Computers and Electrical Engineering 100 (2022) 107862

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

A novel nomad migration-inspired algorithm for global optimization☆,☆☆

Na Lin[a], Luwei Fu[b,a], Liang Zhao[a,*], Ammar Hawbani[c], Zhiyuan Tan[d], Ahmed Al-Dubai[d], Geyong Min[e]

[a] School of Computer Science, Shenyang Aerospace University, Shenyang, China
[b] School of Computer Science and Engineering, University of Electronic Science and Technology of China, China
[c] School of Computer Science and Technology, University of Science and Technology of China, China
[d] School of Computing, Edinburgh Napier University, UK
[e] Department of Computer Science, College of Engineering, Mathematics, and Physical Science, University of Exeter, UK

ARTICLE INFO

Keywords:
Nomad algorithm
Nature-inspired algorithm
Optimizer
Function optimization
Global search

ABSTRACT

Nature-inspired computing (NIC) has been widely studied for many optimization scenarios. However, miscellaneous solution space of real-world problem causes it is challenging to guarantee the global optimum. Besides, cumbersome structure and complex parameters setting-up make the existed algorithms hard for most users who are not specializing in NIC, to understand and use. To alleviate these limitations, this paper devises a succinct and efficient optimization algorithm called Nomad Algorithm (NA). It is inspired by the migratory behavior of nomadic tribes on the prairie. Extensive experiments are implemented with respects to accuracy, rate, stability, and cost of optimization. Mathematical proof is given to guarantee the global convergence, and the nonparametric tests are conducted to confirm the significance of experiment results. The statistical results of optimization accuracy denote NA outperforms its rivals for most cases (23/28) by orders of magnitude significantly. It is considered as a promising optimizer with excellent efficiency and adaptability.

1. Introduction

The natural world has gone through a long evolution and shown strong adaptability. According to the observation of natural systems, many ingenious mechanisms are learned which help human beings construct models to solve complex real-world problems. This promotes great success in the recent invention of nature-inspired optimization algorithms which mimic natural phenomena or biological behaviors to solve the function optimization problem. Recently, it has been widely utilized in many real-world applications including intelligent transportation, automated manufacturing, feature selection, pattern classification, etc [1,2].

In general, for a target problem to be solved, an evaluation function could be formulated to calculate the profit or cost of a scheme (i.e., a potential solution). The goal of optimization is to locate the best solution of the evaluation function, which leads to the

☆ This paper is for special section VSI-bioc. Reviews were processed by Guest Editor Dr. Xiaochun Cheng and recommended for publication.

☆☆ This paper is supported in part by the Key Projects of Liaoning Provincial Department of Education Science Foundation, China under Grant L201702, the National Natural Science Foundation of China under Grant 61701322, the Young and Middle-aged Science and Technology Innovation Talent Support Plan of Shenyang, China under Grant RC190026, the Natural Science Foundation of Liaoning Province, China under Grant 2020-MS-237, and the Liaoning Provincial Department of Education Science Foundation, China under Grant JYT19052.

* Corresponding author.
E-mail address: lzhao@sau.edu.cn (L. Zhao).

https://doi.org/10.1016/j.compeleceng.2022.107862
Received 21 October 2020; Received in revised form 22 February 2022; Accepted 23 February 2022
Available online 17 March 2022

maximum profit or minimum loss. Faced with the vast number of variables and operations, conventional deterministic optimization techniques only guarantee the local optimum and become extremely time-consuming with the increasing complexity of problems [3]. However, the best solution calling for global optimum and time-efficiency is actually the most essential for an optimization algorithm.

Since the proposal of Genetic Algorithms (GA) [4], the above shortcomings have been alleviated by the stochastic algorithms. As a typical case of NIC algorithm, GA mathematically imitates the natural evolution process to seek the global optimum by heuristic information. The optimization mechanism described by gene behaviors of selection, reproduction, and mutation are stochastic and determined by probability. Moreover, there is no requirement for the derivation or gradient information of the problem. This property makes the algorithm efficient in solving complicated problems with an unknown high-dimensional search space. These novel techniques attracted more researchers devoting to the developments of the stochastic algorithm. Subsequently, inspired by the social behaviors of bird swarms, the most famous Particle Swarm Optimization (PSO) was proposed [5]. It has evolved from its initial design into substantial variations taking various forms [6]. Since a great success achieved by PSO, the number of newly proposed nature-inspired algorithms simulating natural behaviors have seen substantial growth over recent years.

Although these existing algorithms show a promising development, it is neglected that the users of algorithms are usually in diverse fields rather than specializing in Computational Intelligence (CI). Most users just employ the algorithm as a black box to optimize their particular application. As a result, it is hard to fully exploit the performance of a NIC optimizer because of the following two hinders:

- *Improper parameter setting*: the efficiency of an optimization algorithm significantly relies on the parameters setting which directly influences the balance of local exploiting and global exploring. The complicated computation with an ambiguous setting of parameters results in that many excellent algorithms only be perfectly performed by CI researchers, while the real users (such as engineers, economists) cannot make full use of these algorithms.
- *Redundant operation*: some algorithms over-imitate the natural phenomena or biological behaviors [7,8], that introduce unnecessary parameters and operations. It complicates the computation but makes little improvement to an algorithm.

In the light of above discussion, this work aim to develop a succinct and efficient NIC algorithm for global optimization of complex functions. The main contributions are summarized below.

- To address the aforementioned deficiencies, we firstly propose a novel light-weight nature-inspired optimization algorithm, named Nomad Algorithm (NA), which is inspired by the migration of the nomadic tribe. NA leverages a succinct mechanism to achieve an adaptive balance of local exploitation and global exploration. Compared with the existing algorithms, NA is easy to understand which could avoid improper setting and keep reliably high accuracy.
- To analyze the technical aspects of NA, (1) the mathematical proof is presented to guarantee the global convergence; (2) benchmark tests are implemented to confirm the parameters setting; (3) comparative experiments with five rivals are implemented to verify the superiority of NA; (4) nonparametric test are conducted to analyze the significance of statistical results. The experiment results demonstrate that this light-weight algorithm has significant advantages over its rivals in both convergence accuracy and rate.

The rest of this paper is organized as follows. Section 2 reviews the related work of NIC algorithms studied in the field of function optimization. Section 1 outlines the proposed Nomad Algorithm and gives the proof of convergence. In Section 4, adequate experiments and tests have been taken to evaluate NA. Finally, we conclude our work and suggest some directions of future research in Section 5.

2. Literature review

Based on the inspiration sources, these nature-inspired optimizers are commonly classified into three categories: natural phenomena (evolution)-based, biology-based, and physics-based algorithms.

Natural phenomena-based algorithms simulate the evolutionary process in nature. The GA is the earliest and most famous one among many [4]. For more instances, Invasive Weed Optimization (IWO) mimics the weed colonizing in a new environment [9]; Water Cycle Algorithm (WCA) is based on the natural cycle process that rivers and streams flow to the sea [10]. Other well-known natural phenomena-based algorithms are Slime Mould Algorithm (SMA) [11], Flower Pollination Algorithm (FPA) [12], Differential Evolution (DE) [13], and Gaining-Sharing Knowledge algorithm (GSK) [14].

Among these nature-inspired optimizers, some algorithms are inspired by the biological behaviors. For example, by imitating the bee colony seeking for honey, Artificial Bee Colony (ABC) algorithm was proposed to global optimization of search space [15]. Inspired by the bubble-net feeding method of humpback whales, Whale Optimization Algorithm (WOA) was presented [7]. In conclusion of the recent study of biology-based algorithms, the predation and movement behaviors of animal population are the main inspiration source. The most representative ones are Horse herd Optimization Algorithm (HOA) [16], Salp Swarm Algorithm (SSA) [17], and Manta Ray Foraging Optimization (MRFO) [18], Falcon Optimization Algorithm (FOA) [19].

Other heuristics are inspired by physics. The Simulated Annealing (SA) derives from the principle of solid annealing [20], which is the most cited physics-based algorithm. For another typical instance, Gravitational Search Algorithm (GSA) is proposed on the law of universal gravity [21]. Other well-known algorithms of this category includes Fireworks Algorithm (FWA) [8], Archimedes Optimization Algorithm (AOA) [22], Water Wave Optimization (WWO) [23], and Chemical Reaction Optimization (CRO) [24].

The features of the mentioned algorithms are listed in Table 1. More algorithms denoting state of the art in the field of NIC are outlined by literature [1], it also identifies open challenges in detail.

Table 1
The features of related algorithms.

Algorithm	Inspiration source	Proposal year
SA [20]	Solid annealing	1983
GA [4]	Gene evolution	1992
PSO [5]	Bird flock	1995
IWO [9]	Weed colonization	2006
ABC [15]	Honey bee	2007
GSA [21]	Law of gravity	2009
WCA [10]	Water cycling	2012
FPA [12]	Flower Pollination	2014
WWO [23]	Shallow water wave	2015
WOA [7]	Whale predation	2016
SHO [25]	Predatory interactions	2017
CRO [24]	Chemical reaction	2017
DE [13]	Biological evolution	2018
FWA [8]	Fireworks explosions	2018
SOA [26]	Migration of seagull	2019
SMA [11]	Oscillation of slime mold	2020
MRFO [18]	Manta rays	2020
FOA [19]	Hunt behavior of falcons	2020
GSK [14]	Knowledge acquisition of human	2021
SSA [17]	Salp swarm	2021
AOA [22]	Principle of buoyant force	2021
HOA [16]	Horses' herding	2021

The growing interest on optimization promotes a vast number of optimizers including the variations and hybrids of existing algorithms [27]. However, according to the 'No Free Lunch (NFL)' theory, when an algorithm shows perfect effectiveness on some problems, poor effectiveness is bound to happen on some other problems. In other words, a 'good' algorithm (i.e., be able to obtain the global optimum with the highest accuracy and minimal consumption of time and space) is just good at some specific problems; algorithms are problem-oriented. Therefore, it is still necessary to develop new algorithms. The purposes of new algorithms are: handling the specific problems which are unsolvable for previous algorithms, and optimizing most problems with better performance.

