Applied Soft Computing 114 (2022) 108043

Contents lists available at ScienceDirect

Applied Soft Computing

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

Komodo Mlipir Algorithm☆

Suyanto Suyanto *, Alifya Aisyah Ariyanto, Alifya Fatimah Ariyanto

School of Computing, Telkom University, Bandung, Indonesia

ARTICLE INFO

Article history:
Received 15 March 2021
Received in revised form 18 October 2021
Accepted 4 November 2021
Available online 17 November 2021

Keywords:
Komodo *mlipir* algorithm
Metaheuristic optimization
Self-adaptation of population
Exploitation–exploration balance
Scalable to thousand dimensions

ABSTRACT

This paper proposes Komodo Mlipir Algorithm (KMA) as a new metaheuristic optimizer. It is inspired by two phenomena: the behavior of Komodo dragons living in the East Nusa Tenggara, Indonesia, and the Javanese gait named *mlipir*. Adopted the foraging and reproduction of Komodo dragons, the population of a few Komodo individuals (candidate solutions) in KMA are split into three groups based on their qualities: big males, female, and small males. First, the high-quality big males do a novel movement called high-exploitation low-exploration to produce better solutions. Next, the middle-quality female generates a better solution by either mating the highest-quality big male (exploitation) or doing parthenogenesis (exploration). Finally, the low-quality small males diversify candidate solutions using a novel movement called *mlipir* (a Javanese term defined as a walk on the side of the road to reach a particular destination safely), which is implemented by following the big males in a part of their dimensions. A self-adaptation of the population is also proposed to control the exploitation–exploration balance. An examination using the well-documented twenty-three benchmark functions shows that KMA outperforms the recent metaheuristic algorithms. Besides, it provides high scalability to optimize thousand-dimensional functions. The source code of KMA is publicly available at: https://suyanto.staff.telkomuniversity.ac.id/komodo-mlipir-algorithm and https://www.mathworks.com/matlabcentral/fileexchange/102514-komodo-mlipir-algorithm.

1. Introduction

Mathematical optimization techniques can be categorized into deterministic and stochastic. The former relies on the gradient information in the given problem to get the solution; hence it commonly works well for unimodal problems but not for multimodal ones. In contrast, the latter can handle both problem types. Besides, it is simple, flexible, gradient-free, and also problem-independent. Therefore, it is more popular to tackle many real-world problems with non-linear and non-convex search spaces.

The stochastic methods, also known as the population-based metaheuristic optimization algorithms, are commonly inspired by the nature, such as evolutionary concepts, genetics, animal behaviors, phenomena in planets, cultures, societies, physical phenomena, or musical instruments. Many researchers have been proposing hundreds algorithms and their variants, which can be grouped into three categories: (1) most well studied algorithms, such as Genetic Algorithm (GA) [1,2], Differential Evolution (DE) [3], Genetic Programming (GP) [4,5], Particle Swarm Optimization (PSO) [6], and Ant System (AS) [7]; (2) almost recently developed algorithms, such as Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [8], Success-History Based Parameter Adaptation Differential Evolution (SHADE) [9], SHADE with linear population size reduction (LSHADE) [10], LSHADE with Ensemble Sinusoidal Differential Covariance Matrix Adaptation with Euclidean Neighborhood (LSHADE-cnEpSin) [11], Firefly Algorithm (FA) [12,13], Cuckoo Search (CS) [14,15], Bat Algorithm (BA) [16,17], Bacterial Foraging Optimization (BFO) [18,19], Gravitational Search Algorithm (GSA) [20], Salp Swarm Algorithm (SSA) [21], Teaching–Learning-Based Optimization (TLBO) [22], Gray Wolf Optimizer (GWO) [23,24], Dragonfly Algorithm (DA) [25,26], Ant Lion Optimizer (ALO) [27,28], Moth–Flame Optimization (MFO) [29,30], and Artificial Algae Algorithm (AAA) [31,32]; and (3) recently created algorithms, such as Atomic Search Optimization (ASO) [33], Rao Algorithms (RAs) [34], Evolutionary Rao Algorithm (ERA) [35], Equilibrium Optimizer (EO) [36], Marine Predators Algorithm (MPA) [37,38], and Slime Mould Algorithm (SMA) [39,40].

