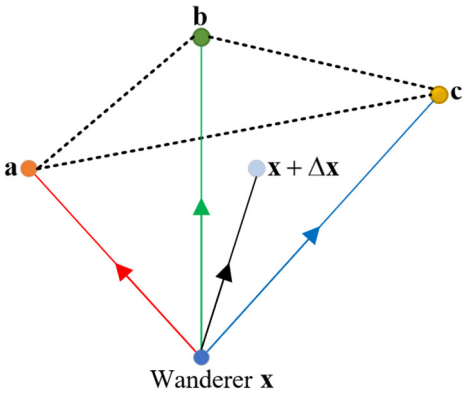

Fig. 7. The wanderer travels within the quadrangular pyramid enclosed by vectors $\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ in a 3-dimensional search space. $\boldsymbol{x}$ is the solution of the wanderer, and $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ are solutions of the three anchor agents.

$$\boldsymbol{C} = \begin{pmatrix} \sum_{t=1}^{T} v^{T-t} r_t(s_1|s_1) & \sum_{t=1}^{T} v^{T-t} r_t(s_2|s_1) \\ \sum_{t=1}^{T} v^{T-t} r_t(s_1|s_2) & \sum_{t=1}^{T} v^{T-t} r_t(s_2|s_2) \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} \tag{15}$$

A Markov transition matrix $\boldsymbol{P}$ proportional to $\boldsymbol{C}$ can be used for managing the role switching task:

$$\boldsymbol{P} = \begin{pmatrix} \Pr(s_1|s_1) & \Pr(s_2|s_1) \\ \Pr(s_1|s_2) & \Pr(s_2|s_2) \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix}, p_{11} = \frac{c_{11}}{c_{11}+c_{12}}, p_{12} = \frac{c_{12}}{c_{11}+c_{12}}, p_{21} = \frac{c_{21}}{c_{21}+c_{22}}, p_{22} = \frac{c_{22}}{c_{21}+c_{22}} \tag{16}$$

Each non-leader search agent decides which role to play based on the past cumulative experience collected from the environment according to $\boldsymbol{P}$. Thus, the local versus global search is handled automatically without user interventions or arduous hyper-parameter tuning.

3. Global optima estimation framework

In this section, a global optima estimation framework is proposed. GOEF provides two conditions that an optimization algorithm should satisfy to obtain the global optima. In IOA, the follower search satisfies both conditions and is used for finding the global optima. The leader search satisfies condition II and it can improve the current optimal solutions. The wanderer search satisfies condition I and it is applied to explore the search space comprehensively. Although the crossover search and role learning are not directly related to these two conditions, they are indispensable complementary components of IOA.

3.1. Global optima conditions

Define f as a n dimensional objective function with a global optimum $\boldsymbol{c}$; $\mathbb{C}$ as the search space, $\boldsymbol{c} \in \mathbb{C}$; $\mathbb{S}_t$ as the search subspace at time t. Since an optimization algorithm should not have any search space blind spot, the relationship between $\mathbb{C}$ and $\mathbb{S}_t$ is:

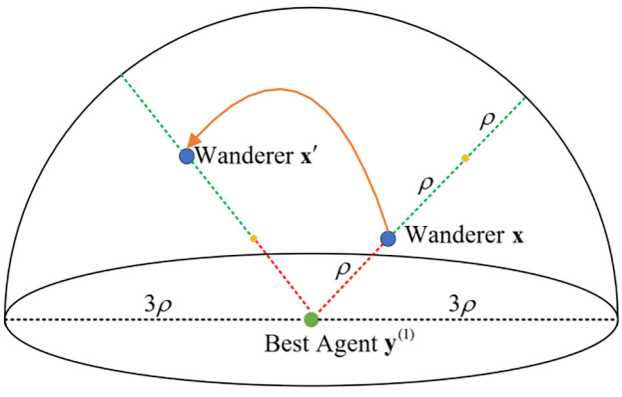

Fig. 8. The wanderer travels inside the sphere. The distance from the wanderer to the best agent is between ρ and 3ρ.

$$\mathbb{C} = \bigcup_{t=1}^{\infty} \mathbb{S}_t \tag{17}$$

The following two lemmas apply:

Lemma 1. If $\forall \boldsymbol{v} \in \mathbb{C}$, $\exists \boldsymbol{x}_t$ such that $\boldsymbol{v} = \boldsymbol{x}_t \in \mathbb{S}_t$, then (17) is satisfied.

Proof of Lemma 1:
Since $\boldsymbol{v} \in \mathbb{C}$, $\boldsymbol{v} = \boldsymbol{x}_t \in \mathbb{S}_t \subset \bigcup_{t=1}^{\infty} \mathbb{S}_t$, $\mathbb{C} \subset \bigcup_{t=1}^{\infty} \mathbb{S}_t$. Notice that $\bigcup_{t=1}^{\infty} \mathbb{S}_t \subset \mathbb{C}$, $\mathbb{C} = \bigcup_{t=1}^{\infty} \mathbb{S}_t$.

Lemma 2. Given that (17) is satisfied, a search algorithm can obtain $\boldsymbol{c}$ by creating a series $\{\boldsymbol{x}_t | \boldsymbol{x}_t \in \mathbb{S}_t\}$ such that $\{f(\boldsymbol{x}_t)\}$ monotonically decreases: $f(\boldsymbol{x}_{t+1}) \leqslant f(\boldsymbol{x}_t)$. And $\boldsymbol{c}$ can be approximated by $\boldsymbol{x}_T$ when T is sufficiently large.

Proof of Lemma 2:
Since $\{f(\boldsymbol{x}_t)\}$ is a monotonically decreasing series, and $f(\boldsymbol{c}) = \inf(f(\boldsymbol{x}_t))$, $\lim_{t\to\infty} f(\boldsymbol{x}_t) = f(\boldsymbol{c})$. Thus, $\forall \varepsilon > 0$, $\exists T$, when $t \geqslant T$, $|f(\boldsymbol{x}_t) - f(\boldsymbol{c})| < \varepsilon$, which is identical to $f(\boldsymbol{c}) \in (f(\boldsymbol{x}_T) - \varepsilon, f(\boldsymbol{x}_T) + \varepsilon)$. Therefore $f(\boldsymbol{c})$ can be approximated by $f(\boldsymbol{c}) \approx f(\boldsymbol{x}_T)$ and the global optimum $\boldsymbol{c}$ can be approximated by $\boldsymbol{c} \approx \boldsymbol{x}_T$. Given the aforementioned time T, there exists a search subspace $\mathbb{S} = \bigcup_{t=1}^{T} \mathbb{S}_t$ such that $\mathbb{C}$ is approximated by $\mathbb{S}$.

Two conditions below should be satisfied before $\boldsymbol{c}$ is accurately estimated.

Condition I: A search agent can travel freely within $\mathbb{C}$ as specified in lemma 1.

Condition II: A monotonically decreasing series $\{f(\boldsymbol{x}_t)\}$ can be obtained.

If condition I is not satisfied, the search algorithm may get trapped in local optima when f is non-convex; if condition II cannot be efficiently satisfied, the search algorithm may converge slowly; if condition I fails but condition II is satisfied, the search algorithm is suitable for solving convex optimization problems; if both conditions are satisfied, the search algorithm can solve non-convex optimization problems and obtain the global optima.

3.2. Estimation for global optima in follower search

Define t as the time index; $\Theta_t = \{\boldsymbol{y}_t^{(1)}, \boldsymbol{y}_t^{(2)}, \ldots, \boldsymbol{y}_t^{(l)}\}$ as solutions of leaders at time t, $f(\boldsymbol{y}_t^{(1)}) \leqslant f(\boldsymbol{y}_t^{(2)}) \leqslant \ldots \leqslant f(\boldsymbol{y}_t^{(l)})$; $\Phi_t = \{\boldsymbol{x}_t^{(1)}, \boldsymbol{x}_t^{(2)}, \ldots, \boldsymbol{x}_t^{(fl)}\}$ as solutions of followers at time t, $f(\boldsymbol{y}_t^{(l)}) \leqslant \inf(f(\Phi_t))$.

Firstly, the relationship between the minimal number of leaders and the dimension of the objective function is analyzed to satisfy condition I. Consider a n dimensional search space $\mathbb{C}$ with l leaders and one follower, each search agent has an associated n dimensional solution vector. For demonstration purposes, assume that the role learning is disabled, and the solutions of leaders remain constant. It should be guaranteed that the follower can travel freely within $\mathbb{C}$ when tracing its leaders. Let the solutions of leaders be $\{\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(l)}\}$; the solution of the follower at time t be $\boldsymbol{x}_t$. The follower's solution can be described as the following expressions given that the follower follows its leaders one after another:

$$\boldsymbol{x}_2 = \boldsymbol{x}_1 + \lambda_1(\boldsymbol{y}^{(1)} - \boldsymbol{x}_1), \boldsymbol{x}_3 = \boldsymbol{x}_2 + \lambda_2(\boldsymbol{y}^{(2)} - \boldsymbol{x}_2), \ldots, \boldsymbol{x}_{l+1} = \boldsymbol{x}_l + \lambda_l(\boldsymbol{y}^{(l)} - \boldsymbol{x}_l) \tag{18}$$

The process can be summarized as:

$$\boldsymbol{x}_{l+1} = \boldsymbol{x}_1 + \sum_{i=1}^{l} \mu_i(\boldsymbol{y}^{(i)} - \boldsymbol{x}_1), \mu_i = \lambda_i \prod_{j=i+1}^{l} 1 - \lambda_j \tag{19}$$

Define the direction vector as $\boldsymbol{d}^{(i)} = \boldsymbol{y}^{(i)} - \boldsymbol{x}_1$, the solution of the follower after tracing all leaders is:

$$\boldsymbol{x}_{l+1} = \boldsymbol{x}_1 + \sum_{i=1}^{l} \mu_i \boldsymbol{d}^{(i)} \tag{20}$$

To let the follower travel freely within the n dimensional search space $\mathbb{C}$, n linearly independent direction vectors $\{\boldsymbol{d}^{(1)}, \boldsymbol{d}^{(2)}, \ldots, \boldsymbol{d}^{(n)}\}$ are required to form the basis of $\mathbb{C}$ by which every vector in $\mathbb{C}$ can be expressed. Therefore, at least n leaders should be elected. When many followers are used and the number of leaders is no less than the dimension, condition I will be satisfied more easily.

Secondly, we show that the monotonically decreasing series $\{f(\boldsymbol{x}_t)\}$ is attenable and condition II can be satisfied. Suppose a follower with solution $\boldsymbol{x}_t \in \Phi_t$ is randomly assigned a leader with solution $\boldsymbol{y}_t \in \Theta_t$ at time t. If $f(\boldsymbol{y}_t) < f(\boldsymbol{x}_t)$, then $\exists p_t \in [0, 1]$ such that $f(\boldsymbol{x}_{t+1}) < f(\boldsymbol{x}_t), \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + 2p_t(\boldsymbol{y}_t - \boldsymbol{x}_t)$. This inequality holds and the monotonically decreasing series $\{f(\boldsymbol{x}_t)\}$ exists regardless of the objective function properties because when $p_t = 0.5, f(\boldsymbol{x}_{t+1}) = f(\boldsymbol{y}_t) < f(\boldsymbol{x}_t)$. In more general cases, $\boldsymbol{x}_{t+1}$ can be found via a simple trial-and-error approach: firstly, p_t is generated from $[0, 1]$, if $f(\boldsymbol{x}_{t+1}) \geqslant f(\boldsymbol{x}_t)$ then p_t is discarded and generated again until $f(\boldsymbol{x}_{t+1}) < f(\boldsymbol{x}_t)$. Although $f(\boldsymbol{x}_{t+1}) < f(\boldsymbol{y}_t) < f(\boldsymbol{x}_t)$ is appreciated, the condition is not mandatory because even if $f(\boldsymbol{y}_t) < f(\boldsymbol{x}_{t+1}) < f(\boldsymbol{x}_t)$, the follower is one step closer to its leader and a better solution is obtained.

Since both conditions in GOEF are satisfied, the global optimum $\boldsymbol{c}$ can be accurately approximated by the last solution $\boldsymbol{x}_T$ of the follower if T is sufficiently large. Generally, the follower-leader system has many search agents. To obtain a more accurate optimization result, the solution of the best leader at time T $\boldsymbol{y}_T^{(1)}$ can be used as an estimation for $\boldsymbol{c}$.