3. Methodology of Nomad Algorithm (NA)

Nomadic tribes are known for their migration, who always search for and migrate to a better area with lush pasture. Fig. 1 demonstrates how a nomadic tribe works. As shown, a nomadic tribe consists of many members which are divided into herdsmen or rangers. Once the tribe arrives at a new region: (1) herdsmen work and search around the tribe within a range, which is determined by the fitness of the environment; (2) For avoiding stagnation and exhaustion of resources, some individuals are selected as Rangers to quickly explore a farther area for a better environment. The purpose of a tribe and its members is to find a better region with more resource and migrate to here. These simple mechanisms performing efficient searching will be mathematically modeled to devise an intelligent optimizer, NA.

3.1. Design of NA

In the design of NA, the migrating of a nomad tribe can be considered as the searching and optimizing process. The most livable place represents the best solution, and the vast prairie is viewed as a search space. Each member of the tribe will work as either Herdsmen or Rangers. The former are responsible for local exploiting and the latter are responsible for global exploring. Based on the description above, the NA consists of Herdsmen grazing, Rangers exploring, and tribe updating.

A tribe is always located at the most livable position found, i.e., the current optimal solution. The tribe's coordinates are defined as a d-dimensional vector $X_{tribe} = \{x^1, x^2, \dots, x^d\}$ where each element indicates a variable to be optimized of an objective function. X_{tribe} is randomly generated at initialization of NA and updated with iteration. The Herdsmen graze around the tribe with a dynamic radius. The grazing radius is calculated as follows:

$$R_a(n) = \begin{cases} X_{up} - X_{low} & n = 1 \; or \; 2 \\ R_a(n-1) \times \alpha & f_{n-1} < f_{n-2} \\ R_a(n-1) \times \beta & f_{n-1} = f_{n-2} \end{cases} \tag{1}$$

where $R_a(n)$ is the grazing radius at iteration n. The lower and upper bounds of the searching space are represented as X_{low} and X_{up}. α is an enlarging factor greater than 1, while β is a reducing factor in the range of $(0, 1)$. Too big or too small values will result in dramatic fluctuations and break the stability of searching. Through experimental verification, it is found when α is 1.1 and β is 0.9 could guarantee reliable efficiency for most problems. f_n represents the fitness of X_{tribe} at iteration n, i.e. the value of the current optimum.

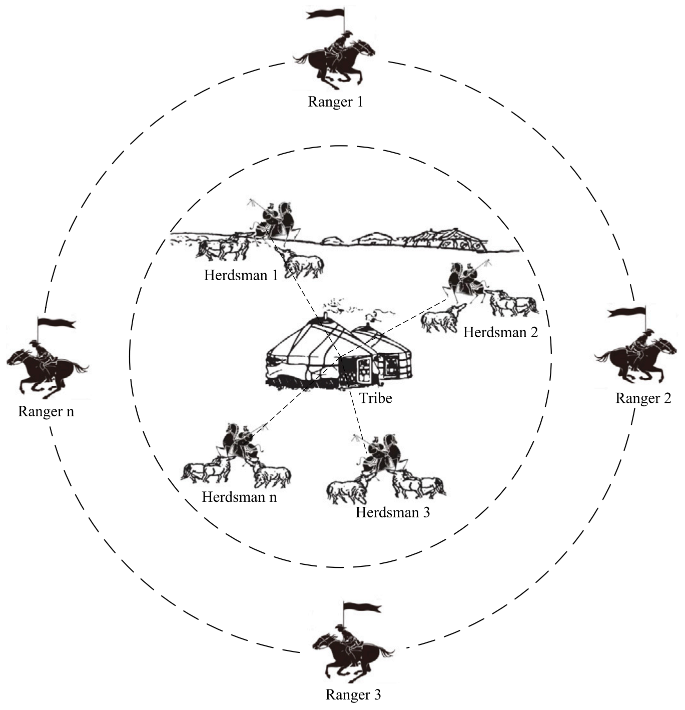

Fig. 1. The working of a nomadic tribe.

Eq. (1) means that the exploiting scale should be as big as the whole search space in initialization. If the tribe finds a better territory where is more livable, and Herdsmen are supported to graze on broader space; the graze radius should be enlarged in subsequent iteration to further exploit a wider space. In contrast, if the NA cannot find a better solution in the current iteration, Herdsmen have to reduce their live range to make the search more detailed in the next iteration. These strategies provide more probability to find a better solution and accelerate convergence.

The Herdsmen grazing behavior, representing local exploiting of NA, as mentioned above is given by Algorithm 1.

Algorithm 1: Herdsmen grazing

```
   Input: The current iteration times $n$, coordinate of tribe $X_{tribe}$ and its fitness $f_n$;
   Output: New coordinates of Herdsmen;
1  Calculate the grazing radius $R_a(n)$ by Eq. 1;
2  for each Herdsmen do
3      for each dimension do
4          $X^k = {X_{tribe}}^k + \text{Rand}(-R_a(n), R_a(n))$
           //Avoid searching beyond boundaries;
5          if $X^k < {X_{low}}^k$ or $X^k > {X_{up}}^k$ then
6              $X^k = \text{Rand}({X_{low}}^k, {X_{up}}^k)$;
7          end
8      end
9  end
```

Once the current grazing radius is obtained by step 1 of algorithm, each herdsman updates its coordinate by step 3 to step 8. Specifically, step 4 determines the coordinate of each dimension, and step 5 to step 7 could recall herdsmen who are beyond the boundaries. ${X_{low}}^k$ and ${X_{up}}^k$ represent the lower bound and upper bounds of kth dimension in potential space. Function Rand(A, B) is used to generate a random number uniformly distributed between A and B. Herdsmen will exploit a vicinity of tribe X_{tribe} randomly within a corresponding radius $R_a(n)$.

Herdsmen grazing process enables flexible exploitability of the NA at the vicinity of the present best solution. However, it is not enough to make the tribe find the most livable place. Somewhere better than other adjoining areas but is not the optimum in whole research space, which is called local optimum. To prevent the tribe from stagnating at local optimum, Rangers are dispatched to explore remoter areas. The Rangers have global search capability and move in the entire space depending on the current coordinate and migration result of the tribe. This behavior of Rangers can be briefly represented as follows:

$$X_{Rangers} \sim \mathrm{N}(X_{tribe}, \sigma^2(n)) \tag{2}$$

The coordinates $X_{Rangers}$ to be explored by Rangers accord with a Gaussian distribution $\mathrm{N}(X_{tribe}, \sigma^2(n))$ with mean value X_{tribe} and standard deviation $\sigma(n)$. While $\sigma(n)$ determines the amplitude of Rangers exploring at iteration n. This parameter is calculated by the following equation:

$$\sigma(n) = \begin{cases} X_{up} - X_{low} & n = 1 \ or \ 2 \\ \sigma(n-1) \times 0.5 & f_{n-1} = f_{n-2} \\ X_{up} - X_{low} & f_{n-1} < f_{n-2} \end{cases} \tag{3}$$

This process can be abstracted as a Gaussian probability sampling. At the beginning, exploring amplitude σ fills the whole search space. During the computation process, this parameter will rapidly reduce if no better solution found at the previous searching. Once the ranger finds somewhere with better fitness, the exploring amplitude σ returns to the original range. Diverse from exploitation employs the uniform distribution within a certain range, Gaussian distribution has more focus on the closer region, while also keep probability to search anywhere in the search space.

The above process is represented by the pseudocode in Algorithm 2. Function $\mathrm{N}(A, B)$ could generate a random number based on Gaussian distribution whose mean value is A and variance is B.

Algorithm 2: Rangers exploring

```
   Input: The current iteration times $n$, coordinate of tribe $X_{tribe}$ and its fitness $f_n$;
   Output: New coordinates of Rangers;
1  Calculate the exploring amplitude $\sigma(n)$ by Eq. 3;
2  for each Ranger do
3      for each dimension do
4          $X^k = {X_{tribe}}^k + \mathrm{N}(0, \sigma^2(n))$;
           //Avoid searching beyond boundaries;
5          if $X^k < {X_{low}}^k$ or $X^k > {X_{up}}^k$ then
6              $X^k = \mathrm{Rand}({X_{low}}^k, {X_{up}}^k)$;
7          end
8      end
9  end
```

Once the exploring amplitude of the current iteration is obtained by step 1, each Ranger updates its coordinate by step 3 to step 8. Leveraging Gaussian sampling, a series of new positions are explored by step 4. Step 5 to step 7 are responsible for mapping the out-of-bounds nodes back to potential space. This strategy makes full use of the heuristic information. The range and intensity of exploration are dynamically determined by the current optimum and search process.

The proportion of the two types of searching will determine the tradeoff between local exploitation and global exploration. Under a fixed population size, the nomadic tribe should select more Rangers for exploring to avoid local optimum. In contrast, if they find a promising area, more Herdsmen should be employed for local intensification. We assume the nomadic tribe can learn helpful information from their previous migration to balance the proportion of Herdsmen and Rangers adaptively.

When the nomadic tribe has stagnated for γ iterations, the probability P of increasing Rangers in the next iteration is defined as below:

$$P = 1 - exp\left(-\left(\frac{\gamma}{\lambda I_{max}}\right)^2\right) \tag{4}$$

where I_{max} is the maximum iterations. λ is a sensitivity coefficient in range of (0, 1] to determine the changing rate of probability, in which a smaller value corresponds to a sharper change. A tribe consists of total M members is adaptively organized as Algorithm 3. H_{min} and H_{max} represent the minimum and maximum proportion of Herdsmen in the tribe, respectively. M_H denotes the number of Herdsmen while M_R is the number of Rangers.