3.3. Solution improvement in leader search

LCPS significantly improves the solution qualities of the leaders before LCDS. Define $\boldsymbol{x}_t^{(i)}$, $i \in [1, \Omega]$, as the solution of the search agent i at time t; $\boldsymbol{y}_t^{(j)}$, $j \in [1, l]$, as the solution of the leader j at time t. Assume $\boldsymbol{x}_t^{(i)}$ converges in probability to $\boldsymbol{\mu}$, *i.e.,* $\forall \varepsilon > 0$, $\lim_{t\to\infty} \Pr\left(\| \boldsymbol{x}_t^{(i)} - \boldsymbol{\mu} \| < \varepsilon \right) = 1$, where $\boldsymbol{\mu}$ is either a local or global optimum, $f(\boldsymbol{\mu}) = \inf_{i\in[1,\Omega]} f(\boldsymbol{x}_t^{(i)})$. If there exists at least two $\boldsymbol{x}_t^{(i_1)}$, $\boldsymbol{x}_t^{(i_2)}$, $i_1 \neq i_2$, such that $(\boldsymbol{\mu} - \boldsymbol{x}_t^{(i_1)}) \cdot (\boldsymbol{\mu} - \boldsymbol{x}_t^{(i_2)}) < 0$, then $\boldsymbol{\mu}$ is said to be besieged by $\boldsymbol{x}_t^{(1)}, \boldsymbol{x}_t^{(2)}, \ldots, \boldsymbol{x}_t^{(\Omega)}$. Let $\bar{\boldsymbol{x}}_t = (\boldsymbol{x}_t^{(1)} + \boldsymbol{x}_t^{(2)} + \cdots + \boldsymbol{x}_t^{(\Omega)})/\Omega$ be an approximation of $\boldsymbol{\mu}$. Since $f(\boldsymbol{\mu}) \leqslant f(\bar{\boldsymbol{x}}_t)$, if $f(\bar{\boldsymbol{x}}_t) \leqslant f(\boldsymbol{y}_{t+k}^{(j)})$, $k \geqslant 1$, then condition II is satisfied because a monotonically decreasing series is obtained: $f(\boldsymbol{\mu}) \leqslant f(\bar{\boldsymbol{x}}_t) \leqslant f(\boldsymbol{y}_{t+k}^{(j)}) \leqslant f(\boldsymbol{y}_{t+k-1}^{(j)}) \leqslant \cdots \leqslant f(\boldsymbol{y}_t^{(j)})$.

Define $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ as the solution of a leader after LCPS. A single round coordinate descent alters each scalar y_i and obtains an updated scalar y_i' for $i \in [1, n]$. Thus n solutions $\boldsymbol{y}_i' = (y_1', \ldots, y_i', \ldots, y_n), i \in [1, n]$ are generated. If $f(\boldsymbol{y}_i') \geqslant f(\boldsymbol{y}_{i-1}')$, then $y_i' \leftarrow y_i$, otherwise the altered scalar y_i' is kept. Thus, a monotonically decreasing series $\{f(\boldsymbol{y}_i')\}$ is obtained because $f(\boldsymbol{y}_{i+1}') \leqslant f(\boldsymbol{y}_i')$. Therefore, the leader search satisfies condition II and improves the solution in a step-by-step approach.

3.4. Search space expansion in wanderer search

Define $\boldsymbol{x}_t$ as the solution of a wanderer at time t; $\boldsymbol{a}_{ti}, i \in [1, n]$ as solutions of randomly selected anchor search agents. Suppose $\boldsymbol{a}_{ti} - \boldsymbol{x}_t, i \in [1, n]$ are linearly independent, since the dimension of $\mathbb{C}$ is n, $\boldsymbol{a}_{ti} - \boldsymbol{x}_t$ forms the basis of $\mathbb{C}$, hence any vector in $\mathbb{C}$ can be expressed as:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \beta \sum_{i=1}^{n} \alpha_i (\boldsymbol{a}_{ti} - \boldsymbol{x}_t) \tag{21}$$

Therefore, condition I is satisfied, and the wanderer can travel freely within $\mathbb{C}$ when α_i and β are unconstrained. Replace α_i, β by P_i, R in (11), the updated equation guarantees that every vector in the subspace enclosed by $\boldsymbol{x}_t$ and $\boldsymbol{a}_{ti}, i \in [1, n]$ can be expressed by $\boldsymbol{x}_{t+1}$.

If a better solution is not found, the wanderer encircles and deviates from the best search agent and obtains the next solution according to (22), where $\boldsymbol{\varepsilon}_{ti}, i \in [1, n]$, $\boldsymbol{\varepsilon}_{t1} = (1, 0, \ldots, 0)^T$, $\boldsymbol{\varepsilon}_{tn} = (0, 0, \ldots, 1)^T$ are linearly independent vectors that form the basis of the n dimensional $\mathbb{C}$.

$$\boldsymbol{x}_{t+1} = \boldsymbol{y}_t^{(1)} + \beta\rho\boldsymbol{e}_t = \boldsymbol{y}_t^{(1)} + \beta\rho \sum_{i=1}^{n} \alpha_i \boldsymbol{\varepsilon}_{ti} \tag{22}$$

Therefore, condition I is satisfied because any vector in $\mathbb{C}$ can be expressed by $\boldsymbol{x}_{t+1}$ when α_i and β are unconstrained. Replace β by P in (12) and ensure that $\sum_{i=1}^{n} \alpha_i = 1$, the updated equation guarantees that every point inside the sphere whose distance to the center is between ρ and 3ρ can be expressed by $\boldsymbol{x}_{t+1}$.

4. DIOA for neural network training

Each sub-algorithm in IOA can be enabled or disabled when tackling different kinds of optimization problems. DIOA is a revised version of IOA in which the wanderer search, role learning, and crossover search are disabled. Since neural networks are differentiable, LCDS is replaced by the leader gradient descent search (LGDS). DIOA is thus proposed to train large-scale neural networks efficiently. Mathematical principles for the follower search are explained. Techniques for efficient neural network weight tensors access are illustrated.

4.1. Differentiable integrated optimization algorithm

IOA requires modifications before it can train neural networks efficiently. To clarify illustrations, a sub-algorithm is defined as one or a series of procedures. The original IOA consists of the following procedures in chronological order: LCPS, LCDS, role learning search (RLS) that combines follower search (FS) and wanderer search (WS) procedures with the help of role learning, and crossover search (CS). In DIOA, the chronological procedures are simplified as LCPS, LGDS, and FS. LCDS is replaced by LGDS for two reasons. First, the objective function of a neural network is differentiable almost everywhere. Second, LCDS is computationally impractical for neural network training. If LCDS is still enabled, the forward propagation execution times should equal the dimension, *i.e.,* the number of trainable parameters. Since neural network training is a heavyweight task, the number of search agents is limited and far less than the dimension. As a result, the precondition of the wanderer search is violated. Therefore, the wanderer search and role learning are disabled. To further speed up computations, the crossover search is canceled because it is more suitable for non-continuous optimization, *e.g.,* integer programming. Fig. 9 compares the original IOA to DIOA.

4.2. Quasi-gradient descent in follower search

We demonstrate that the follower search is a quasi-gradient descent that fully utilizes the gradient information obtained from LGDS. Let f be a continuous and differentiable neural network objective function or cost function, $\boldsymbol{y}$ be the solution of a leader, and $\boldsymbol{x}$ be the solution of a follower. According to the first-order Taylor's expansion:

$$f(\boldsymbol{y}+\Delta\boldsymbol{y}) = f(\boldsymbol{y}) + \nabla f(\boldsymbol{y})\cdot\Delta\boldsymbol{y} + o(||\Delta\boldsymbol{y}||) \tag{23}$$

To minimize $f(\boldsymbol{y}+\Delta\boldsymbol{y})$, $\nabla f(\boldsymbol{y})\cdot\Delta\boldsymbol{y}$ needs to be minimized:

$$\nabla f(\boldsymbol{y})\cdot\Delta\boldsymbol{y} = \parallel\nabla f(\boldsymbol{y})\parallel\parallel\Delta\boldsymbol{y}\parallel\cos\langle\nabla f(\boldsymbol{y}),\Delta\boldsymbol{y}\rangle \geqslant -\parallel\nabla f(\boldsymbol{y})\parallel\parallel\Delta\boldsymbol{y}\parallel \tag{24}$$

The equality sign above is obtained when $\Delta\boldsymbol{y} = -\alpha\nabla f(\boldsymbol{y})$, $\alpha > 0$, where α is the learning rate. Let the next solution for the leader be $\boldsymbol{y}' = \boldsymbol{y} - \alpha\nabla f(\boldsymbol{y})$. When an appropriate α is given, $f(\boldsymbol{y}') \leqslant f(\boldsymbol{y})$. Although gradient descent is effective for minimizing convex differentiable functions, it has two limitations. First, the backpropagation process is time-consuming when training large-scale neural networks. Second, gradient descent may obtain bad local optima because f is not necessarily convex. Due to the first limitation, only a few leaders can be used. Thus, more followers should be utilized to boost the performance of DIOA. Since the follower tightly follows the leader, let $\nabla f(\boldsymbol{x}) = \nabla f(\boldsymbol{y}) + \boldsymbol{\delta}$ and $\Delta\boldsymbol{x}\parallel\Delta\boldsymbol{x}\parallel^{-1} = \Delta\boldsymbol{y}\parallel\Delta\boldsymbol{y}\parallel^{-1} + \boldsymbol{\varepsilon}$, where $\parallel\boldsymbol{\delta}\parallel$ and $\parallel\boldsymbol{\varepsilon}\parallel$ are assumed to be small. The first-order term of $f(\boldsymbol{x}+\Delta\boldsymbol{x})$ is:

$$\begin{aligned}
&\nabla f(\boldsymbol{x})\cdot\Delta\boldsymbol{x} = \parallel\Delta\boldsymbol{x}\parallel(\nabla f(\boldsymbol{y})+\boldsymbol{\delta})\cdot\left(\Delta\boldsymbol{y}\parallel\Delta\boldsymbol{y}\parallel^{-1}+\boldsymbol{\varepsilon}\right)\\
&\parallel\Delta\boldsymbol{x}\parallel^{-1}\nabla f(\boldsymbol{x})\cdot\Delta\boldsymbol{x} = \parallel\Delta\boldsymbol{y}\parallel^{-1}\nabla f(\boldsymbol{y})\cdot\Delta\boldsymbol{y} + \nabla f(\boldsymbol{y})\cdot\boldsymbol{\varepsilon} + \parallel\Delta\boldsymbol{y}\parallel^{-1}\boldsymbol{\delta}\cdot\Delta\boldsymbol{y} + \boldsymbol{\delta}\cdot\boldsymbol{\varepsilon}\\
&= \nabla f(\boldsymbol{y})\cdot\boldsymbol{\varepsilon} + \parallel\Delta\boldsymbol{y}\parallel^{-1}\boldsymbol{\delta}\cdot\Delta\boldsymbol{y} + \boldsymbol{\delta}\cdot\boldsymbol{\varepsilon} - \parallel\nabla f(\boldsymbol{y})\parallel \leqslant \parallel\nabla f(\boldsymbol{y})\parallel\parallel\boldsymbol{\varepsilon}\parallel + \parallel\boldsymbol{\delta}\parallel + \parallel\boldsymbol{\delta}\parallel\parallel\boldsymbol{\varepsilon}\parallel - \parallel\nabla f(\boldsymbol{y})\parallel \leqslant M - \parallel\nabla f(\boldsymbol{y})\parallel
\end{aligned} \tag{25}$$

If $M \leqslant \parallel\nabla f(\boldsymbol{y})\parallel$, then $\nabla f(\boldsymbol{x})\cdot\Delta\boldsymbol{x} \leqslant 0$, and $f(\boldsymbol{x}+\Delta\boldsymbol{x}) \leqslant f(\boldsymbol{x})$ if the influence of $o(||\Delta\boldsymbol{x}||)$ is negligible. According to (1), P serves as an effective learning rate in the follower search:

$$\nabla f(\boldsymbol{x})\cdot\Delta\boldsymbol{x} = 2P\nabla f(\boldsymbol{x})\cdot(\boldsymbol{y}-\boldsymbol{x}) \in [2\nabla f(\boldsymbol{x})\cdot(\boldsymbol{y}-\boldsymbol{x}), 0] \tag{26}$$

Specifically, the lower and upper bounds of $\nabla f(\boldsymbol{x})\cdot\Delta\boldsymbol{x}$ are obtained when $P = 1$ and $P = 0$. Since different followers have different P, the second limitation is compensated because as the number of followers increases the search space is more thoroughly explored. Fig. 10 explains how the quasi-gradient descent works for the follower.

4.3. Efficient access to trainable parameter tensors

Unlike many optimization problems whose solutions are vectors, trainable parameters of a neural network are stored in high-dimensional tensors, *e.g.*, 4D weight filters in CNN, weight matrices in fully connected neural networks (FC). To obtain a vector-form solution in LCPS and the follower search under DIOA, tensors can be copied and flattened into multiple vectors which are then concatenated. However, this approach creates significant runtime overhead. An addressing technique can avoid data copying and boost the algorithm's efficiency. Concretely, tensors are not flattened into a concatenated vector that contains real numbers. Instead, scalar addresses of all tensors are flattened into a concatenated address vector. In low-level programming languages such as C/C++, addresses are pointers. In higher-level programming languages such as Java that disable pointer access, the addresses are the scalar offsets in tensors. Fig. 11 shows how trainable parameters are accessed.

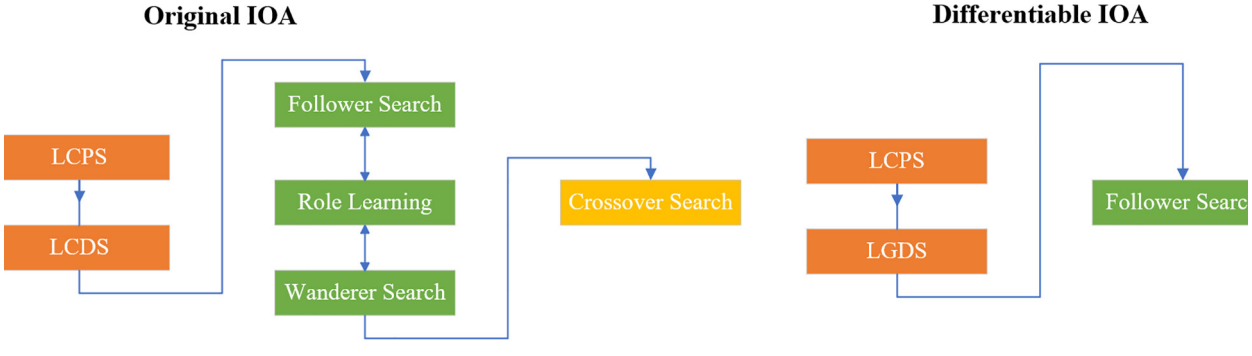

Fig. 9. Procedures for IOA and DIOA.

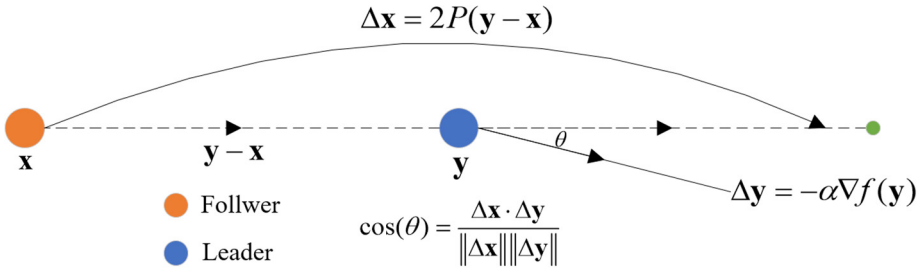

Fig. 10. Quasi-gradient descent in the follower search.

5. Experiments on benchmark functions

IOA is compared with the other 8 popular and highly-cited algorithms: HHO (2019) [7], BES (2020) [1], SSA (2017) [20], NAA (2016) [18,19], GWO (2014) [21], BA (2010) [28], DE (2006) [4,24], and PSO (1995) [15]. HHO and BES are state-of-the-art algorithms published in recent years; GWO and NAA can find global optima efficiently. GWO has also been applied to solve unit commitment problems [13,22]. All 9 algorithms are implemented in Java 11 and accelerated by CPU multi-threading. The benchmark function experiments are performed on a workstation with an AMD Ryzen 7 3700X CPU, an NVIDIA RTX 2070 GPU, and 16 GB of DDR4 memory. The 9 algorithms are compared to each other based on 27 benchmark functions [9,10]. 5 benchmark function experiment results are provided in Table 1–5. The other 22 experiment results are archived in the supplementary materials. Most benchmark functions are chosen from CEC 2020, 2017, 2014, and 2005 [2,17,25,30], six cases are performed for each benchmark function, each case differs in dimension D and search agent population Ω. m independent experiments are carried out so that the results are more stable and reliable. We measure the performance of each algorithm using seven indicators: mean fitness, best fitness, worst fitness, fitness *std.*, mean iterations, championships, and milestone fitness values.

The **fitness** values for each algorithm given D and Ω are obtained from $m = 30$ independent experiments. Define $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$ as the fitness value vector; $\mu(\boldsymbol{v}) = (v_1 + v_2 + \ldots + v_m)/m$ as the **mean fitness** value; $\min(v_1, v_2, \ldots, v_m)$ as the **best fitness** value, $\max(v_1, v_2, \ldots, v_m)$ as the **worst fitness** value; $\left(\boldsymbol{v}^T \cdot \boldsymbol{v}/m - \mu(\boldsymbol{v})^2\right)^{1/2}$ as the standard deviation of the fitness values (**fitness *std.***) which measures the stability of an optimization algorithm. The stability increases as fitness *std.* decreases.

To measure how fast each algorithm converges as the fitness value decreases, we define a series of **milestone fitness values** that reside within the range between the smallest fitness value and the initial fitness value. For instance, the milestone fitness values for $y = \sin(x) + 1$ can be 0.5, 1, 1.5 given that the range of y is $[0, 2]$. Each benchmark function differs in milestone fitness values because each function has a different range.

The **championship** is a counter that measures the number of times the fitness value of an algorithm **firstly** drops below each milestone fitness value (we use "an algorithm attain a milestone" in short for "the fitness value of an algorithm drops below a milestone fitness value"). In other words, championship depicts the frequency that each algorithm beats the other algorithms based on many experiments. One milestone fitness value can have more than one champion if multiple algorithms attain the milestone simultaneously. Suppose 100 independent experiments are carried out, the milestone fitness values are (9, 5, 1), the championships for IOA, NAA, and DE are (50, 60, 70), (30, 20, 20), (20, 20, 10) respectively. Then IOA firstly attains milestone 9 in 50 out of 100 experiments, milestone 5 in 60 out of 100 experiments, and milestone 1 in 70 out of 100 experiments.

The **mean iterations** represent the average number of iterations required for an algorithm to attain each milestone. That is, the mean iteration measures how fast an algorithm converges based on many experiments. For example, suppose the

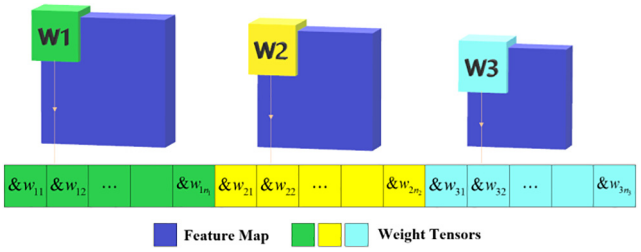

Fig. 11. Efficient access to trainable parameter tensors in CNN. $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{W}_3$ are trainable weight tensors. & is the address symbol of a weight scalar. n_1, n_2, n_3 are the number of weight scalars in $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{W}_3$.

Table 1
Performance comparison for shifted Rosenbrock function f_1

f_2	D	Ω	Mean Fitness	Best Fitness	Worst Fitness	Fitness Std.	Iterations	Championships	Milestones Values
IOA	10	50	**3.144**	**0.135**	17.549	4.647	2,2,24	24,30,25	8.969e8,4.484e8,8.567
		100	**2.461**	**2.578e-4**	29.213	5.761	2,2,42	26,30,22	5.550e8,2.775e8,1.471
	50	250	**85.746**	**39.627**	141.821	37.721	2,2,32	29,30,11	1.821e10,9.107e9,43
		500	**55.634**	**36.036**	173.583	33.935	2,3,38	29,30,26	1.171e10,5.855e9,106
	100	500	**150.104**	**90.385**	401.355	69.523	1,1,27	30,30,15	3.577e11,1.788e11,155
		1000	**152.818**	**88.607**	542.296	96.540	1,1,34	30,30,11	1.727e11,8.636e10,92
HHO	10	50	2184478.241	377374.812	2,963,024	706750.849	3,3,-	11,23,0	8.969e8,4.484e8,8.567
		100	1803784.520	658287.375	2864406.5	612750.590	3,3,-	8,25,0	5.550e8,2.775e8,1.471
	50	250	1.671e7	1.366e7	1.794e7	1111983.170	3,3,-	27,30,0	1.821e10,9.107e9,43
		500	1.589e7	1.280e7	1.763e7	1182585.711	3,3,-	26,30,0	1.171e10,5.855e9,106
	100	500	3.526e7	3.316e7	3.658e7	804653.584	2,2,-	18,30,0	3.577e11,1.788e11,155
		1000	3.416e7	3.113e7	3.611e7	1291405.782	2,2,-	13,30,0	1.727e11,8.636e10,92
BES	10	50	3.262e8	377524.625	2.055e9	4.614e8	14,21,-	10,8,0	8.969e8,4.484e8,8.567
		100	7935836.003	6110.519	1.038e8	1.968e7	3,5,-	10,10,0	5.550e8,2.775e8,1.471
	50	250	1.940e10	9.598e9	2.779e10	4.796e9	7,-,-	1,0,0	1.821e10,9.107e9,43
		500	9.661e9	2.349e9	2.258e10	4.831e9	9,-,-	2,0,0	1.171e10,5.855e9,106
	100	500	4.768e10	3.544e10	5.707e10	5.539e9	1,1,-	30,30,0	3.577e11,1.788e11,155
		1000	3.184e10	1.765e10	4.549e10	6.829e9	1,1,-	30,30,0	1.727e11,8.636e10,92
SSA	10	50	7.986e8	2.011e7	3.422e9	8.156e8	17,17,-	2,0,0	8.969e8,4.484e8,8.567
		100	4.316e8	1.225e7	1.418e9	3.784e8	15,15,-	2,1,0	5.550e8,2.775e8,1.471
	50	250	1.538e10	1.052e10	2.508e10	2.950e9	18,-,-	0,0,0	1.821e10,9.107e9,43
		500	1.143e10	7.655e9	1.518e10	1.664e9	22,-,-	0,0,0	1.171e10,5.855e9,106
	100	500	4.155e10	3.251e10	5.052e10	4.372e9	1,2,-	30,30,0	3.577e11,1.788e11,155
		1000	3.554e10	2.804e10	4.119e10	3.041e9	2,3,-	20,13,0	1.727e11,8.636e10,92
NAA	10	50	2589.786	491.962	18727.294	3344.209	13,15,-	0,0,0	8.969e8,4.484e8,8.567
		100	736.193	142.555	4406.901	901.462	10,13,-	1,0,0	5.550e8,2.775e8,1.471
	50	250	1.330e10	3.190e9	5.049e10	8.670e9	68,-,-	0,0,0	1.821e10,9.107e9,43
		500	3.419e9	9.735e8	8.616e9	1.608e9	73,80,-	0,0,0	1.171e10,5.855e9,106
	100	500	2.435e11	9.963e10	4.220e11	9.714e10	56,-,-	0,0,0	3.577e11,1.788e11,155
		1000	1.270e11	3.560e10	3.502e11	6.538e10	63,-,-	0,0,0	1.727e11,8.636e10,92
GWO	10	50	448526.724	1672.812	708257.25	202761.314	5,7,-	0,1,0	8.969e8,4.484e8,8.567
		100	300832.246	44.791	563705.625	215825.106	4,5,-	1,1,0	5.550e8,2.775e8,1.471
	50	250	8.534e9	5.578e9	1.175e10	1.639e9	23,33,-	0,0,0	1.821e10,9.107e9,43
		500	6.107e9	4.179e9	9.417e9	1.598e9	30,39,-	0,0,0	1.171e10,5.855e9,106
	100	500	3.870e10	3.335e10	4.355e10	2.269e9	13,18,-	0,0,0	3.577e11,1.788e11,155
		1000	3.300e10	2.750e10	3.904e10	3.024e9	16,22,-	0,0,0	1.727e11,8.636e10,92
BA	10	50	8.053e8	2109574.5	2.299e9	5.621e8	20,-,-	1,0,0	8.969e8,4.484e8,8.567
		100	2.707e8	1.869e7	8.873e8	1.963e8	20,22,-	0,3,0	5.550e8,2.775e8,1.471
	50	250	1.753e10	7.852e9	3.249e10	5.888e9	42,-,-	0,0,0	1.821e10,9.107e9,43
		500	1.236e10	5.443e9	2.137e10	4.013e9	35,-,-	0,0,0	1.171e10,5.855e9,106
	100	500	4.833e10	3.213e10	6.756e10	1.024e10	3,7,-	0,0,0	3.577e11,1.788e11,155
		1000	3.511e10	2.375e10	5.348e10	6.850e9	5,26,-	0,0,0	1.727e11,8.636e10,92
DE	10	50	5960.663	223.049	77965.687	13662.781	14,17,-	0,0,0	8.969e8,4.484e8,8.567
		100	2350.525	282.884	7176.911	1734.159	13,17,-	0,0,0	5.550e8,2.775e8,1.471
	50	250	7.443e9	3.753e9	1.032e10	1.754e9	74,74,-	0,0,0	1.821e10,9.107e9,43
		500	6.319e9	2.580e9	8.686e9	1.352e9	82,-,-	0,0,0	1.171e10,5.855e9,106
	100	500	1.125e11	7.820e10	1.452e11	1.471e10	36,73,-	0,0,0	3.577e11,1.788e11,155
		1000	1.050e11	7.743e10	1.254e11	1.288e10	72,-,-	0,0,0	1.727e11,8.636e10,92
PSO	10	50	2.246e8	3,187,706	9.339e8	2.547e8	7,8,-	0,0,0	8.969e8,4.484e8,8.567
		100	7.610e7	96442.898	6.045e8	1.262e8	7,8,-	0,0,0	5.550e8,2.775e8,1.471
	50	250	9.697e9	4.479e9	1.360e10	2.006e9	10,-,-	0,0,0	1.821e10,9.107e9,43
		500	7.145e9	3.482e9	1.049e10	1.646e9	11,-,-	0,0,0	1.171e10,5.855e9,106
	100	500	3.677e10	2.930e10	4.990e10	4.767e9	4,5,-	0,0,0	3.577e11,1.788e11,155
		1000	2.939e10	2.466e10	3.761e10	3.062e9	5,7,-	0,0,0	1.727e11,8.636e10,92

Iterations = 100, Experiments = 30, Bold: the best case.
$f_1(\boldsymbol{y}) = \sum_{i=1}^{n-1} 100(y_i^2 - y_{i+1})^2 + (y_i - 1)^2, \boldsymbol{y} = \boldsymbol{x} - \boldsymbol{a}, s.t. |x_i| \leqslant 128$
$a_i = \omega_{(i-1) \bmod 5+1}, \{\omega_m\} = \{40, 45, 50, 55, 60\}$
$\min(f_1) = f_1(\mathbf{1}) = 0$

milestone fitness values are (14, 9, 4), then (3, 5, 9) means the algorithm requires 3, 5, and 9 iterations on average to attain milestones 14, 9, and 4. Note that if the number of failures to attain a milestone is more than 90% of the number of experiments, the mean iteration is denoted as "-".