In Algorithm 3, the stagnation iteration γ is set to 0 at the beginning and increased by 1 once the current iteration of searching did not find a better solution (shown as step 1 to step 2). Then a Herdsman turns into Ranger by probability P which is calculated by γ and Eq. (4) (shown as step 3 to step 7). Once a better solution is found, γ is reset to 0 and a Ranger turns into Herdsman (shown as step 8 to step 13). Given the population size and the current number of herdsmen, the number of Rangers could be calculated accordingly (shown as step 14). Note that the switching of Herdsmen Rangers should satisfy the pre-defined proportion bound (judge by step 4 and step 10).

3.2. Summary of NA

Based on the above definitions, the architecture of NA is established. It consists of Herdsmen grazing, Rangers exploring and tribe updating, which correspond to local search, global search, and updating of solution and search strategy. The complete workflow is

```
Algorithm 3: Population balancing
   Input: The maximum iterations $I_{max}$, current stagnation iterations $\gamma$, current best fitness $f_n$;
   Output: The number of Herdsmen $M_H$ and Rangers $M_R$;
 1 if $f_n = f_{n-1}$ then
 2     $\gamma = \gamma + 1$;
 3     Calculate the probability $P$ by Eq. 4;
 4     if Rand(0, 1) $< P$ and $M_H > MH_{min}$ then
 5         $M_H = M_H - 1$;
 6         $\gamma = 0$;
 7     end
 8 else
 9     $\gamma = 0$;
10     if $M_H < MH_{max}$ then
11         $M_H = M_H + 1$;
12     end
13 end
14 $M_R = M - M_H$;
15 return $M_H$ and $M_R$;
```

```
Algorithm 4: Nomad Algorithm
   Input: Bound of search space, objective function, population size, dimensions, maximum iterations;
   Output: The best solution and its fitness;
 1 Randomly initialize $X_{tribe}$ of NA;
 2 repeat
 3     for each tribe do
 4         Herdsmen grazing by Algorithm 1;
 5         Rangers exploring by Algorithm 2;
 6         Evaluate each member's fitness;
 7         Update $X_{tribe}$ and $f_n$ by the best member;
 8         Update $M_H$ and $M_R$ by Algorithm 3;
 9     end
10 until Achieve the expected precision or maximum iterations;
11 return $X_{tribe}$ and its fitness $f_n$ ;
```

defined as Algorithm 4. Following the algorithm, the tribe moves to the best position X_{tribe} in each iteration; they ultimately migrate to the most livable position with the best fitness, i.e. the best solution and optimum.

The computational complexity of the proposed NA can be defined as $O\left(I_{max} \times \left(M \times \left(C_{eva} + C_{compare} + C_{update}\right)\right)\right)$, where M denotes the population size of the tribe in which all members are evaluated during each generation, given the algorithm ends at the I_{max}th generation. C_{eva} denotes the time complexity of evaluating a solution by objective function, which is correlated to the dimensions d. $C_{compare}$ denotes the time complexity of comparing the fitness of a solution with the present best solution. This complexity is a constant. C_{update} denotes the time complexity of updating a member (search agent) of tribe, which is correlated to the dimensions d. Therefore, the time complexity of NA can be transformed to $O\left(I_{max} \times (M \times (O(d) + O(d) + O(1)))\right)$, and further simplified to $O(I_{max} \times M \times d)$.

The proposed NA employs two kinds of search agents, Herdsmen and Rangers, to implement the local exploitation and global exploration, respectively. The superiorities and differences between NA and other similar algorithms are summarized below.

(1) Some algorithms update agents according to other agents' locations [5,7]. In the later execution, the population will converge and lose the diversity while each agent can never escape from the local optimum. However, the migration of NA enables the population to move dramatically to the global optimum all the time.

(2) Some algorithms have specific strategies or agents for global search, but only with a mutation according to a potential location and a pre-defined distribution [8,12]. These mutation strategies do not provide strong logic where they do not make more sense than a random selection in search space. In NA, the mutation strategy relies on the state of search, which can adaptively adjust the distribution of mutation.

(3) Some algorithms employ agents specifically for exploration but too simple without any heuristic information or showing intelligence [15]. In contrast, the global agents of NA leverage much useful information include the location of the current optimum, stagnation iterations, and change of search result, to seek the global optimum sensibly.

(4) Both Rangers and Herdsmen of NA work with a very simple searching strategy. The adaptive tuning of parameters further simplifies the algorithm initialization as well as makes search fit to solution space. Employing these two types of agents and adaptive transferring of them, NA performs a great efficiency without cumbersome computation and setting.

3.3. Analyses and proof of global convergence

For a new algorithm, the proof of global convergence must be discussed. As we know, the ultimate convergence cannot be guaranteed by many algorithms, although they have global mechanisms. Some exploration strategies rely on the locations of search agents, which have only minor mutations and risk to stagnate in the later iterations because the search agents usually converge to a narrow area at the end. Thus, this section will analyze the search process and prove the convergence of NA.

In the stochastic model of NA, the essential infimum α of f on space S is defined as the following equation:

$$\alpha = inf(n : v[x \in s|f(x) < t] > 0) \tag{5}$$

that $v[A]$ is the Lebesgue measure of the potential space A. The stochastic processes of NA are represented by a discrete sequence of solution as $\{x_n\}_{n=0}^{\infty}$, where x_n is the optimum maintained in the nth iteration. Since the optimum is difficult to be exactly located by a stochastic algorithm in practice, the vicinity of the optimum within an acceptable range is generally considered as the global optimum. In this way, the continuous problems with their infinite size search space can be viewed as a finite number of discrete spaces. Also, the optimization process can be model as a discrete sequence with finite states. Referring to convergence theorem of global search [28], the definition of convergence about NA is given as below.

Condition 1. For an absorbing Markov process $\{x_n\}_{n=0}^{\infty}$ and an optimal state space $Y^* \subset Y$, $\varphi(n) = P\{x_n \in Y^*\}$ indicates the probability of reaching the optimal state at the time of n. If $\lim_{n\to\infty} \varphi(n) = 1$, $\{x_n\}_{n=0}^{\infty}$ converges to global optimum.

Theorem 1. *Stochastic process of NA is an absorbing Markov process.*

Proof of Theorem 1. $\{x_n\}_{n=0}^{\infty}$ is a discrete stochastic process, $\{x_n\}$ is determined by $\{x_{n-1}\}$, i.e., $P\{x_{n+1}|x_1, x_2, \dots, x_n\} = P\{x_{n+1}|x_n\}$. It means that the state of time $n+1$ is only related to state of time n, therefore, $\{x_n\}_{n=0}^{\infty}$ is a Markov process.

If x_n located at optimum solution space, there is $x_n \in Y^*$. Since the NA only remain the current best solution as X in the next generation, x_{n+1} cannot be worse than x_n, there must be $x_{n+1} \in Y^*$, i.e. $P\{x_{n+1} \notin Y^*|n_t \in Y^*\} = 0$. Hence the stochastic processes of NA, $\{x_n\}_{n=0}^{\infty}$, is an absorbing Markov process.

Theorem 2. *For an absorbing Markov process $\{x_n\}_{n=0}^{\infty}$ and an optimal state space $Y^* \subset Y$, for $\forall n$, if $1 \geqslant P\{x_n \in Y^*|x_{n-1} \notin Y^*\} \geqslant d \geqslant 0$ and $P\{x_n \in Y^*|x_{n-1} \in Y^*\} = 1$, then $P\{x \in Y^*\} \geqslant 1 - (1-d)^n$.*

Proof of Theorem 2. Assumed $n = 1$, there are:

$$\begin{aligned}
&P\{x_1 \in Y^*\}\\
=&P\{x_1 \in Y^*|x_0 \in Y^*\}P\{x_0 \in Y^*\} + P\{x_1 \in Y^*|x_0 \notin\\
&Y^*\}P\{x_0 \notin Y^*\}\\
\geqslant&P\{x_0 \in Y^*\} + dP\{x_0 \notin Y^*\}\\
=&P\{x_0 \in Y^*\} + d(1 - P\{x_0 \in Y^*\})\\
=&d + (1-d)P\{x_0 \in Y^*\}
\end{aligned}$$

Since $(1-d) > 0$, then $d + (1-d)^1 P\{x_0 \in Y^*\} \geqslant d$, and $P\{x_1 \in Y^*\} \geqslant d = 1 - (1-d)^1$.

Assumed $\forall n < k-1$, $P\{n_t \in Y^*\} \geqslant 1 - (1-d)^n$. Then $n = k$ can satisfy follows:

$$\begin{aligned}
&P\{x_k \in Y^*\}\\
=&P\{x_k \in Y^*|x_{k-1} \in Y^*\}P\{x_{k-1} \in Y^*\} + P\{x_k \in Y^*|\\
&x_{k-1} \notin Y^*\}P\{x_{k-1} \notin Y^*\}\\
\geqslant&P\{x_{k-1} \in Y^*\} + dP\{x_{k-1} \notin Y^*\}\\
=&P\{x_{k-1} \in Y^*\} + d(1 - P\{x_{k-1} \in Y^*\})\\
=&d + (1-d)P\{x_{k-1} \in Y^*\}\\
\geqslant&d + (1-d)(1-(1-d)^{k-1})\\
=&1-(1-d)^k
\end{aligned}$$

Based on the above deduction, $P\{x \in Y^*\} \geqslant 1 - (1-d)^n$ can be satisfied for any $n \geqslant 1$. Theorem 2 has been proofed by mathematical induction.

Theorem 3. *NA will converge to global optimum with probability 1.*

Proof of Theorem 3. The exploring process of NA is a Gaussian mutation but constrained by bounds of search space. For the sake of simplicity, this process is regarded as a uniform random sampling. $P_{Re}(n)$ indicates the probability of transferring to optimal state

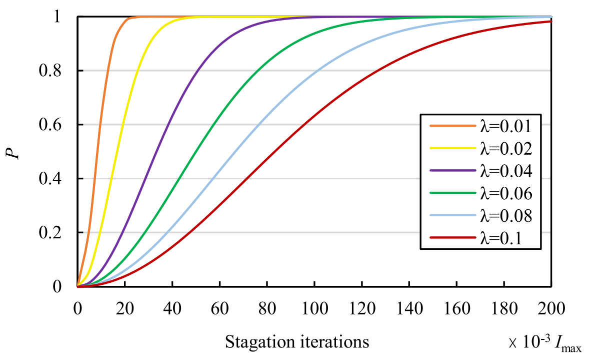

Fig. 2. Probability curves of different λ with growing iterations.

space Y^* from non-optimal space by Rangers exploring, i.e.

$$P_{Re}(n) = \frac{v(Y^*)M_R}{v(S)} \tag{6}$$

where $v(S)$ represents the Lebesgue measure of the potential space S. It is obviously greater than 0.

Due to $v(Y^*) > 0$, there is $P_{Re} > 0$.

The Markov stochastic process of NA, $\{x_n\}_{n=0}^{\infty}$, satisfies following equation:

$$\varphi(n) = P\{x_n \in Y^* | x_{n-1} \notin Y^*\} \geqslant P_{Re}(n) + P_{Hg}(n) \tag{7}$$

$P_{Hg}(n)$ represents the probability of reaching optimal state space Y^* from non-optimal space by Herdsmen grazing at n generation. Therefore, $P\{x_n \in Y^* | x_{n-1} \notin Y^*\}$. According to Theorems 1 and 2, there is

$$\varphi(t) = P\{x_n \in Y^*\} \geqslant 1 - (1 - P_{Re}(n))^n \tag{8}$$

i.e. $\lim_{n\to\infty} \varphi(n) = 1$, Condition 1 is satisfied. The Markov process of NA, $\{x_n\}_{n=0}^{\infty}$, converges to the global optimum with probability 1.

4. Experiments and analyses

This section studies the control parameters and compares NA with other classical optimizers to verify the efficiency of Nomad Algorithm (NA). All the experiments are implemented by Matlab R2016b software on the environment of Windows 10 (64bits), 4 GB RAM, and 3.2 GHz CPU (Core I5-3470).

4.1. Test functions setting

According to the common evaluation process in this field, CEC 2013 benchmark functions are employed to evaluate NA and compares it with several well-known optimizers [29]. There are total 28 functions in CEC 2013 including 5 unimodal ($f_1 - f_5$), 15 multimodal functions ($f_6 - f_{20}$), and 8 composition functions ($f_{21} - f_{28}$).

For convenience, the same bound $[-100, 100]^D$ is set for all cases, in which D represents the function's dimensions, i.e., the number of variables that determine the result of a problem. The mathematical formulations and other features of these functions are given in detail in the literature of problem definitions [29]. For the sake of intuitive comparison, some functions are shifted to make the optimal value (i.e. f_{min}) of all functions are 0. Thus, the optimum found by the algorithm can be regarded as the value of error.

The unimodal functions ($f_1 - f_5$) without local optimum examines the exploitability of algorithms, while the multimodal functions ($f_6 - f_{20}$) have quite a number of local optima examines the global ability of local optima avoidance. Eventually, the remaining 8 composition functions ($f_{21} - f_{28}$) are integration of different multimodal functions with rotation and shift. Thus, these scenarios are very challenging for optimization algorithms that the capability and balance of the local and global search are both significantly considered during the text.

4.2. Investigation of control parameters

The performance of an algorithm significantly relies on parameter tuning. Besides the common parameters (population size and maximum iterations) determined by the problem, the sensitivity factor λ is the key and specific parameter of the proposed NA. A suitable value should be set to make NA adapt to most optimization problems.

With the maximum iterations I_{max}, the changes of P (probability of increasing a Ranger) versus continuous stagnation iterations γ under the different sensitivity coefficient λ are depicted in Fig. 2. These curves mean that P will grow significantly when NA meets

Table 2
Statistical mean (STD) and average rankings of NA with different λ.

	$\lambda = 0.01$	$\lambda = 0.02$	$\lambda = 0.04$	$\lambda = 0.06$	$\lambda = 0.08$	$\lambda = 0.1$
	Mean (STD)	Mean (STD)	Mean (STD)	Mean (STD)	Mean (STD)	Mean (STD)
f_1	1.60E−13 (4.19E−13)	3.22E−14 (3.66E−14)	9.31E−14 (1.95E−13)	4.33E−14 (7.67E−14)	**2.80E−14 (5.30E−14)**	4.80E−14 (8.32E−14)
f_2	3.23E+06 (1.32E+06)	3.13E+06 (1.87E+06)	**2.98E+06 (1.41E+06)**	3.35E+06 (1.27E+06)	3.40E+06 (1.48E+06)	3.14E+06 (1.62E+06)
f_3	6.17E+08 (8.57E+08)	**2.37E+08 (2.64E+08)**	6.52E+08 (9.02E+08)	8.49E+08 (1.49E+09)	5.02E+08 (5.37E+08)	3.70E+08 (3.20E+08)
f_4	48 553.59 (18 543.16)	41 700.15 (9479.251)	40 046.26 (8374.235)	39 727.55 (15 553.63)	42 987.12 (13 019.23)	**38 972.36 (16 204.11)**
f_5	0.000647 (0.000216)	0.000715 (0.000439)	0.000756 (0.000500)	**0.000487 (0.000290)**	0.000710 (0.000413)	0.000615 (0.000311)
f_6	28.98278 (21.89594)	30.99343 (25.50985)	35.73462 (27.92715)	**28.85264 (26.39833)**	31.68720 (25.29552)	28.92662 (25.12247)
f_7	177.1652 (59.74837)	**154.3019 (32.20803)**	155.8607 (63.28115)	168.8776 (49.86577)	178.9854 (60.65222)	162.9958 (33.12783)
f_8	21.06772 (0.087273)	21.02788 (0.079610)	**21.01545 (0.094226)**	21.05489 (0.065534)	21.02017 (0.094009)	21.02835 (0.062685)
f_9	32.40667 (3.229968)	33.40594 (4.574122)	33.10223 (4.258095)	32.34457 (4.065964)	32.80885 (3.861020)	**32.21077 (3.841808)**
f_{10}	0.073698 (0.039290)	0.063439 (0.038313)	0.068884 (0.043761)	0.067283 (0.032312)	**0.061131 (0.035597)**	0.071473 (0.036503)
f_{11}	**7.090203 (3.308869)**	9.719774 (3.675924)	10.56439 (4.924078)	9.51979 (3.118230)	10.19970 (3.420520)	11.153784 (4.521600)
f_{12}	235.7676 (82.92181)	**194.7423 (53.44140)**	252.2277 (93.47301)	237.1672 (92.24124)	219.2713 (67.06527)	200.5025 (61.31060)
f_{13}	**290.0314 (67.08325)**	342.4718 (73.62129)	315.1089 (82.73770)	319.7689 (102.7610)	321.2035 (89.95275)	321.0533 (76.77332)
f_{14}	**390.9203 (180.6567)**	456.2020 (320.2956)	662.8712 (233.6292)	687.3736 (270.0346)	674.2895 (218.9758)	704.6007 (250.6285)
f_{15}	4896.996 (824.6379)	4746.159 (735.8666)	4962.069 (1049.190)	**4654.465 (569.5511)**	4725.479 (623.2614)	4694.472 (872.4831)
f_{16}	0.903856 (0.350875)	1.012645 (0.449355)	1.367546 (0.728539)	0.940954 (0.441021)	**0.888194 (0.370080)**	1.054917 (0.615923)
f_{17}	**36.15671 (3.803661)**	37.42863 (9.196523)	39.78961 (4.822880)	40.07587 (2.811524)	38.27449 (6.708751)	41.66757 (4.812873)
f_{18}	281.7291 (75.87061)	**247.1401 (80.92806)**	268.4519 (114.6317)	253.1871 (73.52509)	263.2395 (70.21799)	276.9385 (86.62634)
f_{19}	**2.469993 (0.775375)**	2.749147 (0.530315)	3.047411 (0.710717)	3.028048 (0.754484)	2.805032 (0.631827)	2.904449 (0.513232)
f_{20}	14.66802 (0.475456)	14.56116 (0.828295)	14.87694 (0.218696)	14.57316 (0.846568)	14.52680 (0.631742)	**14.64418 (0.592347)**
f_{21}	**276.5316 (91.99527)**	313.7088 (75.62408)	294.3544 (65.12029)	310.8860 (89.04596)	300.8860 (100.7887)	308.7088 (79.77033)
f_{22}	**403.5741 (209.8814)**	454.1308 (201.9366)	722.4815 (242.9252)	677.4981 (251.2865)	619.8607 (195.5712)	579.2604 (264.1989)
f_{23}	5780.592 (1021.068)	5826.331 (884.5152)	5828.496 (895.7737)	5699.789 (890.0223)	**5555.922** (888.6475)	5771.998 (1013.188)
f_{24}	299.9569 (16.03353)	296.9968 (11.00429)	299.6100 (13.66878)	306.6948 (16.60983)	**294.1516 (12.73522)**	298.3698 (9.972044)
f_{25}	312.2500 (9.409537)	310.5078 (10.34371)	**306.9390 (13.23561)**	309.2323 (10.93587)	307.9417 (10.64837)	316.5928 (15.90595)
f_{26}	357.4716 (68.65072)	374.1987 (42.22614)	367.1785 (58.16725)	371.9610 (41.26123)	**354.8099 (67.11805)**	373.6162 (43.20230)
f_{27}	1184.870 (120.6620)	1197.660 (116.7902)	1186.339 (109.1804)	1180.538 (126.6832)	1152.145 (142.8823)	**1139.492 (108.4728)**
f_{28}	511.3326 (517.2004)	**300.0000 (0.000009)**	553.4594 (521.0763)	384.8744 (379.5698)	429.8189 (399.8157)	362.0477 (277.4857)
Rank	3.6	3.2	4.2	3.5	**3**	3.5

stagnations. Furthermore, with the growing number of stagnation iterations, the change rate of P becomes smooth and ultimately converges to 1. It is also seen that the larger λ leads to the smoother change of P. Considering the range and effect of λ, it is set to 0.01, 0.02, 0.04, 0.06, 0.08, and 0.1, respectively, to select the best one by experiments. With 50 independent runs of 1000 iterations, Table 2 gives the statistical values of mean and standard deviation (STD) of optimization error under different parameters. Due to the randomicity of the stochastic method, multiple independent executions of algorithm should be conducted. The average ranking is introduced as an evaluation criterion. Specifically, for each benchmark function, 6 different λ settings are ordered according to the optimization error and ranks are assigned accordingly. The average ranking of a setting on all functions is calculated as its final rank. The best result on each benchmark case is highlighted in bold.