In **IOA**, the number of leaders equals the dimension, the discounting factor for the role learner is 0.9. IOA favors the convention over configuration software engineering paradigm and encourages users to use default parameter settings unless the computation budget exists or IOA is used for solving special problems such as high-dimensional integer programming. In

Table 2
Performance comparison for shifted Rastrigin function f_2

f_3	D	Ω	Mean Fitness	Best Fitness	Worst Fitness	Fitness Std.	Iterations	Championships	Milestones Values
IOA	10	50	**0.265**	**0**	0.994	0.439	2,3,32	28,30,22	7773,3886,0
		100	**0.198**	**0**	2.984	0.596	2,2,35	25,30,26	7640,3820,0
	50	250	**0**	**0**	0	0	2,3,89	28,29,30	74945,37472,1.131e-6
		500	**0**	**0**	0	0	2,3,87	28,28,30	57818,28909,0
	100	500	**0.591**	**0.110**	1.335	0.314	2,3,89	22,17,30	200854,100433,13
		1000	**0.405**	**0.030**	0.812	0.208	2,3,92	20,19,30	178814,89410,6.664
HHO	10	50	521.217	369.827	593.052	55.984	3,3,-	14,21,0	7773,3886,0
		100	455.603	152.159	584.594	88.535	2,3,-	6,28,0	7640,3820,0
	50	250	2568.334	2465.135	2773.313	94.039	3,3,-	24,29,0	74945,37472,1.131e-6
		500	2543.599	2176.750	2824.967	113.045	2,3,-	27,30,0	57818,28909,0
	100	500	5068.923	4996.363	5681.476	181.562	2,3,-	15,30,0	200854,100433,13
		1000	5072.097	4992.772	5706.227	180.604	2,2,-	25,30,0	178814,89410,6.664
BES	10	50	5655.503	1143.623	12059.113	2847.340	4,-,-	6,1,0	7773,3886,0
		100	2301.055	396.508	6136.916	1633.718	2,6,-	20,9,0	7640,3820,0
	50	250	69,613	29519.433	102059.867	14956.636	3,-,-	7,0,0	74945,37472,1.131e-6
		500	47426.388	28185.572	68868.257	9717.054	6,-,-	8,0,0	57818,28909,0
	100	500	161694.777	127934.203	191481.843	16801.522	3,-,-	14,0,0	200854,100433,13
		1000	133073.780	98497.75	176258.859	19237.870	5,-,-	19,0,0	178814,89410,6.664
SSA	10	50	5370.203	1813.341	10657.190	2427.654	16,-,-	2,0,0	7773,3886,0
		100	5317.882	2606.862	9893.380	2206.576	14,19,-	6,1,0	7640,3820,0
	50	250	68977.492	53447.519	78838.906	6704.623	19,-,-	2,0,0	74945,37472,1.131e-6
		500	56693.627	43241.359	69381.734	6458.322	24,-,-	0,0,0	57818,28909,0
	100	500	159690.727	141821.640	187163.921	10406.483	5,-,-	2,0,0	200854,100433,13
		1000	143536.850	122410.429	165120.984	9784.321	4,-,-	3,0,0	178814,89410,6.664
NAA	10	50	44.155	32.383	53.831	6.507	12,19,-	0,0,0	7773,3886,0
		100	36.384	26.518	44.685	4.412	8,14,-	1,0,0	7640,3820,0
	50	250	20115.659	9780.794	35645.687	5872.529	50,77,-	0,0,0	74945,37472,1.131e-6
		500	9585.499	5392.495	17030.332	3240.844	46,67,-	0,0,0	57818,28909,0
	100	500	165466.572	84237.804	242510.468	34708.233	69,-,-	0,0,0	200854,100433,13
		1000	103055.755	68843.835	156573.187	20258.550	72,-,-	0,0,0	178814,89410,6.664
GWO	10	50	423.261	42.180	3754.553	859.651	5,13,-	5,1,0	7773,3886,0
		100	299.036	28.209	2091.563	638.379	4,7,-	0,0,0	7640,3820,0
	50	250	37104.814	29095.458	45074.519	4281.994	18,41,-	0,0,0	74945,37472,1.131e-6
		500	29997.487	19826.498	47479.156	5753.650	24,37,-	0,0,0	57818,28909,0
	100	500	135298.243	112782.179	154466.812	10835.380	27,-,-	0,0,0	200854,100433,13
		1000	122985.796	104515.390	141934.093	9119.627	28,-,-	0,0,0	178814,89410,6.664
BA	10	50	8114.491	3194.006	20249.273	3760.643	18,-,-	0,0,0	7773,3886,0
		100	5574.997	1933.701	11893.352	2468.242	15,-,-	0,0,0	7640,3820,0
	50	250	63586.347	50978.574	89186.437	9310.664	43,-,-	0,0,0	74945,37472,1.131e-6
		500	55208.371	40744.390	71086.148	7598.763	38,-,-	0,0,0	57818,28909,0
	100	500	142529.207	111542.203	162864.328	14381.620	32,-,-	0,0,0	200854,100433,13
		1000	122272.893	91674.460	149812.671	12025.152	32,-,-	0,0,0	178814,89410,6.664
DE	10	50	40.952	20.988	57.531	8.169	12,18,-	0,0,0	7773,3886,0
		100	36.296	21.296	47.594	6.887	11,17,-	0,0,0	7640,3820,0
	50	250	24663.786	16653.193	32759.130	3524.930	50,81,-	0,0,0	74945,37472,1.131e-6
		500	24043.049	18093.957	29744.121	2651.444	59,88,-	0,0,0	57818,28909,0
	100	500	164897.363	140921.296	190475.859	10392.870	82,-,-	0,0,0	200854,100433,13
		1000	158593.835	130029.976	185909.828	11547.535	86,-,-	0,0,0	178814,89410,6.664
PSO	10	50	3715.526	197.487	8349.396	1964.826	7,-,-	0,0,0	7773,3886,0
		100	1940.642	90.809	6032.198	1249.686	6,9,-	0,0,0	7640,3820,0
	50	250	52558.785	44857.171	66291.898	4610.389	8,-,-	0,0,0	74945,37472,1.131e-6
		500	43699.982	34884.230	49462.332	4140.939	10,-,-	0,0,0	57818,28909,0
	100	500	137692.055	120734.039	153308.890	8424.796	7,-,-	0,0,0	200854,100433,13
		1000	123272.118	109356.968	140148.859	8669.564	8,-,-	0,0,0	178814,89410,6.664

Iterations = 100, Experiments = 30, Bold: the best case
$f_2(\boldsymbol{y}) = \sum_{i=1}^{n}(y_i^2 - 10\cos(2\pi y_i) + 10), \boldsymbol{y} = \boldsymbol{x} - \boldsymbol{a}, s.t. |x_i| \leqslant 128$
$a_i = \omega_{(i-1) \bmod 5+1}, \{\omega_m\} = \{40, 45, 50, 55, 60\}$
$\min(f_2) = f_2(\mathbf{0}) = 0$

BES, default parameter settings in [1] are used: change controlling parameter $\alpha = 2$, corner parameter $a = 10$, numbers of search cycles $R = 1.5$, movement intensities $c_1 = 2$, $c_2 = 2$. In **NAA**, default parameter settings III in [18] are used: number of shelters $N^S = 8$, shelter capacity $Cp^S = avg$, scaling factor $\delta = 2$, located crossover factor $Cr_{local} = 0.8$, movement amplification $\alpha = 1.5$, and generalized crossover factor $Cr_{global} = 0.5$. In **BA** [28], minimal frequency $f_{\min} = 0$, maximum frequency $f_{\max} = 1$, initial loudness $A_t = 1$, initial pulse rate $r_t = 0.5$, cooling factors $\alpha = 0.9$, $\gamma = 0.9$. In **DE**, default parameter settings in [4] are used: minimal amplification factor $F_l = 0.1$, maximal amplification factor $F_u = 0.9$, control probabilities $\tau_1 = 0.1$,

Table 3
Performance comparison for shifted HgBat function f_3

f_5	D	Ω	Mean Fitness	Best Fitness	Worst Fitness	Fitness Std.	Iterations	Championships	Milestones Values
IOA	10	50	**0.200**	**0.069**	0.321	0.074	2,3,18	27,29,29	6665,3333,0.308
		100	**0.214**	**0.053**	0.406	0.075	2,3,-	28,30,1	3503,1752,0.071
	50	250	**0.057**	**0.014**	0.096	0.020	2,3,24	27,28,16	72473,36237,0.059
		500	**0.155**	**0.103**	0.260	0.038	2,3,26	28,29,27	53806,26903,0.210
	100	500	**0.057**	**0.028**	0.112	0.020	1,3,9	30,18,7	227544,113772,0.040
		1000	**0.184**	**0.101**	0.309	0.048	2,3,15	21,18,12	183435,91718,0.173
HHO	10	50	440.201	182.749	529.154	77.456	3,3,-	13,22,0	6665,3333,0.308
		100	384.696	222.583	520.779	74.812	3,3,-	18,21,0	3503,1752,0.071
	50	250	2379.130	2163.985	2517.291	94.880	2,3,-	27,30,0	72473,36237,0.059
		500	2257.638	1969.458	2470.369	140.505	2,3,-	29,29,0	53806,26903,0.210
	100	500	4778.114	4277.145	4949.090	130.178	2,3,-	6,29,0	227544,113772,0.040
		1000	4675.818	4029.928	4939.508	188.083	2,3,-	22,30,0	183435,91718,0.173
BES	10	50	4404.675	411.163	12260.812	3171.062	6,6,-	7,0,0	6665,3333,0.308
		100	716.570	10.054	3105.622	810.192	9,11,-	9,2,0	3503,1752,0.071
	50	250	67071.455	40415.496	91503.398	10928.388	14,-,-	3,0,0	72473,36237,0.059
		500	41767.752	24470.25	55430.433	9370.582	17,-,-	6,0,0	53806,26903,0.210
	100	500	166294.704	139674.765	185631.031	11586.089	2,-,-	9,0,0	227544,113772,0.040
		1000	113693.110	87911.273	149331.812	13614.819	2,7,-	16,0,0	183435,91718,0.173
SSA	10	50	6726.298	735.422	13706.647	3326.707	14,-,-	2,0,0	6665,3333,0.308
		100	4798.336	1926.090	9290.313	1889.354	16,-,-	0,0,0	3503,1752,0.071
	50	250	67088.838	53398.382	79251.859	6766.104	24,-,-	0,0,0	72473,36237,0.059
		500	59636.975	47855.457	70920.945	6110.779	13,-,-	0,0,0	53806,26903,0.210
	100	500	159324.926	121853.664	192330.671	13284.951	3,-,-	0,0,0	227544,113772,0.040
		1000	145407.408	127,591	166977.656	9562.611	4,-,-	2,0,0	183435,91718,0.173
NAA	10	50	0.566	0.344	1.035	0.156	13,20,-	0,0,0	6665,3333,0.308
		100	0.402	0.246	0.573	0.079	14,19,-	0,0,0	3503,1752,0.071
	50	250	18522.084	9171.252	30730.691	4907.245	51,76,-	0,0,0	72473,36237,0.059
		500	8378.927	5572.424	15998.916	2453.959	46,66,-	0,0,0	53806,26903,0.210
	100	500	166078.792	124933.765	242144.203	30688.497	74,-,-	0,0,0	227544,113772,0.040
		1000	101291.732	51858.574	175940.937	25437.182	70,-,-	0,0,0	183435,91718,0.173
GWO	10	50	115.263	0.229	1738.640	428.647	5,11,-	3,1,0	6665,3333,0.308
		100	0.584	0.301	1.173	0.149	6,13,-	1,1,0	3503,1752,0.071
	50	250	38243.694	28084.574	48541.269	4577.017	22,26,-	0,0,0	72473,36237,0.059
		500	30470.741	15839.681	39570.265	6219.381	28,-,-	0,0,0	53806,26903,0.210
	100	500	140421.518	121349.859	161884.812	11733.629	22,-,-	0,0,0	227544,113772,0.040
		1000	123177.727	103519.289	137262.406	7351.114	27,-,-	0,0,0	183435,91718,0.173
BA	10	50	7221.572	2064.014	14778.302	3131.408	15,-,-	0,0,0	6665,3333,0.308
		100	5006.701	2240.206	9197.404	2008.445	12,-,-	0,0,0	3503,1752,0.071
	50	250	61704.121	42860.917	89834.695	10159.136	43,-,-	0,0,0	72473,36237,0.059
		500	51584.417	39970.570	73487.921	8451.485	36,-,-	0,0,0	53806,26903,0.210
	100	500	140963.635	110536.125	166661.187	16990.451	18,-,-	0,0,0	227544,113772,0.040
		1000	124152.653	101082.687	160144.468	15402.135	31,-,-	0,0,0	183435,91718,0.173
DE	10	50	0.728	0.288	1.962	0.391	15,20,-	0,0,0	6665,3333,0.308
		100	0.484	0.354	0.694	0.085	18,25,-	0,0,0	3503,1752,0.071
	50	250	24420.404	18967.652	33078.546	2993.532	50,83,-	0,0,0	72473,36237,0.059
		500	23346.833	18726.892	27700.5	2341.204	63,90,-	0,0,0	53806,26903,0.210
	100	500	159745.885	130232.523	193416.265	13767.818	72,-,-	0,0,0	227544,113772,0.040
		1000	159548.367	132526.171	181928.5	9643.387	89,-,-	0,0,0	183435,91718,0.173
PSO	10	50	3751.236	866.988	11100.148	2140.414	8,-,-	0,0,0	6665,3333,0.308
		100	1812.766	137.500	4390.377	1086.406	9,-,-	0,0,0	3503,1752,0.071
	50	250	51664.182	41388.625	65257.367	5514.214	8,-,-	0,0,0	72473,36237,0.059
		500	44055.516	33641.621	51606.789	4519.317	11,-,-	0,0,0	53806,26903,0.210
	100	500	137999.915	118009.507	166594.312	12029.673	6,-,-	0,0,0	227544,113772,0.040
		1000	125851.823	112671.343	140,683	7658.860	8,-,-	0,0,0	183435,91718,0.173