It is seen that the control parameter is problem-oriented. When a set of parameter values perform excellently on a certain problem, it will be mediocre at some other problems. The 'best parameter' does not exist, and the parameter should be tuned according to a specific problem. Nevertheless, the frequent testing and tuning of parameters are impractical. The parameters with the minimal average ranking are generally considered as performing less error on most problems. Therefore, the sensitivity factor λ is set to 0.08 as the standard parameters of NA. Note that if the objective functions are strongly multimodal or extremely high dimensional, the population size should be appropriately enlarged to boost the optimization capability.

4.3. Comparison and discussion

Based on the above discussion and setting, the final form of NA is determined. In this part, NA is compared with five well-known algorithms: PSO [5], ABC [15], GSA [21], FPA [12], and WOA [7]. PSO and GSA are very famous and highly cited. The FPA, WOA, and ABC are recent popularity which are widely applied in various fields of optimization. The simulation platform is implemented by Matlab and can be found at https://github.com/luwei-fu/Nomad-Algorithm. All these algorithms are used for the application scenario of single-objective numerical optimization. Their constructions and operators are different, whereas their mechanisms are similar. On one hand, their constructions can be abstracted as sampling (searching) and updating (deciding). On the other hand, except the population size and maximum iterations are common operators for most algorithms, other operators have the same purpose which is tuning the local and the global search. Despite the discrepancy of details, these five algorithms with the same purpose and similar operators are compared with our proposal. The parameters are set according to their original literature shown in Table 3.

M represent the population size of the diverse algorithms, where the same value as the dimensions (i.e., 30 for each benchmark function) of the objective problem is set to NA. The w is the inertia factor and c are learning factors of PSO. N_O of ABC indicates the percentage of onlooker bees, while if a solution cannot be improved through $limit$ generations, it will be replaced by a new

Table 3
Parameters of NA, PSO, ABC, GSA, FPA, WOA.

Algorithm		Parameters	
NA	$M = 30$	$\lambda = 0.08$	
PSO	$M = 25$	$w = 0.7298$	$c_1 = c_2 = 0.9$
ABC	$M = 125$	$N_O = 50\%$	$limit = Dimensions$
GSA	$M = 50$	$G_0 = 100$	$\alpha = 20$
FPA	$M = 25$	$p = 0.8$	
WOA	$M = 30$	$p = 0.5$	

Table 4
Statistics of mean (STD) on unimodal functions.

	NA	PSO	ABC	GSA	FPA	WOA
f_1	**3.1E−14**	246.833	51 134.2	12 679.8	6961.81	761.501
	(7.1E−14)	(119.595)	(6835.41)	(2995.50)	(2744.48)	(470.082)
f_2	**3.5E+06**	6.2E+07	8.0E+08	1.9E+08	1.4E+07	1.1E+08
	(1.9E+06)	(2.2E+07)	(1.7E+08)	(9.4E+07)	(7.2E+06)	(3.8E+07)
f_3	**2.9E+08**	7.2E+09	1.6E+13	6.9E+15	2.6E+10	1.5E+11
	(3.8E+08)	(3.7E+09)	(2.7E+13)	(1.2E+16)	(6.5E+09)	(2.9E+11)
f_4	**4.3E+04**	5.3E+04	1.5E+05	7.0E+04	4.8E+04	1.2E+05
	(1.0E+04)	(1.6E+04)	(3.3E+04)	(5.3E+03)	(1.1E+04)	(4.5E+04)
f_5	**6.9E−04**	7.0E+01	1.6E+04	4.2E+03	5.4E+02	1.1E+03
	(3.4E−04)	(2.7E+01)	(2.9E+03)	(7.8E+02)	(2.1E+02)	(2.9E+02)

Table 5
P-values of the rank-sum test for unimodal functions.

	NA	PSO	ABC	GSA	FPA	WOA
f_1	N/A	3.02E−11	3.02E−11	1.07E−07	3.02E−11	3.02E−11
f_2	N/A	3.02E−11	3.02E−11	5.07E−10	7.11E−09	3.02E−11
f_3	N/A	3.02E−11	3.02E−11	3.02E−11	3.02E−11	3.02E−11
f_4	N/A	0.001597	3.02E−11	3.02E−11	0.006097	3.69E−11
f_5	N/A	3.02E−11	3.02E−11	3.02E−11	3.02E−11	3.02E−11

position. G_0 is the initial gravitation value and α is an unnamed constant of GSA. The parameters p in FPA and WOA represent the switching probability, which determines the global search of the algorithm.

Due to the different population sizes (i.e., search agents) of these 6 algorithms, the same number of iterations does not mean the same workload. Hence, the number of evaluations is applied as a criterion instead of the number of iterations. Once a member (i.e., a potential solution) of the population is calculated by the objective function, it is recorded as an evaluation. The dimensions of all functions are set to 30, meanwhile, the adequate 50 independent runs 30,000 evaluations are utilized for each of these algorithms to avoid contingency.

Table 4 summarizes the mean errors and standard deviations (STD) of the unimodal functions $f_1 - f_5$. The best performance of the distinct functions are highlighted in bold. Considering these two metrics only evaluate the overall performance of the algorithm, a Wilcoxon statistical test is conducted to confirm the significance of the experiment results. In Table 5, the p-values between each algorithm and the best results are recorded as metrics of significance. A significant difference between two sets of samples could be guaranteed if the p-value less than 0.05. The convergence curves are depicted in Fig. 3 to study the optimization process intuitively.

It is demonstrated that NA highly outperforms other algorithms on all the 5 unimodal functions in terms of optimization errors, as well as the minimum STDs of NA indicate its advance on stability. Although GSA obtains less STD than NA on f_4, it is stable on a terrible solution with the worse error. In Table 5, the p-values have further verified the optimization results of NA significantly outperform other counterparts on all unimodal scenarios of CEC 2013, which strongly proves the powerful exploitation of NA in statistics. For the convergence process, the curves of others in Fig. 3 are relatively slow or have stagnated. In contrast, the convergence curves of NA demonstrate a decreasing trend and they are promising to converge to the smaller error with further iterations. All these superiorities are explained by the Herdsmen grazing strategy of NA, where a sufficient number of Herdsmen and the adaptive range of search can help the population to operate an intensification of local exploitation.

In terms of multimodal functions, $f_6 - f_{20}$, the statistical results are given in Tables 6 and 7 under the same simulation environment. These results denote NA achieves the minimum errors on 12 scenarios of the total 15 multimodal functions, apart from f_7, f_{13}, and f_{20}. According to the problem definitions of these particular cases, on the one hand, f_7 and f_{20} are variations of Schaffer function which are extremely multimodal. These huge numbers of local optima are very like each other but different from the global optima, which provides little useful information for NA during the search process. However, the search strategies of NA (including adaptive changes of agent proportions, local search range, and global exploring amplitude) rely on the heuristic information significantly. It leads to the poor performance of NA on Schaffer functions. On the other hand, f_{13} is a non-continuous

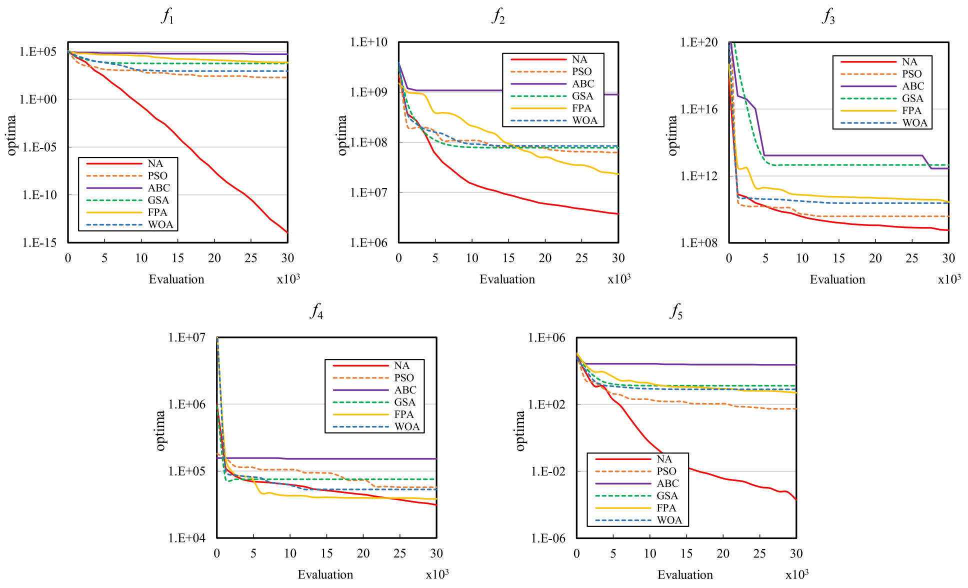

Fig. 3. Convergence curves on unimodal benchmark functions $f_1 - f_5$.

Table 6
Statistics of mean (STD) on multimodal functions.