Iterations = 100, Experiments = 30, Bold: the best case
$f_3(\boldsymbol{y}) = \left|\left(\sum_{i=1}^{n} y_i^2\right)^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right|^{1/2} + (0.5\sum_{i=1}^{n} y_i^2 + \sum_{i=1}^{n} y_i)/n + 0.5, \boldsymbol{y} = \boldsymbol{x} - \boldsymbol{a}, s.t. |x_i| \leqslant 128$
$a_i = \omega_{(i-1) \bmod 5+1}, \{\omega_m\} = \{40, 45, 50, 55, 60\}$
$\min(f_3) = 0$

$\tau_2 = 0.1$. In **PSO** [15], inertia weight, personal best weight, and global best weight are all set as 1/3. **HHO** [7], **SSA** [20], and **GWO** [21] have no hyper-parameters.

To evaluate the convergence speed and execution duration of IOA in detail, the standard IOA is further divided into several procedures: LCPS, LCDS, FS, WS, and RLS that combines FS and WS. Thus, 3 cases are considered and illustrated in Figs. 12-14: case 1 disables WS, case 2 disables FS, and case 3 is based on RLS. For each case, 30 independent experiments are performed, the average fitness values and execution durations of each procedure are provided based on the discrete Ras-

Table 4
Performance comparison for shifted Happy Cat function f_4

f_6	D	Ω	Mean Fitness	Best Fitness	Worst Fitness	Fitness Std.	Iterations	Championships	Milestones Values
IOA	10	50	**0.112**	**0.037**	0.210	0.047	2,3,16	30,30,16	239,119,0.117
		100	**0.105**	**0.017**	0.187	0.041	2,3,23	25,29,27	267,133,0.166
	50	250	**0.037**	**0.015**	0.095	0.015	2,3,22	30,30,13	680,340,0.033
		500	**0.130**	**0.072**	0.194	0.030	2,3,16	29,29,11	617,308,0.112
	100	500	**0.050**	**0.013**	0.105	0.022	2,3,30	27,28,15	848,424,0.044
		1000	**0.134**	**0.072**	0.193	0.027	2,3,33	20,19,21	881,441,0.146
HHO	10	50	26.632	14.573	29.720	3.357	3,4,-	19,8,0	239,119,0.117
		100	24.416	12.577	29.085	3.967	3,3,-	13,22,0	267,133,0.166
	50	250	29.864	26.097	32.015	1.516	3,3,-	29,29,0	680,340,0.033
		500	29.206	24.730	31.668	1.690	3,3,-	28,30,0	617,308,0.112
	100	500	32.129	30.182	33.164	0.766	2,3,-	29,30,0	848,424,0.044
		1000	31.560	29.728	33.223	0.857	2,2,-	25,30,0	881,441,0.146
BES	10	50	247.598	17.445	562.313	161.828	6,12,-	5,1,0	239,119,0.117
		100	36.725	1.797	203.422	49.189	4,9,-	16,10,0	267,133,0.166
	50	250	616.043	420.068	819.645	113.082	7,-,-	3,0,0	680,340,0.033
		500	404.658	184.096	614.806	98.889	6,6,-	11,0,0	617,308,0.112
	100	500	769.350	613.338	913.252	80.532	9,-,-	6,0,0	848,424,0.044
		1000	568.713	435.239	712.243	65.147	2,-,-	21,0,0	881,441,0.146
SSA	10	50	253.688	78.649	565.075	125.894	20,-,-	0,0,0	239,119,0.117
		100	215.936	105.708	403.557	84.112	19,-,-	3,0,0	267,133,0.166
	50	250	657.085	525.077	847.802	68.816	21,-,-	0,0,0	680,340,0.033
		500	577.010	486.494	658.829	40.780	20,-,-	1,0,0	617,308,0.112
	100	500	786.256	697.714	880.953	45.970	16,-,-	0,0,0	848,424,0.044
		1000	720.931	643.107	799.920	35.449	5,-,-	0,0,0	881,441,0.146
NAA	10	50	0.538	0.227	0.804	0.121	16,22,-	0,0,0	239,119,0.117
		100	0.443	0.293	0.674	0.074	12,17,-	0,0,0	267,133,0.166
	50	250	211.485	127.413	346.892	47.608	55,79,-	0,0,0	680,340,0.033
		500	94.801	47.353	137.237	23.930	43,63,-	0,0,0	617,308,0.112
	100	500	832.793	542.108	1164.676	157.001	58,-,-	0,0,0	848,424,0.044
		1000	524.484	290.778	785.459	118.268	72,-,-	0,0,0	881,441,0.146
GWO	10	50	9.881	0.462	104.255	26.509	8,21,-	1,0,0	239,119,0.117
		100	0.542	0.266	0.895	0.143	5,8,-	0,0,0	267,133,0.166
	50	250	388.901	250.678	509.626	60.756	23,-,-	0,0,0	680,340,0.033
		500	294.384	176.415	388.706	47.009	19,48,-	0,0,0	617,308,0.112
	100	500	669.796	564.491	760.227	45.157	41,-,-	0,0,0	848,424,0.044
		1000	613.944	516.643	678.217	35.232	29,-,-	0,0,0	881,441,0.146
BA	10	50	319.275	104.864	566.261	133.978	11,-,-	0,0,0	239,119,0.117
		100	212.979	84.354	392.292	70.354	25,-,-	0,0,0	267,133,0.166
	50	250	623.180	448.139	766.435	101.199	36,-,-	0,0,0	680,340,0.033
		500	528.232	385.697	659.022	74.286	45,-,-	0,0,0	617,308,0.112
	100	500	718.315	587.074	939.180	78.724	51,-,-	0,0,0	848,424,0.044
		1000	621.777	487.835	800.135	75.556	34,-,-	0,0,0	881,441,0.146
DE	10	50	0.563	0.226	0.727	0.127	17,23,-	0,0,0	239,119,0.117
		100	0.518	0.238	0.690	0.092	14,20,-	0,0,0	267,133,0.166
	50	250	255.172	195.926	311.517	29.754	54,86,-	0,0,0	680,340,0.033
		500	252.185	167.539	292.612	25.592	58,91,-	0,0,0	617,308,0.112
	100	500	819.469	672.798	938.418	70.764	65,-,-	0,0,0	848,424,0.044
		1000	806.682	662.483	875.516	50.654	92,-,-	0,0,0	881,441,0.146
PSO	10	50	160.176	17.587	398.091	98.610	7,-,-	0,0,0	239,119,0.117
		100	84.998	11.239	198.398	46.306	8,9,-	0,0,0	267,133,0.166
	50	250	501.543	362.199	677.146	72.545	9,-,-	0,0,0	680,340,0.033
		500	421.695	274.605	545.796	62.537	9,-,-	0,0,0	617,308,0.112
	100	500	669.169	582.846	728.488	35.121	9,-,-	0,0,0	848,424,0.044
		1000	617.401	545.610	721.032	38.450	8,-,-	0,0,0	881,441,0.146

Iterations = 100, Experiments = 30, Bold: the best case

$f_4(\boldsymbol{y}) = \left|\sum_{i=1}^{n} y_i^2 - n\right|^{1/4} + (0.5\sum_{i=1}^{n} y_i^2 + \sum_{i=1}^{n} y_i)/n + 0.5, \boldsymbol{y} = \boldsymbol{x} - \boldsymbol{a}, s.t. |x_i| \leqslant 128$
$a_i = \omega_{(i-1) \bmod 5+1}, \{\omega_m\} = \{40, 45, 50, 55, 60\}$
$\min(f_4) = 0$

trigin benchmark function. In IOA, the dimension is 100, the search agent population is 1000, the number of leaders is 100 which equals the dimension, and the discounting factor for the role learner is 0.9. All 3 figures show that LCDS has the highest overall convergence speed and the longest execution duration. LCPS has the lowest execution duration and converges remarkably faster when FS or RLS is enabled. Compare Fig. 12 with Fig. 13, FS significantly speeds up IOA than WS, and WS is more time-consuming than FS. This implies that FS has a great convergence property and makes the LCPS procedure converge faster. However, it does not conclude that WS is an inferior procedure because exploring the search space is the main purpose of WS.

Table 5
Performance comparison for shifted Griewank function f_5

f_7	D	Ω	Mean Fitness	Best Fitness	Worst Fitness	Fitness Std.	Iterations	Championships	Milestones Values
IOA	10	50	**0.056**	**0.007**	0.152	0.038	2,2,6	29,30,9	3.644,1.840,0.036
		100	**0.047**	**0**	0.123	0.030	2,3,15	24,29,29	2.302,1.207,0.113
	50	250	**4.924e-4**	**0**	0.014	0.002	2,3,53	30,30,29	18,9.215,0
		500	**0.004**	**0**	0.024	0.007	2,3,38	25,28,21	18,8.977,0
	100	500	**0.001**	**0**	0.019	0.004	2,3,50	28,10,21	53,26,0
		1000	**0.004**	**0**	0.029	0.006	2,3,43	13,12,19	48,24,0
HHO	10	50	1.065	0.816	1.119	0.073	2,3,-	6,14,0	3.644,1.840,0.036
		100	1.016	0.748	1.115	0.096	3,5,-	14,7,0	2.302,1.207,0.113
	50	250	1.577	1.501	1.618	0.030	3,3,-	29,30,0	18,9.215,0
		500	1.557	1.465	1.612	0.034	2,3,-	25,29,0	18,8.977,0
	100	500	2.192	2.103	2.241	0.031	2,2,-	14,30,0	53,26,0
		1000	2.160	2.072	2.235	0.041	2,3,-	17,30,0	48,24,0
BES	10	50	2.367	0.667	5.573	1.172	2,9,-	14,2,0	3.644,1.840,0.036
		100	1.326	0.437	2.476	0.507	6,11,-	14,0,0	2.302,1.207,0.113
	50	250	17.985	13.491	22.466	2.284	10,-,-	0,0,0	18,9.215,0
		500	11.508	7.290	19.576	2.662	4,6,-	14,0,0	18,8.977,0
	100	500	40.768	31.201	46.958	3.978	3,-,-	9,0,0	53,26,0
		1000	30.316	19.960	38.699	4.986	2,4,-	23,0,0	48,24,0
SSA	10	50	2.665	1.457	4.922	0.830	17,-,-	4,0,0	3.644,1.840,0.036
		100	2.028	1.031	3.571	0.560	18,-,-	5,0,0	2.302,1.207,0.113
	50	250	18.197	14.887	20.819	1.403	18,-,-	0,0,0	18,9.215,0
		500	15.861	12.356	19.062	1.568	20,-,-	0,0,0	18,8.977,0
	100	500	40.625	36.682	44.515	2.125	3,-,-	3,0,0	53,26,0
		1000	38.204	32.620	42.110	2.238	4,-,-	2,0,0	48,24,0
NAA	10	50	0.450	0.261	0.600	0.089	9,19,-	2,0,0	3.644,1.840,0.036
		100	0.366	0.214	0.505	0.074	12,24,-	0,0,0	2.302,1.207,0.113
	50	250	6.403	3.596	11.678	1.779	57,78,-	0,0,0	18,9.215,0
		500	3.141	2.188	4.552	0.630	41,64,-	0,0,0	18,8.977,0
	100	500	46.897	30.966	71.934	9.731	72,-,-	0,0,0	53,26,0
		1000	27.977	17.860	42.516	6.214	70,-,-	0,0,0	48,24,0
GWO	10	50	0.695	0.130	1.141	0.260	4,13,-	0,0,0	3.644,1.840,0.036
		100	0.641	0.108	0.921	0.210	5,26,-	1,0,0	2.302,1.207,0.113
	50	250	10.067	7.286	15.923	1.814	23,-,-	0,0,0	18,9.215,0
		500	8.470	5.814	12.087	1.377	16,46,-	0,0,0	18,8.977,0
	100	500	35.075	30.887	38.737	2.406	25,-,-	0,0,0	53,26,0
		1000	31.983	27.552	36.117	2.384	26,-,-	0,0,0	48,24,0
BA	10	50	3.214	1.567	5.537	0.913	17,-,-	0,0,0	3.644,1.840,0.036
		100	2.595	1.326	4.100	0.703	12,-,-	0,0,0	2.302,1.207,0.113
	50	250	17.303	13.646	22.352	2.127	45,-,-	0,0,0	18,9.215,0
		500	14.076	10.666	18.124	1.870	44,-,-	0,0,0	18,8.977,0
	100	500	36.840	26.958	46.464	4.024	28,-,-	0,0,0	53,26,0
		1000	32.744	25.117	41.037	4.155	30,-,-	0,0,0	48,24,0
DE	10	50	0.422	0.215	0.567	0.091	10,21,-	1,0,0	3.644,1.840,0.036
		100	0.380	0.179	0.542	0.085	14,32,-	0,0,0	2.302,1.207,0.113
	50	250	6.759	4.909	8.464	0.754	52,86,-	0,0,0	18,9.215,0
		500	6.827	5.609	7.866	0.504	51,86,-	0,0,0	18,8.977,0
	100	500	41.312	31.991	47.454	3.324	77,-,-	0,0,0	53,26,0
		1000	39.510	32.097	45.148	3.211	85,-,-	0,0,0	48,24,0
PSO	10	50	1.718	0.718	3.327	0.685	7,9,-	0,0,0	3.644,1.840,0.036
		100	1.156	0.262	2.305	0.470	8,15,-	0,0,0	2.302,1.207,0.113
	50	250	13.906	10.836	17.035	1.262	9,-,-	0,0,0	18,9.215,0
		500	12.148	9.112	15.008	1.391	8,-,-	0,0,0	18,8.977,0
	100	500	35.036	29.963	38.925	2.282	7,-,-	0,0,0	53,26,0
		1000	32.661	28.096	37.164	2.295	7,-,-	0,0,0	48,24,0