	NA	PSO	ABC	GSA	FPA	WOA
f_6	**34.2993** **(26.0086)**	192.059 (30.8947)	7238.00 (1384.74)	294.214 (215.277)	411.024 (116.258)	300.583 (77.8251)
f_7	172.146 (66.8715)	**95.5420** **(25.5302)**	4210.69 (5804.13)	591895 (571714)	172.460 (31.3911)	4494.36 (12457.2)
f_8	**21.0213** **(0.08166)**	21.0598 (0.03408)	21.0436 (0.04975)	21.0694 (0.04076)	21.0310 (0.04843)	21.0561 (0.04000)
f_9	**32.3287** **(3.78377)**	32.5353 (3.75555)	41.7355 (1.37428)	40.9172 (2.79008)	35.4161 (1.08368)	38.7293 (2.53089)
f_{10}	**0.07822** **(0.04085)**	221.682 (75.7126)	7240.36 (795.397)	2128.04 (421.011)	488.882 (198.598)	656.620 (190.309)
f_{11}	**10.8892** **(4.57256)**	145.676 (31.6768)	886.584 (81.9583)	556.695 (54.3356)	300.701 (60.6101)	578.039 (70.6390)
f_{12}	**214.057** **(61.5520)**	243.803 (22.4262)	893.732 (82.3379)	638.685 (89.1132)	331.134 (60.9911)	608.074 (122.137)
f_{13}	321.866 (67.3873)	**244.141** **(15.4904)**	862.863 (107.535)	802.363 (83.9000)	401.170 (64.0598)	606.081 (113.671)
f_{14}	**682.159** **(210.372)**	4958.29 (730.080)	8279.69 (312.258)	4259.14 (656.388)	5509.96 (249.749)	5761.95 (880.590)
f_{15}	**4228.79** **(739.199)**	7469.75 (499.136)	8420.23 (385.411)	4430.85 (543.436)	6187.49 (218.123)	6529.23 (844.054)
f_{16}	**1.08813** **(0.57710)**	2.83395 (0.44142)	3.10000 (0.32815)	2.45991 (0.81697)	3.00184 (0.39360)	2.24419 (0.53354)
f_{17}	**41.0289** **(3.73051)**	304.317 (38.4732)	1623.66 (147.851)	423.017 (62.3409)	452.697 (79.8105)	644.679 (111.494)
f_{18}	**271.836** **(94.8016)**	352.716 (35.3014)	1580.91 (202.456)	424.517 (57.9943)	438.177 (64.0039)	695.007 (112.233)
f_{19}	**3.22370** **(0.88575)**	29.9533 (5.17000)	1.7E+06 (4.7E+05)	14306.7 (6250.22)	3344.00 (7326.12)	223.927 (227.170)
f_{20}	14.5159 (0.67311)	14.5018 (0.95362)	15.0000 (2.8E−07)	14.9832 (8.8E−02)	**13.8519** **(0.38325)**	14.8425 (0.22760)

Table 7
P-values of the rank-sum test for multimodal functions.

	NA	PSO	ABC	GSA	FPA	WOA
f_6	N/A	3.02E−11	3.02E−11	3.02E−11	3.02E−11	3.02E−11
f_7	4.80E−07	N/A	3.02E−11	3.02E−11	2.37E−10	3.69E−11
f_8	N/A	0.099253	0.420390	0.016285	0.982310	0.145319
f_9	N/A	0.935191	3.69E−11	1.78E−10	9.79E−05	2.44E−09
f_{10}	N/A	1.01E−08	3.02E−11	2.39E−08	3.69E−11	3.02E−11
f_{11}	N/A	3.02E−11	3.02E−11	3.02E−11	3.02E−11	3.02E−11
f_{12}	N/A	0.010313	3.02E−11	3.02E−11	4.69E−08	3.02E−11
f_{13}	3.32E−06	N/A	3.02E−11	3.02E−11	8.99E−11	3.02E−11
f_{14}	N/A	3.02E−11	3.02E−11	3.02E−11	3.02E−11	3.02E−11
f_{15}	N/A	3.34E−11	3.02E−11	0.19579	2.87E−10	4.62E−10
f_{16}	N/A	3.82E−10	2.61E−10	3.01E−07	2.61E−10	5.46E−09
f_{17}	N/A	3.02E−11	3.02E−11	3.02E−11	3.02E−11	3.02E−11
f_{18}	N/A	6.36E−05	3.02E−11	1.10E−08	1.56E−08	4.50E−11
f_{19}	N/A	3.02E−11	3.02E−11	3.02E−11	3.02E−11	3.02E−11
f_{20}	9.77E−07	2.07E−04	2.98E−11	1.98E−11	N/A	3.22E−11

Table 8
Statistics of mean (STD) on composition functions.

	NA	PSO	ABC	GSA	FPA	WOA
f_{21}	**318.450** **(80.3945)**	569.442 (103.476)	4264.64 (252.984)	1888.82 (42.1282)	1223.15 (287.070)	1020.46 (471.885)
f_{22}	**621.618** **(332.531)**	5011.03 (774.709)	8684.51 (310.501)	7114.14 (691.066)	6303.22 (368.179)	6980.04 (1184.67)
f_{23}	**5695.66** **(705.824)**	7609.00 (527.775)	8764.60 (346.941)	6671.23 (445.633)	6647.93 (329.209)	7653.72 (858.937)
f_{24}	**295.496** **(8.82772)**	300.538 (11.9088)	306.596 (2.87171)	550.927 (60.2002)	310.061 (5.91868)	323.818 (9.48666)
f_{25}	309.630 (12.9470)	314.979 (6.98130)	**305.340** **(3.60775)**	420.196 (12.2462)	323.896 (6.49773)	327.049 (12.2165)
f_{26}	357.018 (63.3964)	364.146 (54.4969)	384.629 (27.6455)	428.174 (38.1113)	**215.594** **(49.4309)**	371.463 (76.3585)
f_{27}	**1124.34** **(94.0953)**	1184.94 (87.8204)	1360.32 (37.0000)	1456.20 (90.8954)	1320.40 (47.8540)	1383.50 (80.0896)
f_{28}	**342.224** **(231.269)**	1114.74 (138.712)	7003.53 (490.761)	4986.53 (477.406)	2562.57 (435.658)	4793.50 (777.446)

Table 9
P-values of the rank-sum test for composition functions.

	NA	PSO	ABC	GSA	FPA	WOA
f_{21}	N/A	7.34E−11	3.00E−11	3.00E−11	3.00E−11	3.00E−11
f_{22}	N/A	3.02E−11	3.02E−11	3.02E−11	3.02E−11	3.02E−11
f_{23}	N/A	3.02E−11	3.02E−11	3.02E−11	3.02E−11	3.02E−11
f_{24}	N/A	0.063533	2.49E−06	3.02E−11	4.31E−08	8.15E−11
f_{25}	0.379030	2.20E−07	N/A	3.02E−11	4.98E−11	1.46E−10
f_{26}	2.96E−05	8.48E−09	4.62E−10	6.70E−11	N/A	3.16E−10
f_{27}	N/A	0.013271	1.33E−10	5.49E−11	1.69E−09	1.78E−10
f_{28}	N/A	5.57E−10	3.02E−11	3.02E−11	3.02E−11	3.02E−11

function whose solution space has abrupt mutations. However, most search parameters of NA change gradually which are limited by the state of the current best solution. Hence, the NA does not address f_{13} in an efficient way.

Nevertheless, Fig. 4 reveals that NA has notable advantages of both convergence accuracy and speed on the other 12 functions. Only a few cases (e.g. PSO on f_6, f_{19}, GSA on f_{15}) converge faster than NA temporarily. However, they have the problem of premature convergence whose ultimate accuracy is not good as NA. In Table 7, the p-values suggest that both the superiority and the inferiority of NA are statistically significant in most cases, besides f_8, f_9, and f_{15}. The NA obtains the best accuracy on f_8, followed by FPA, ABC, WOA, and PSO whose p-values are greater than 0.05. In other words, the probability of the wrong assertion that NA is significantly better than them is greater than 5%. Thus, the NA has no significant difference to these 4 algorithms on f_8. For the same reason, although NA gets the minimum error on f_9 and f_{15}, PSO and GSA show the similar performance only inferior to NA, respectively. Besides that, the significant superiorities of NA on other multimodal functions are remarkable. It can be considered that NA has a strong capability of global exploration on most problems.

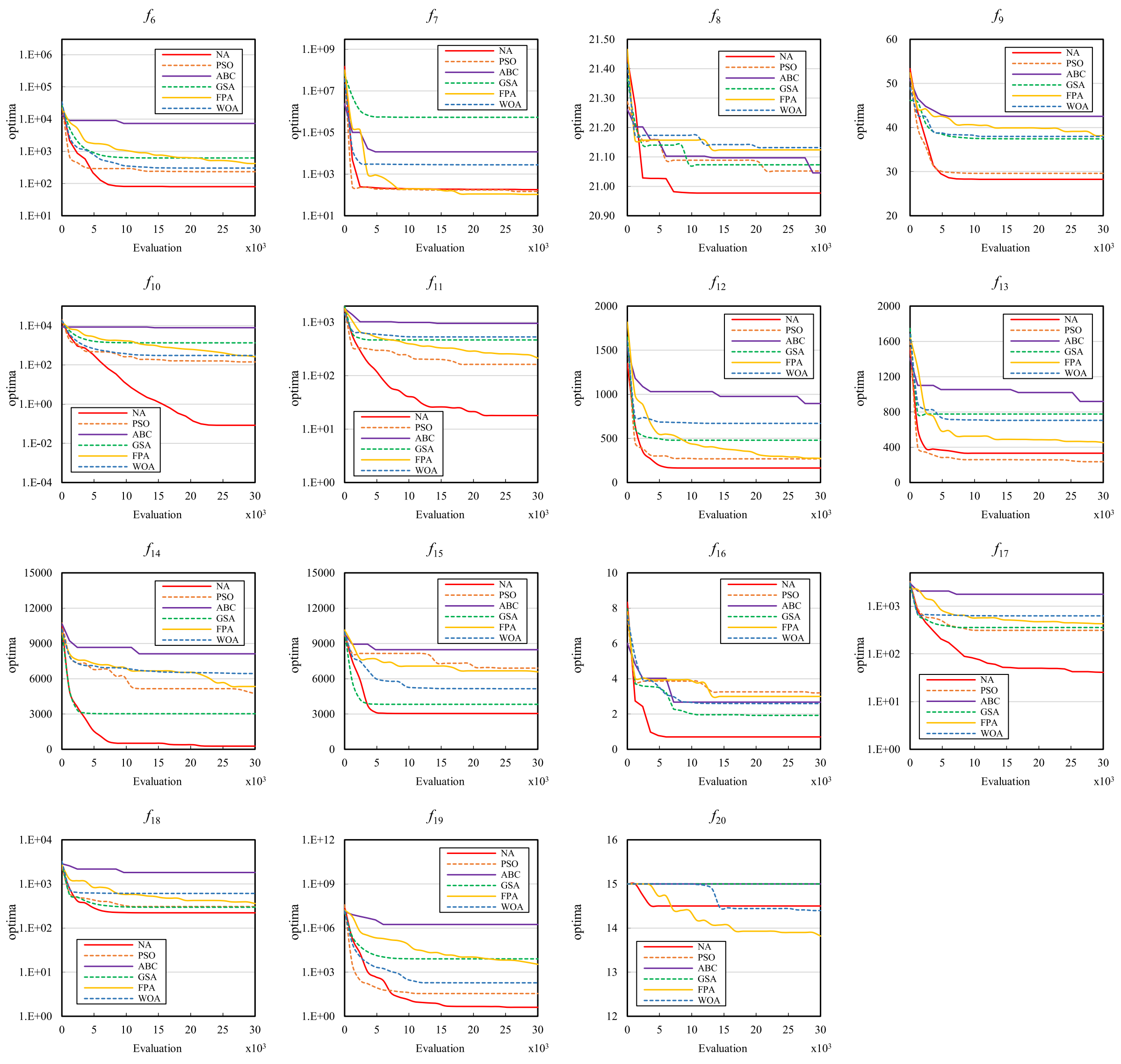

Fig. 4. Convergence curves on multimodal benchmark functions $f_6 - f_{20}$.

The results of the rest composition functions $f_{21} - f_{28}$ are given in Tables 8 and 9. NA shows the best performance on 6 functions of the total 8 composition functions. Table 8 also provides the competitive results of NA are statistically significant, except only one case that p-value of PSO is greater than 0.05 on f_{24}. Although NA shows the second-best accuracy on f_{25}, it is the only one whose p-value (0.37) is greater than 0.05. It statistically indicates the difference between NA and the best ABC is not significant. As shown in Fig. 5, NA often finds better accuracy than its counterparts rapidly, while almost all these algorithms converge prematurely on composition functions. Even in the poor cases f_{25} and f_{26}, NA does not show the notable weakness in the searching process. The gap between the optima of NA and the best algorithm is too small to be noticed. These composition functions are extremely complicated and challenging, which have higher requirements for both local exploitation and global exploration. A series of competitive performance of NA has verified that NA is a promising optimizer which handles the two types of search and their trade-off rationally and efficiently.

The time cost is also an important criterion and demonstrates the actual time consumed in an optimization process. To fairly compare the real-time cost, we set the same 30,000 evaluations for all algorithms. Keep the other conditions unchanged, 50 independent runs are used to obtain the average value of time cost, which reflects the search time of different algorithms in general. Table 10 lists the average runtime of these 6 algorithms on every function.

NA, as shown in Table 10, is the second-fastest one among the six algorithms. Under the same scenarios, the time cost of NA is only inferior to PSO and relatively acceptable. The difference in time cost becomes clearer while handling simple problems, such as the basic unimodal functions. Since the simplicity of the objective functions, the major time consumption of the optimization

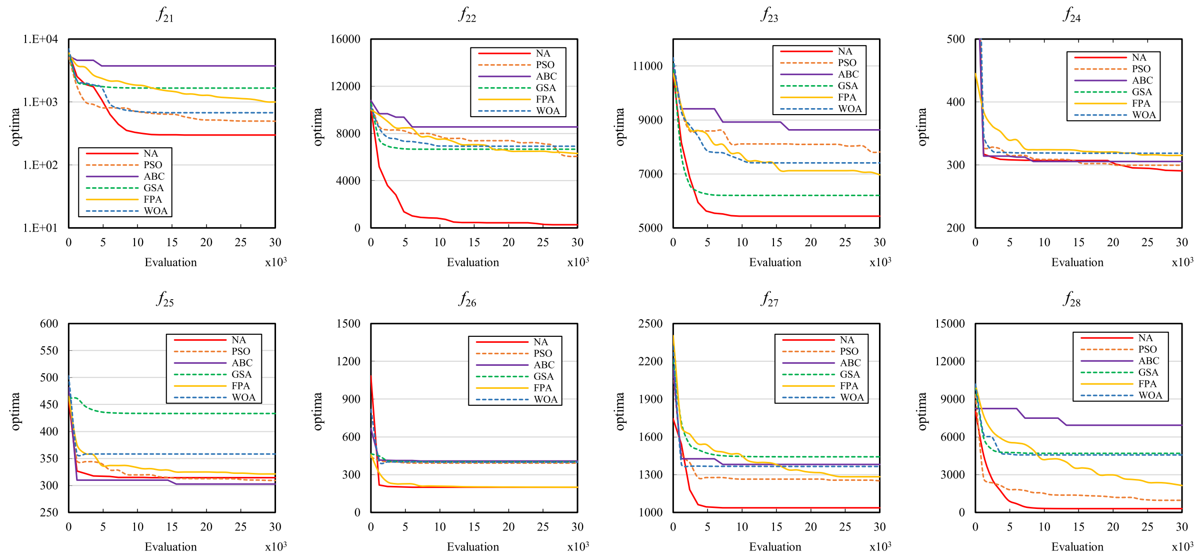

Fig. 5. Convergence curves on composition benchmark functions $f_{21} - f_{28}$.

Table 10
The time cost of 6 algorithms (measured in seconds).

	Func.	NA	PSO	ABC	GSA	FPA	WOA
Unimodal	f_1	0.2201	0.1010	1.3994	2.2421	0.6500	0.3067
	f_2	0.3536	0.2541	1.5135	3.5666	0.7968	0.4343
	f_3	0.3760	0.2453	1.5427	3.5234	0.8260	0.4609
	f_4	0.2906	0.1734	1.4312	3.3916	0.7411	0.3796
	f_5	0.3052	0.1510	1.6328	2.6197	0.7656	0.3937
Multimodal	f_6	0.2656	0.1609	1.4401	3.4000	0.6968	0.3479
	f_7	0.6265	0.5140	1.7354	3.5578	1.0812	0.7208
	f_8	0.8052	0.4114	1.7739	2.5244	1.0161	0.6208
	f_9	4.3562	4.2312	5.4734	7.2859	4.8031	4.4343
	f_{10}	0.4223	0.3119	1.5609	3.3927	0.9213	0.4890
	f_{11}	0.4145	0.3140	1.5026	3.2713	0.8099	0.4880
	f_{12}	0.5770	0.4651	1.6708	3.4468	0.9812	0.6328
	f_{13}	0.7603	0.4868	1.8343	3.7463	1.0661	0.6849
	f_{14}	0.4250	0.2843	1.5229	2.2828	0.8302	0.4781
	f_{15}	0.4645	0.3307	1.5572	3.3567	0.8770	0.5114
	f_{16}	1.6858	1.4729	2.7791	4.6744	2.0609	1.6912
	f_{17}	0.3562	0.2531	1.5450	3.5075	0.7812	0.4343
	f_{18}	0.4781	0.3606	1.6837	3.6256	0.8943	0.5387
	f_{19}	0.3350	0.2181	1.5112	3.5087	0.7512	0.4087
	f_{20}	0.7854	0.3015	1.5458	3.3427	0.8557	0.5026
Composition	f_{21}	0.6475	0.5181	1.8037	3.6318	1.0718	0.7206
	f_{22}	0.7651	0.6458	1.8614	3.6411	1.1953	0.8250
	f_{23}	0.9276	0.8000	2.0312	3.7859	1.3760	0.9843
	f_{24}	5.1962	5.0625	6.4625	8.0906	5.6000	5.3250
	f_{25}	5.1770	4.9755	6.3947	8.3015	5.6536	5.2994
	f_{26}	5.4125	5.1515	6.5072	8.2224	5.7312	5.3786
	f_{27}	5.3015	5.0468	6.4067	8.2250	5.7671	5.2625
	f_{28}	1.2244	1.0869	2.3776	4.1609	1.6500	1.2864

process is contributed from the algorithms' search strategies. Although most optimizations are completed around 1 s, it is observed that f_9 and $f_{24} - f_{27}$ takes remarkably more time consumption for all algorithms. This phenomenon is explained by the difficulty of the problem, i.e., the complexity of objective functions. According to the problem definitions in CEC 2013, f_9 (Rotated Weierstrass function) has a more complex computation which is multimodal, asymmetrical, continuous but differentiable only on a set of points, which costs more time for optimization. Introduction, composition functions $f_{21} - f_{28}$ are integration of different basic functions, while only $f_{24} - f_{27}$ include f_9 which leads to obvious increasing on time cost.

By contrast, with the composition of multiple complicated functions, each evaluation consumes non-negligible amount of time that accounts for the major part of total time. Evaluation time of every algorithm is similar, which reduces the gap of total time among these algorithms on complex problems, likes $f_{21} - f_{28}$. Although NA is a little slower than PSO, it gets the better optima than PSO on 26 of the total 28 benchmark functions. NA, thus, has been demonstrated a light-weight and effective nature-inspired algorithm for global optimization.