Iterations = 100, Experiments = 30, Bold: the best case
$f_5(\boldsymbol{y}) = \frac{1}{4000}\sum_{i=1}^{n} y_i^2 - \prod_{i=1}^{n}\cos\left(\frac{y_i}{\sqrt{i}}\right) + 1, \boldsymbol{y} = \boldsymbol{x} - \boldsymbol{a}, s.t. |x_i| \leqslant 128$
$a_i = \omega_{(i-1) \ mod \ 5+1}, \{\omega_m\} = \{40, 45, 50, 55, 60\}$
$\min(f_5) = f_5(\mathbf{0}) = 0$

6. Experiments on unit commitment problems

IOA is designed as a general-purpose optimization solver that tackles both continuous and non-continuous optimization problems. We apply IOA to solve the unit commitment (UC) problems in the power system as a real-world application. The optimization results of IOA are compared to the results of HHO, NAA, GWO, DE, and PSO. IOA shows the fastest convergence speed and obtains the best overall fitness values.

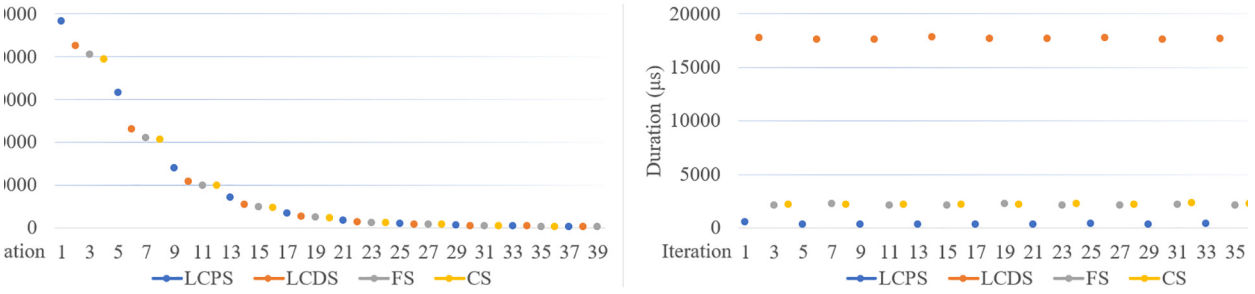

Fig. 12. Fitness values and execution durations when the follower search is enabled.

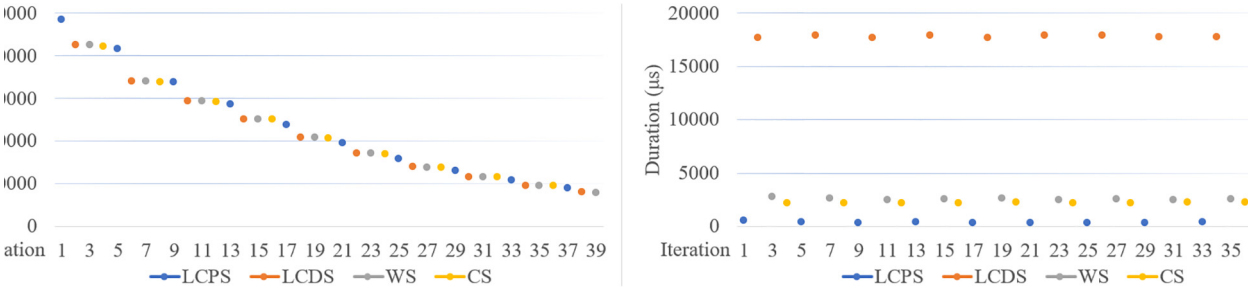

Fig. 13. Fitness values and execution durations when the wanderer search is enabled.

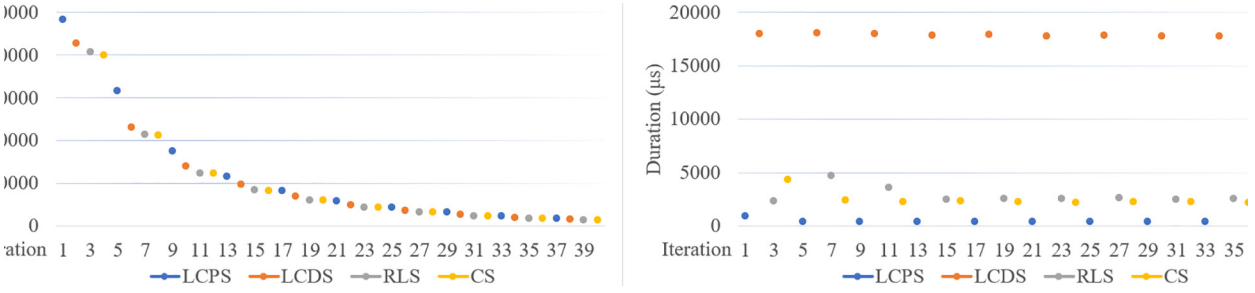

Fig. 14. Fitness values and execution durations when both the follower search and wanderer search are enabled.

6.1. The unit commitment problem

Solving UC problems is a vital task in power system operation, and it has attracted researchers' interest over the years [6,13,14,16,22]. UC aims to find the best unit on–off combinations and power generation schedules to minimize the generation cost. The constraints in UC include power balance, spinning reserve, power generation boundary, minimal uptime, minimal downtime, *etc.* Mathematically, UC is a large-scale non-linear optimization problem involving both continuous and integer programming.

6.2. Problem formulation

Define N as the number of units; T as the number of time intervals; $i \in [1, N]$ as the unit index; $j \in [1, T]$ as the time interval index; C as the total generation cost; C_i as the generation cost for unit i; C_i^{hot} as the hot-start cost for unit i; C_i^{cold} as the cold-start cost for unit i; p_{ij} as the power generated for unit i at time j; $P_i^{\min}$ and $P_i^{\max}$ as the minimal and maximum power generation for unit i; P_j^D as the system load at time j; P_j^R as the spinning reserve at time j; $s_{ij} \in \{0, 1\}$ as the on–off status for unit i at time j; L_{ij} as the start-up cost for unit i at time j; T_i^{up} as the minimal uptime for unit i; T_i^{down} as the minimal downtime for unit i; t_{ij}^{down} as the consecutive downtime for unit i up to time j; t_i^{up} and t_i^{down} as cumulative uptime and downtime for unit i.

The objective function of UC is:

$$\min C = \sum_{j=1}^{T}\sum_{i=1}^{N} C_i s_{ij} + L_{ij} s_{ij}(1 - s_{i,j-1}), C_i = a_i p_{ij}^2 + b_i p_{ij} + c_i$$
$$L_{ij} = \begin{cases} C_i^{hot} & t_{ij}^{down} \leqslant T_i^{down} + T_i^{cold} \\ C_i^{cold} & t_{ij}^{down} > T_i^{down} + T_i^{cold} \end{cases} \tag{27}$$

The power balance constraints are:

$$\sum_{i=1}^{N} p_{ij} s_{ij} = P_j^D \tag{28}$$

The spinning reserve constraints are:

$$\sum_{i=1}^{N} P_i^{\max} s_{ij} \geqslant P_j^D + P_j^R \tag{29}$$

The power generation boundary constraints are:

$$P_i^{\min} s_{ij} \leqslant p_{ij} s_{ij} \leqslant P_i^{\max} s_{ij} \tag{30}$$

The minimal uptime and downtime constraints are:

$$t_i^{up} \geqslant T_i^{up}, t_i^{down} \geqslant T_i^{down} \tag{31}$$

6.3. Simulation experiments

In the simulation experiment, a 5-unit system and a 10-unit system are used as the UC application scenarios. For both scenarios, the timespan is 24 h. Therefore, there are $5 \times 2 \times 24 = 240$ dimensions (with 5×24 0–1 variables and 5×24 continuous variables) and $10 \times 2 \times 24 = 480$ dimensions (with 10×24 0–1 variables and 10×24 continuous variables) for the 5 and 10 units scenarios respectively.

Both scenarios are performed and compared using IOA, HHO, NAA, GWO, DE, and PSO algorithms (BES, SSA, and BA have inferior performance compared to the 6 aforementioned algorithms). 30 independent experiments are carried out for each scenario. The experiment results render the mean fitness, best fitness, worst fitness, fitness *std*, mean iterations, and milestone fitness values.

The hyperparameter settings for HHO, NAA, GWO, DE, and PSO are the same as in the benchmark function experiments. For IOA, the maximal number of leaders is set as half of the dimension because the unit on–off statuses and power generation values are correlated. The probability of 1 is assigned to the option of following all leaders so that local minima obtained by integer programming can be avoided.

6.3.1. 5-unit scenario

The algorithm convergence graph is shown in Fig. 15, the parameter settings for the 5-unit UC are shown in Table 6 and 7, the unit statuses are shown in Table 8, the power generation schedule is shown in Table 9, and the algorithm comparison results are shown in Table 10.

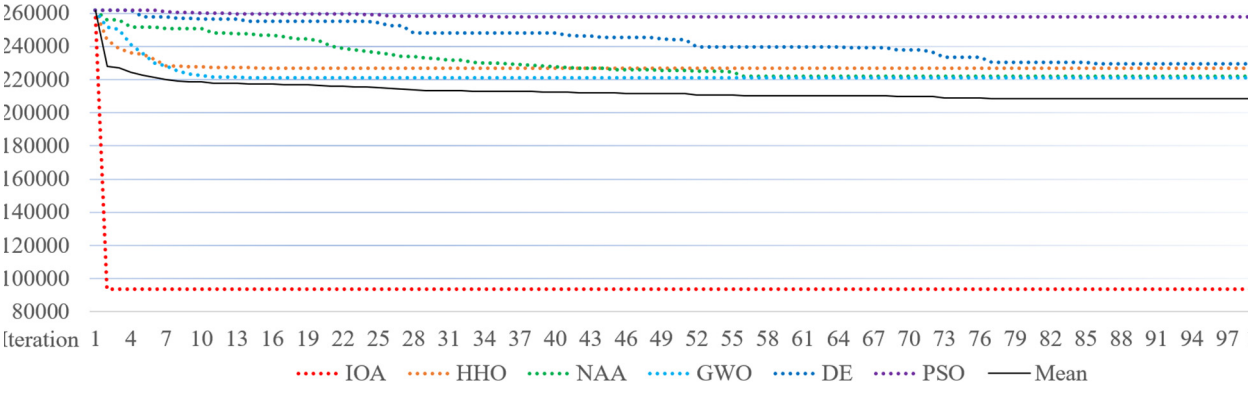

Fig. 15. Algorithm convergence graph for the 5-unit scenario with population of 2400.

Table 6
5-unit parameter settings

Parameters\Unit	Unit 1	Unit 2	Unit 3	Unit 4	Unit 5
g. (MW)	150	20	20	20	55
x.g. (MW)	600	550	450	450	400
a ($/MW^{2}h)	0.00048	0.002	0.00211	0.00712	0.00413
b ($/MWh)	16.19	16.6	16.5	22.26	25.92
c ($/h)	1000	700	680	370	660
m.u. (h)	8	5	5	3	1
m.d. (h)	8	5	5	3	1
c.s.t. (h)	5	4	4	2	0
i.c.d.t. (h)	0	0	1	2	1
h.s.c. ($)	4500	550	560	170	30
c.s.c. ($)	9000	1100	1120	340	60

Note: n.g.: minimal generation; x.g.: maximal generation; m.u.: minimal uptime; m.d.: minimal downtime; c.s.t.: cold start time; i.c.d.t.: initial consecutive downtime; h.s.c.: hot start cost; c.s.c.: cold start cost

Table 7
5-unit load over 24 h.

Time (h)	1	2	3	4	5	6	7	8
P (MW)	300	350	380	400	430	450	480	500
Time (h)	9	10	11	12	13	14	15	16
P (MW)	520	550	580	600	550	520	500	450
Time (h)	17	18	19	20	21	22	23	24
P (MW)	400	450	500	550	500	450	400	350

Note: P: load; spinning reserve equals 10% of the load.

Table 8
5-unit UC unit statuses. Note: u.: unit; h.: hour

u.\h.	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	1	1	1	0	0	0	0	0	1	1
2	1	1	0	0	0	1	1	1	1	1	1	1
3	1	0	1	1	1	0	0	0	1	1	0	0
4	0	0	1	1	0	1	1	1	1	1	0	0
5	1	1	1	0	0	0	1	0	0	0	0	0
u.\h.	13	14	15	16	17	18	19	20	21	22	23	24
1	0	0	0	0	0	1	1	1	0	0	0	0
2	1	1	1	1	1	0	0	1	0	0	0	0
3	1	1	0	0	1	0	0	0	1	1	0	1
4	0	0	1	0	0	1	0	1	1	0	1	0
5	0	1	1	0	0	1	1	1	1	1	0	0

Table 9
5-unit UC power generation schedule (MW). Note: u.: unit; h.: hour.

u.\h.	1	2	3	4	5	6	7	8	9	10	11	12
1	0	150	285	360	150	0	0	0	0	0	150	150
2	20	20	0	0	0	430	405	480	480	510	430	450
3	225	0	20	20	280	0	0	0	20	20	0	0
4	0	0	20	20	0	20	20	20	20	20	0	0
5	55	180	55	0	0	0	55	0	0	0	0	0
u.\h.	13	14	15	16	17	18	19	20	21	22	23	24
1	0	0	0	0	0	150	150	150	0	0	0	0
2	530	20	20	450	20	0	0	325	0	0	0	0
3	20	445	0	0	380	0	0	0	20	395	0	350
4	0	0	425	0	0	245	0	20	425	0	400	0
5	0	55	55	0	0	55	350	55	55	55	0	0

Table 10
Algorithm comparison for the 5-unit UC.

al.	pop.	m.f. ($)	b.f. ($)	w.f. ($)	f.std. ($)	Iterations	Milestone Values ($)
IOA	1200	**171744.34**	108701.08	218400.84	43080.69	1,2,8	257308,237619,217930
	2400	**172306.81**	93764.23	220833.78	37700.60	1,2,2	255384,235823,216261
HHO	1200	228425.24	220227.18	237830.51	4077.66	1,5,-	257308,237619,217930
	2400	226370.52	218790.64	235427.14	3789.01	1,5,-	255384,235823,216261
NAA	1200	219652.74	217470.10	224034.21	1874.63	3,21,-	257308,237619,217930
	2400	219106.81	216857.96	222172.78	1691.56	4,19,-	255384,235823,216261
GWO	1200	221835.22	217367.29	228182.54	2434.01	1,4,-	257308,237619,217930
	2400	221304.96	217053.43	225236.54	2193.40	2,4,-	255384,235823,216261
DE	1200	226891.91	224375.10	233158.64	2257.47	8,64,-	257308,237619,217930
	2400	226283.21	223120.56	230433.93	1767.29	9,68,-	255384,235823,216261
PSO	1200	257437.68	252006.90	262044.20	2560.74	2,-,-	257308,237619,217930
	2400	256249.67	252482.32	259277.65	1613.99	5,-,-	255384,235823,216261

Note: al.: algorithm; pop.: population; m.f.: mean fitness; b.f.: best fitness; w.f.: worst fitness; f.std.: fitness standard deviation; Bold: the best case.

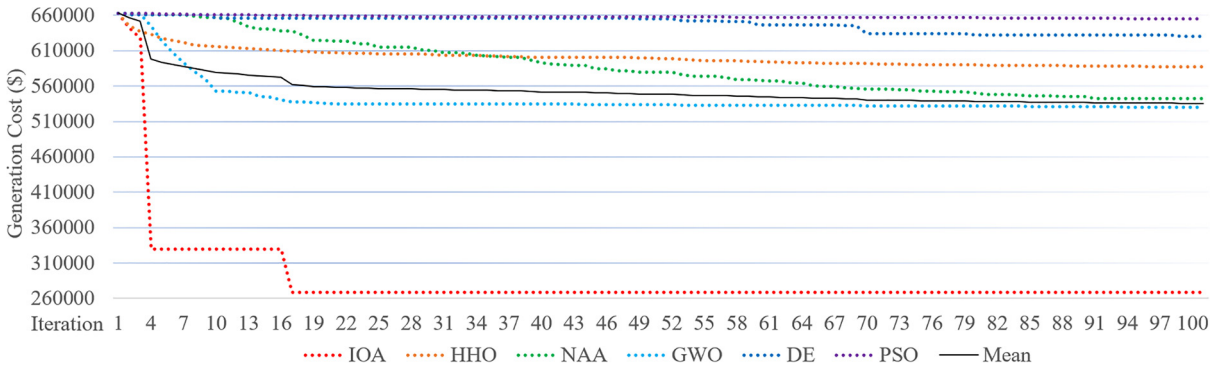

Fig. 16. Algorithm convergence graph for the 10-unit scenario with population of 4800.

Table 11
10-unit parameter settings.

Parameters\Unit	Unit 1	Unit 2	Unit 3	Unit 4	Unit 5	Unit 6	Unit 7	Unit 8	Unit 9	Unit 10
n.g. (MW)	150	150	20	20	25	20	25	10	10	10
x.g. (MW)	600	550	450	450	400	500	460	500	500	450
a ($/MW^{2}h)	0.00048	0.00031	0.002	0.00211	0.00398	0.00712	0.00079	0.00413	0.00222	0.00173
b ($/MWh)	16.19	17.26	16.6	16.5	19.7	22.26	27.74	25.92	27.27	27.79
c ($/h)	1000	970	700	680	450	370	480	660	665	670
m.u. (h)	8	8	5	5	6	3	3	1	1	1
m.d. (h)	8	8	5	5	6	3	3	1	1	1
c.s.t. (h)	5	5	4	4	4	2	2	0	0	0
i.c.d.t. (h)	0	0	1	1	2	1	1	0	0	0
h.s.c. ($)	4500	5000	550	560	900	4400	4500	500	460	720
c.s.c. ($)	9000	10,000	1100	1120	1800	8800	9000	1000	920	1440

Note: n.g.: minimal generation; x.g.: maximal generation; m.u.: minimal uptime; m.d.: minimal downtime; c.s.t.: cold start time; i.c.d.t.: initial consecutive downtime; h.s.c.: hot start cost; c.s.c.: cold start cost.

Table 12
10-unit load over 24 h. Note: P: load; spinning reserve equals 10% of the load

Time (h)	1	2	3	4	5	6	7	8	9	10	11	12
P (MW)	600	650	750	800	900	1000	1050	1100	1200	1300	1350	1400
Time (h)	13	14	15	16	17	18	19	20	21	22	23	24
P (MW)	1300	1200	1100	950	900	1000	1100	1300	1200	1000	800	700

Table 13
10-unit UC unit statuses. Note: u.: unit; h.: hour.

u.\h.	1	2	3	4	5	6	7	8	9	10	11	12
1	1	0	1	1	1	1	0	0	0	0	0	1
2	0	0	0	0	0	0	0	1	1	1	1	1
3	0	1	0	0	0	1	0	1	0	1	1	0
4	0	0	1	0	0	1	1	0	1	1	1	0
5	0	0	0	0	0	0	0	0	1	1	1	1
6	1	1	0	0	0	0	1	1	0	0	0	0
7	0	0	0	1	1	1	1	1	0	0	0	0
8	0	0	0	0	0	1	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0
u.\h.	13	14	15	16	17	18	19	20	21	22	23	24
1	1	1	1	1	1	1	1	1	1	1	0	0
2	1	1	1	1	1	0	0	0	0	0	1	1
3	0	0	0	0	0	1	1	0	1	1	0	0
4	0	0	0	0	0	1	1	1	0	1	1	1
5	1	1	0	0	0	0	0	1	1	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	1	0	0	0	0	0	0	0	0	0
10	0	0	0	1	0	0	0	0	0	0	0	0

Table 14
10-unit UC power generation schedule (MW). Note: u.: unit; h.: hour.

u.\h.	1	2	3	4	5	6	7	8	9	10	11	12
1	401	0	372	547	562	240	0	0	0	0	0	525
2	0	0	0	0	0	0	0	371	550	425	502	534
3	0	372	0	0	0	407	0	311	0	389	433	0
4	0	0	378	0	0	190	443	0	398	312	145	0
5	0	0	0	0	0	0	0	0	252	175	270	341
6	199	278	0	0	0	0	344	85	0	0	0	0
7	0	0	0	253	338	137	263	332	0	0	0	0
8	0	0	0	0	0	26	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0
u.\h.	13	14	15	16	17	18	19	20	21	22	23	24
1	565	545	472	379	392	471	417	526	525	235	0	0
2	490	489	413	506	508	0	0	0	0	0	487	399
3	0	0	0	0	0	239	237	0	355	395	0	0
4	0	0	0	0	0	291	446	399	0	370	313	301
5	245	166	0	0	0	0	0	375	321	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	215	0	0	0	0	0	0	0	0	0
10	0	0	0	65	0	0	0	0	0	0	0	0

6.3.2. 10-unit scenario

The algorithm convergence graph is shown in Fig. 16, the parameter settings for the 10-unit UC are shown in Table 11 and 12, the unit statuses are shown in Table 13, the power generation schedule is shown in Table 14, and the algorithm comparison results are shown in Table 15.

7. Experiments on neural network training

Power line maintenance is an arduous and dangerous task. Drone surveillance is a workable solution that significantly reduces labor costs. Apart from having sensors and telecommunication devices installed, an accurate vision system that recognizes the power line components is indispensable for the flying machine. A deep-learning-based image classification model is trained by DIOA. The model determines whether a captured image is a part of the power line or not. Based on this work, a real-time object detection system can be further created. Compared with the gradient descent (GD) method, DIOA converges much faster and quickly attains a high test set accuracy. Fig. 17 shows some sample images of the power line data set.

Table 15
Algorithm comparison for 10-unit UC.

al.	pop.	m.f. ($)	b.f. ($)	w.f. ($)	f.std. ($)	Iterations	Milestone Values ($)
IOA	2400	**360354.19**	282875.06	516409.53	55637.15	1,5,-	661162,513210,365257
	4800	**357579.76**	268801.12	517858.59	53764.36	2,3,3	647669,510585,373500
HHO	2400	602448.84	558374.93	624128.93	13349.23	1,-,-	661162,513210,365257
	4800	596161.92	567746.37	609332.87	10584.10	2,-,-	647669,510585,373500
NAA	2400	545111.29	515187.90	564381.68	12367.41	6,-,-	661162,513210,365257
	4800	541137.17	520640.28	574989.37	11853.02	12,-,-	647669,510585,373500
GWO	2400	536442.57	515245.62	564695.56	11308.70	2,-,-	661162,513210,365257
	4800	536657.34	513937.75	659793.75	24549.20	4,-,-	647669,510585,373500
DE	2400	625789.62	612407.62	632561.25	4541.25	15,-,-	661162,513210,365257
	4800	623032.27	607325.18	631963.75	5618.00	50,-,-	647669,510585,373500
PSO	2400	654553.86	642707.68	662597.81	4173.96	17,-,-	661162,513210,365257
	4800	651969.88	642573.31	660375.87	4465.35	13,-,-	647669,510585,373500

Note: al.: algorithm; pop.: population; m.f.: mean fitness; b.f.: best fitness; w.f.: worst fitness; f.std.: fitness standard deviation; Bold: the best case.

Fig. 17. Sample images of the power line data set. The first 3 images contain parts of the power lines while the last 3 images do not.

Table 16
Architecture of CNN.

Layer	#Filters	Filter Size	Stride	M.P.	L.N.	L.R.	Layer	#Neurons	L.N.	L.R.
CNN1	24	7	1	2	Yes	0.01	FC1	16	Yes	0.01
CNN2	48	5	1	2	Yes	0.01	FC2	2	No	–
CNN3	24	3	1	–	Yes	0.01				

Note: M.P.: max-pooling size; L.N.: whether the layer normalization [3] is applied; L.R.: negative side slope of the leaky-relu activation function.

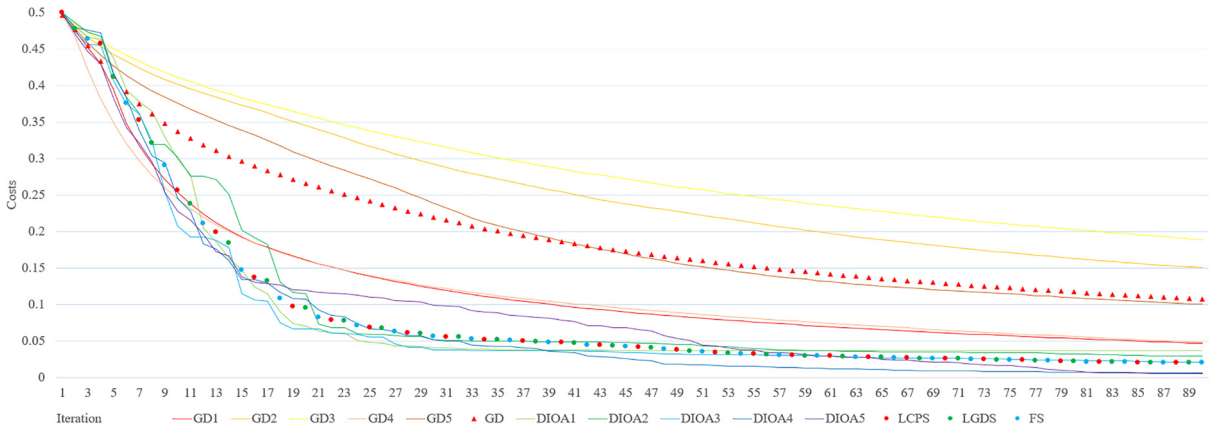

Fig. 18. Costs comparison between GD and DIOA. GD 1–5 represent 5 GD experiments; DIOA 1–5 represent 5 DIOA experiments.