4.4. Characteristics and limitations

Apart from that, NA has also shown the following merits:

- The introduced Herdsmen grazing mechanism with a dynamic radius guarantees the promising exploitation capability of NA. The probability of finding a better solution is increased under the circumstance when no better solution has been found in the previous iteration.
- The perturb strategy of Rangers exploring helps NA jump out of local optima. The adaptively changed proportion of Herdsmen and Rangers makes a good trade-off between exploitation and exploration, which maintains the global ability during the whole process of algorithm execution.
- No matter where the Optima is in the search space, NA is fair without partiality. The efficiency of NA only depends on the scale and shape of the objective function.
- NA demonstrates excellent efficiency in sampling strategy, which results in better accuracy under the same time consumption. It means more reasonable utilization and less waste of computing resources.

With the respect of limitations, two issues should be further addressed: (1) The dynamic factors of search radio, which significantly impacts the speed and precision, are set as constants empirically. This setting is not flexible to handle problems with different scales. (2) Although NA takes a self-balancing mechanism, it is adaptive to the maximum iteration and optimization process, rather than the complexity of problem or shape of solution space. A better sensitivity coefficient should be studied for adaptability.

5. Conclusion

This paper proposes an efficient NIC algorithm for global optimization, named Nomad Algorithm (NA). The parameters tuning is studied and the guarantee of global convergence is also proved mathematically. To verify the superiority of NA, it is compared with five well-known algorithms based on CEC 2013 benchmark functions which consist of 5 unimodal, 15 multimodal, and 8 composition functions. According to the experimental results depicted by figures, both the optimization error and convergence speed of NA are superior to that of its counterparts in all 5 unimodal cases, 12 of 15 multimodal cases, and 6 of 8 composition cases. In a nutshell, NA could achieve the highest accuracy in most cases (23/28) which is better than its rivals by multiple orders of magnitude. The nonparametric Wilcoxon statistical test is conducted to confirm the superiority of NA with mathematical significance. The complexity and the real time consumption of algorithm are analyzed to denote NA is a light-weight optimizer which could be rapidly performed. This succinct and efficient mechanism is easy to understand and use for an amateur without expertise of CI.

By these advances, NA is promising to numerical optimization of the real-world industrial applications with high-dimensional and multimodal solution space, e.g., optimizing the designs of optical buffer, pressure vessel, feature selection, and hyperparameter selection. In future work, two research directions can be considered from the perspective of theoretical analysis. Firstly, further study should focus on the detailed convergence process, especially in big-scale complex problems. Besides, the improvement of NA likes multiple tribes competition and better tuning of agents may be feasible and implemented.

CRediT authorship contribution statement

Na Lin: Conceptualization, Methodology, Writing – original draft, Supervision, Resources. **Luwei Fu:** Data curation, Writing – original draft, Software. **Liang Zhao:** Writing – review & editing, Funding acquisition, Supervision. **Ammar Hawbani:** Validation, Writing – review & editing, Funding acquisition. **Zhiyuan Tan:** Writing – review & editing. **Ahmed Al-Dubai:** Writing – review & editing. **Geyong Min:** Writing – review & editing.

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.

References

[1] Del Ser J, Osaba E, Molina D, Yang X-S, Salcedo-Sanz S, Camacho D, Das S, Suganthan PN, Coello CAC, Herrera F. Bio-inspired computation: Where we stand and what's next. Swarm Evol Comput 2019;48:220–50.

[2] Lin N, Fu L, Zhao L, Min G, Al-Dubai A, Gacanin H. A novel multimodal collaborative drone-assisted VANET networking model. IEEE Trans Wireless Commun 2020;19(7):4919–33.

[3] Qu S-J, Ji Y, Zhang K-C. A deterministic global optimization algorithm based on a linearizing method for nonconvex quadratically constrained programs. Math Comput Modelling 2008;48(11–12):1737–43.

[4] Holland JH. Genetic algorithms. Sci Am 1992;267(1):66–73.

[5] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, Vol. 4. IEEE; 1995, p. 1942–8.

[6] Liu W, Wang Z, Liu X, Zeng N, Bell D. A novel particle swarm optimization approach for patient clustering from emergency departments. IEEE Trans Evol Comput 2018;23(4):632–44.
[7] Mirjalili S, Lewis A. The whale optimization algorithm. Adv Eng Softw 2016;95:51–67.
[8] Li J, Tan Y. Loser-out tournament-based fireworks algorithm for multimodal function optimization. IEEE Trans Evol Comput 2017;22(5):679–91.
[9] Mehrabian AR, Lucas C. A novel numerical optimization algorithm inspired from weed colonization. Ecol Inform 2006;1(4):355–66.
[10] Eskandar H, Sadollah A, Bahreininejad A, Hamdi M. Water cycle algorithm–A novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct 2012;110:151–66.
[11] Li S, Chen H, Wang M, Heidari AA, Mirjalili S. Slime mould algorithm: A new method for stochastic optimization. Future Gener Comput Syst 2020;111:300–23.
[12] Yang X-S, Karamanoglu M, He X. Flower pollination algorithm: a novel approach for multiobjective optimization. Eng Optim 2014;46(9):1222–37.
[13] Wang Z-J, Zhan Z-H, Lin Y, Yu W-J, Yuan H-Q, Gu T-L, Kwong S, Zhang J. Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems. IEEE Trans Evol Comput 2017;22(6):894–908.
[14] Mohamed AW, Hadi AA, Mohamed AK. Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm. Int J Mach Learn Cybern 2020;11(7):1501–29.
[15] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 2007;39(3):459–71.
[16] MiarNaeimi F, Azizyan G, Rashki M. Horse herd optimization algorithm: A nature-inspired algorithm for high-dimensional optimization problems. Knowl-Based Syst 2021;213:106711.
[17] Tubishat M, Ja'afar S, Alswaitti M, Mirjalili S, Idris N, Ismail MA, Omar MS. Dynamic salp swarm algorithm for feature selection. Expert Syst Appl 2021;164:113873.
[18] Zhao W, Zhang Z, Wang L. Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. Eng Appl Artif Intell 2020;87:103300.
[19] de Vasconcelos Segundo EH, Mariani VC, dos Santos Coelho L. Design of heat exchangers using Falcon Optimization Algorithm. Appl Therm Eng 2019;156:119–44.
[20] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220(4598):671–80.
[21] Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: a gravitational search algorithm. Inform Sci 2009;179(13):2232–48.
[22] Hashim FA, Hussain K, Houssein EH, Mabrouk MS, Al-Atabany W. Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems. Appl Intell 2021;51(3):1531–51.
[23] Zheng Y-J. Water wave optimization: a new nature-inspired metaheuristic. Comput Oper Res 2015;55:1–11.
[24] Siddique N, Adeli H. Nature-inspired chemical reaction optimisation algorithms. Cogn Comput 2017;9(4):411–22.
[25] Fausto F, Cuevas E, Valdivia A, González A. A global optimization algorithm inspired in the behavior of selfish herds. Biosystems 2017;160:39–55.
[26] Dhiman G, Kumar V. Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. Knowl-Based Syst 2019;165:169–96.
[27] Brezinski K, Guevarra M, Ferens K. Population based equilibrium in hybrid sa/pso for combinatorial optimization: hybrid sa/pso for combinatorial optimization. Int J Softw Sci Comput Intell (IJSSCI) 2020;12(2):74–86.
[28] Solis FJ, Wets RJ-B. Minimization by random search techniques. Math Oper Res 1981;6(1):19–30.
[29] Liang J, Qu B, Suganthan P, Hernández-Díaz AG. Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, Vol. 201212. Technical Report, Singapore: Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University; 2013, p. 281–95, no. 34.

Na Lin received the Ph.D. degree from Northeastern University, Shenyang, China, in 2005. She is the Head of Ubiquitous Networking and Computing Lab, Shenyang Aerospace University, Shenyang. From 2006 to 2010, she worked as the Postdoctoral Researcher with Northeastern University. Her research interests include air-ground integrated networking, SDVNs, and edge computing.

Luwei Fu is pursuing the Ph.D. degree in the School of Computer Science and Engineering at University of Electronic Science and Technology of China (UESTC). His research interests mainly include Mobile Edge Computing, Future Internet, and Evolutionary Computation.

Liang Zhao received the Ph.D. degree from the School of Computing, Edinburgh Napier University, in 2011. He worked as an Associate Senior Researcher with Hitachi (China) Research and Development Corporation from 2012 to 2014. He is currently an Associate Professor with Shenyang Aerospace University, China. He has published more than 100 peer-reviewed articles. His research interests include ITS, VANET, WMN, and SDN.

Ammar Hawbani received the B.S., M.S., and Ph.D. degrees in computer software and theory from the University of Science and Technology of China (USTC), Hefei, China, in 2009, 2012, and 2016, respectively. He is currently a Postdoctoral Researcher with the School of Computer Science and Technology, USTC. His research interests mainly in WSN and WBAN.

Zhiyuan Tan received the Ph.D. degree in computer systems from the University of Technology Sydney, Ultimo, NSW, Australia. He is an Associate Editor of IEEE ACCESS and an Organizer of Special Issues for the Ad Hoc and Sensor Wireless Networks Journal, the International Journal of Distributed Sensor Networks, Computers and Electrical Engineering, and IEEE ACCESS.

Ahmed Y. Al-Dubai received the Ph.D. degree in computing from the University of Glasgow, Glasgow, U.K., in 2004. He is currently a Professor of networking and communication algorithms with the School of Computing, Edinburgh Napier University, Edinburgh. His research interests include communication algorithms, mobile communication, the Internet of Things, and future internet. He has received several international awards.

Geyong Min received the Ph.D. degree in computing science from the University of Glasgow, U.K., in 2003. He is currently a professor of high-performance computing and networking with the College of Engineering, Mathematics and Physical Sciences, University of Exeter, U.K. His research interests include computer networks, wireless communications, parallel and distributed computing, ubiquitous computing, multimedia systems, modeling, and performance engineering.