The neural network training experiments are based on a small dataset that includes 400 images obtained from Google image search. 90% of the data are used for training and 10% of the data are used for testing. There is no overlap between the training set and the test set. The image classification model is a CNN consisting of 3 convolution layers, 2 fully connected layers, and a softmax classification layer. Table 16 describes the architecture of the CNN model. DIOA utilizes 6 search agents in total. 2 of which are leaders and the other 4 search agents are followers. In both GD and DIOA, the learning rate is 0.001, and the initial upper and lower bounds for the trainable parameters are −0.001 and 0.001. 90 training epochs are carried out

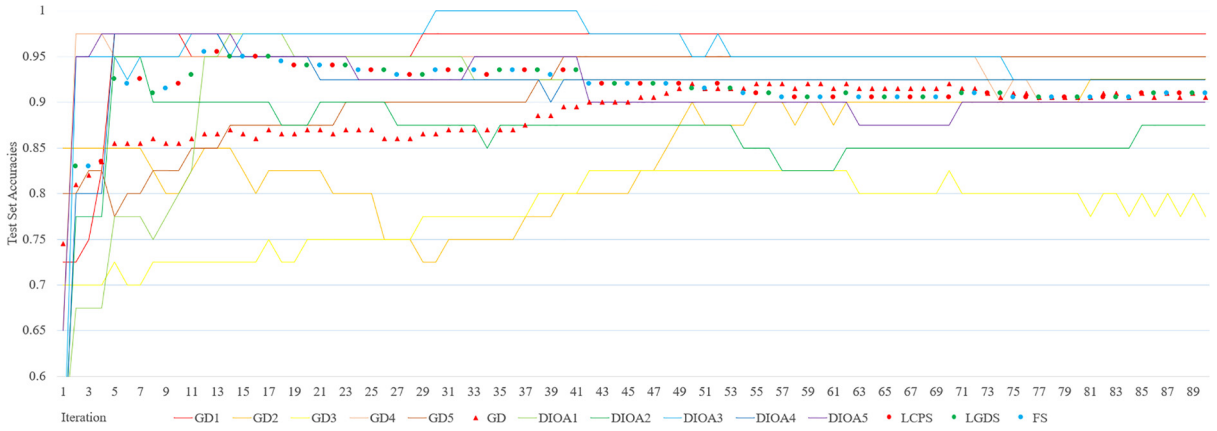

Fig. 19. Test set accuracies comparison between GD and DIOA. GD 1–5 represent 5 GD experiments; DIOA 1–5 represent 5 DIOA experiments.

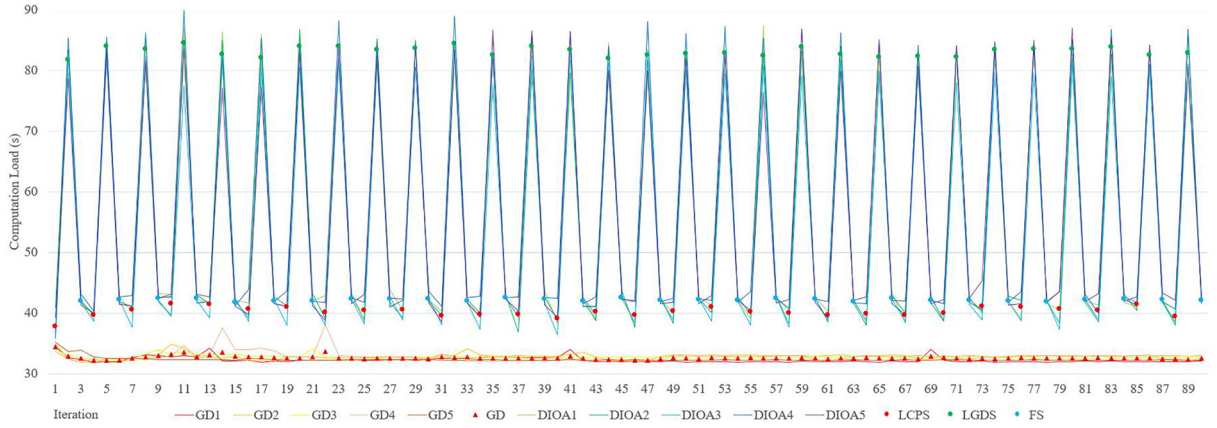

Fig. 20. Computation load comparison between GD and DIOA. GD 1–5 represent 5 GD experiments; DIOA 1–5 represent 5 DIOA experiments.

by GD. To make the comparison fair, only 30 heavyweight training epochs, each of which includes LCPS, LGDS, and FS procedures, are executed by DIOA. Therefore, DIOA also has 90 effective training epochs. 5 independent experiments are carried out for both GD and DIOA. DIOA is implemented in Java 11, C++ 17, and CUDA 10.2. The experiments are conducted on the same workstation that performs the benchmark function experiments.

Fig. 18 compares the cost function values of GD and DIOA. DIOA requires only 20 training epochs on average to achieve a cost of 0.1. While the cost manages to approach 0.1 for GD using 90 epochs on average. Therefore, DIOA converges 4.5 times faster than the conventional gradient descent algorithm in this experiment. All LCPS, LGDS, and FS make significant contributions to DIOA's convergence during the first 30 epochs. In the 15th and 18th epochs, FS converges even faster than LGDS because of the quasi-gradient descent mechanism.

Fig. 19 compares the test set accuracies of GD and DIOA. Since DIOA converges significantly faster than GD, the test set accuracy of DIOA is much higher than GD during the first 50 epochs. Remarkably, 100% of the test set accuracy is obtained in one of the DIOA experiments.

Fig. 20 shows the computation load for GD and DIOA. Although DIOA is more time-consuming than GD, the convergence speed gain of DIOA overweights its extra computation load. Fortunately, since DIOA is a highly parallelized algorithm, the performance of DIOA increases when using a CPU with more cores.

8. Conclusion

In this paper, a new metaheuristic algorithm IOA is proposed. IOA consists of 5 sub-algorithms: follower search, leader search, wanderer search, crossover search, and role learning. A detailed theoretical demonstration shows that IOA has great convergence properties and obtains global optima when a sufficient number of search agents are used. Case study results

based on 27 benchmark functions conclude that IOA has a much faster convergence speed and obtains better solutions than the other 8 algorithms: HHO, BES, SSA, NAA, GWO, BA, DE, and PSO. This implies that synthesizing a group of optimization algorithms is effective in solving complicated optimization problems. Additionally, a benchmark function experiment is performed to evaluate the convergence properties and execution duration of each procedure in IOA. IOA has also been applied to solve high-dimensional, non-convex, and non-continuous unit commitment problems in the power system and obtains satisfactory performance. DIOA is put forward to train a CNN model that classifies power line sub-images. Compared with the pure gradient descent approach, DIOA significantly speeds up model convergence and attains a high test set accuracy with much fewer training epochs.

IOA provides a framework for synthesizing multiple optimization algorithms. In the future, new sub-algorithms can be incorporated into IOA to render a more powerful integrated optimization algorithm. A procedure selection mechanism can be designed to automatically determine what procedures to utilize for a specific optimization problem. Moreover, GPU hardware acceleration can be used for enhancing the computation speed when the dimension is extraordinarily high. Also, distributed computation architectures can be considered so that the whole swarm in IOA is split into multiple sub-swarms which are processed by different machines. All future works above will significantly improve IOA's performance in solving large-scale optimization problems.

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.

Acknowledgments

This work is partially supported by the Australian Research Council under Grants DP180103217, FT190100156, IH180100020, LP200100056 and partially supported by funding from the UNSW Digital Grid Futures Institute, UNSW, Sydney, under a cross-disciplinary fund scheme.

References

[1] H. Alsattar, A. Zaidan, B. Zaidan, Novel meta-heuristic bald eagle search optimization algorithm, Artif. Intell. Rev. 53 (3) (2020) 2237–2264.
[2] N. H. Awad, M. Z. Ali, J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization," Nanyang Technological University, Jordan University of Science and Technology, and Zhengzhou University, Tech. Rep., Oct 2016.
[3] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," arXiv preprint arXiv:1607.06450, 2016.
[4] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (6) (2006) 646–657.
[5] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, A.A. Bharath, Generative adversarial networks: An overview, IEEE Signal Process Mag. 35 (1) (2018) 53–65.
[6] Z. Hasan, M.E. El-Hawary, Unit commitment incorporating wind energy by BBO and GA, in: IEEE Electrical Power and Energy Conference, 2016, pp. 1–7.
[7] A.A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: Algorithm and applications, Future Generation Computer Systems 97 (2019) 849–872.
[8] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1) (1992) 66–72.
[9] K. Hussain, M. Salleh, S. Cheng, R. Naseem, Common benchmark functions for metaheuristic evaluation: A review, JOIV: International Journal on Informatics Visualization 1 (4–2) (2017) 218–223.
[10] M. Jamil, X.-S. Yang, A literature survey of benchmark functions for global optimisation problems, International Journal of Mathematical Modelling and Numerical Optimisation 4 (2) (2013) 150–194.
[11] X. Ji, Y. Zhang, D. Gong, X. Sun, Dual-Surrogate Assisted Cooperative Particle Swarm Optimization for Expensive Multimodal Problems, IEEE Trans. Evol. Comput. (2021).
[12] J.-R. Jian, Z.-G. Chen, Z.-H. Zhan, J. Zhang, Region encoding helps evolutionary computation evolve faster: A new solution encoding scheme in particle swarm for large-scale optimization, IEEE Trans. Evol. Comput. 25 (4) (2021) 779–793.
[13] V.K. Kamboj, A novel hybrid PSO–GWO approach for unit commitment problem, Neural Comput. Appl. 27 (6) (2016) 1643–1655.
[14] S.A. Kazarlis, A.G. Bakirtzis, V. Petridis, A genetic algorithm solution to the unit commitment problem, IEEE Trans. Power Syst. 11 (1) (1996) 83–92.
[15] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proc. IEEE Int. Conf. Neural Network, pp. 1942-1948, Nov. 1995.
[16] P. Khazaei, M. Dabbaghjamanesh, A. Kalantarzadeh, H. Mousavi, Applying the modified TLBO algorithm to solve the unit commitment problem, World Automation Congress (2016) 1–6.
[17] J. Liang, B. Qu, P. Suganthan, Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, Computational Intelligence Laboratory, 2013.
[18] F. Luo, J. Zhao, and Z. Y. Dong, "A new metaheuristic algorithm for real parameter optimization: Natural aggregation algorithm," in Proc. IEEE Congr. Evol. Comput. (IEEE CEC), Vancouver, BC, Canada, Jul. 2016, pp. 94–103.
[19] F. Luo, Z. Y. Dong, Y. Chen, and J. Zhao, "Natural aggregation algorithm: A new efficient metaheuristic tool for power system optimizations," in Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm), Sydney, NSW, Australia, Nov. 2016, pp. 186–192.
[20] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (2017) 163–191.
[21] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[22] L.K. Panwar, S. Reddy K, A. Verma, B.K. Panigrahi, R. Kumar, Binary grey wolf optimizer for large scale unit commitment problem, Swarm Evol. Comput. 38 (2018) 251–266.
[23] K. Song, A. Lim, B. Rodrigues, Sexual selection for genetic algorithms, Artif. Intell. Rev. 19 (2) (2003) 123–152.
[24] R. Storn, K. Price, Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.

[25] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," KanGAL Report 2005005, no. 2005, 2005.
[26] Jianyong Sun, Xin Liu, Thomas Back, Zongben Xu, Learning adaptive differential evolution algorithm from optimization experiences by policy gradient, IEEE Trans. Evol. Comput. 25 (4) (2021) 666–680.
[27] Stephen J. Wright Coordinate descent algorithms 151 1 2015 3 34
[28] X. Yang A new metaheuristic bat-inspired algorithm Nature Inspired Cooperative Strategies for Optimization (NICSO) 2010 Springer Berlin, Heidelberg 65 74
[29] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, Xin Yao, A survey of evolutionary continuous dynamic optimization over two dec-ades–part A, IEEE Trans. Evol. Comput. (2021).
[30] C. T. Yue, K. V. Price, P. N. Suganthan, J. J. Liang, M. Z. Ali., B. Y. Qu, N. H. Awad, and P. P. Biswas, "Problem Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization," Zhengzhou University and Nanyang Technological University, Tech. Rep., November 2019. [Online]. Available: https://github.com/P-N-Suganthan/2020-Bound-Constrained-Opt-Benchmark