All the metaheuristic algorithms look the same. They depend on two similar concepts: the movement of an individual (candidate solution) and the interaction among individuals in the population. In general, the movement uses the evolutionary operators (crossover and mutation) or the random walk techniques with a particular distribution, such as uniform, normal (Gaussian), Lévy,

☆ This research is funded by the Ministry of Education, Culture, Research and Technology, Indonesia, based on Matching Fund Kedaireka Grant with contract number: 3541/E3/SK.09/KL/2021.

* Corresponding author.
E-mail addresses: suyanto@telkomuniversity.ac.id (S. Suyanto), alifyaaisyah@student.telkomuniversity.ac.id (A.A. Ariyanto), alifyafatimah@student.telkomuniversity.ac.id (A.F. Ariyanto).

https://doi.org/10.1016/j.asoc.2021.108043

or Brownian. Meanwhile, the interaction commonly adopts the theory of evolution, the natural phenomena, or the animal behaviors to create a searching strategy that produces an exploitation–exploration balance. The exploration widely searches the space to avoid the local optima. In contrast, the exploitation locally searches around promising solutions.

Many recent algorithms are developed based on the similar ideas used in the former by introducing some improvements. In other words, they are the improved versions of the previous methods. For instance, DE is an improved version of GA. It uses real-number individuals instead of the binary ones and introduces two evolutionary operators: binomial crossover and differential mutation with a scaling factor to controls the magnitude of the mutation. Compared to GA, DE generally gives higher performances in solving continuous optimization problems. However, like other metaheuristic algorithms, its performance depends on control parameter settings, which are problem-dependent. Therefore, many variants are proposed by providing a self-adaptation to tune the control parameters adaptively during the evolution. SHADE, LSHADE, LSHADE-cnEpSin, jDE, and IMODE are some of the state-of-the-art DE variants that give high performances for classic benchmark functions [37] and CEC test suites [11,41,42]. Unfortunately, all those algorithms are both unscalable and unstable. They need higher populations and more function evaluations (the number of individuals should be generated and evaluated during the evolution) to solve the higher-dimensional problems. For instance, the population size of LSHADE-cnEpSin should be set to 18 times the dimension of the given problem [11]. Otherwise, its performance may decrease as the dimension increase. Another problem is that, for many cases, they are unstable for some independent runs with different random seeds of the initial populations. In other words, they do not guarantee to obtain the global optimum solutions.

Hence, some recent algorithms are designed to be both scalable and stable, such as EO, MPA, and SMA. EO is inspired by the mass balance models to approximate dynamic and equilibrium states [36]. A particle in EO represents a candidate solution (search agent). Based on qualities or fitness values, a population of some particles in EO is split into two groups: (1) the five equilibrium candidates containing four fittest particles and their average, which exploit the promising area around the global optimum; and (2) the other particles that do exploration. An evaluation using 58 benchmark functions and three real-world problems shows that EO significantly outperforms six competitors: GA, GSA, PSO, SSA, GWO, and CMA-ES but is slightly defeated by SHADE and LSHADE-SPACMA. However, impressively, EO provides scalability from 10 to 200 dimensions, although for few benchmark functions.

MPA is motivated by the broad foraging tactics in ocean predators along with the optimum policy of contact rate in the predator–prey interaction [37]. The strategy is developed using Lévy and Brownian movements. Evaluation using 59 benchmark functions shows that MPA is competitive to the LSHADE-cnEpSin. It significantly outperforms GA, GSA, PSO, SSA, CS, and CMA-ES. Statistically, it can be categorized as a high-performance optimization algorithm that reaches a similar performance as SHADE and LSHADE-cnEpSin. In addition, MPA also provides scalability from 100 to 500 dimensions for few benchmark functions.

Meanwhile, SMA imitates the oscillation mode of slime mold in nature [39]. This method utilizes adaptive weights to mimic the bio-oscillator feedback (negative and positive) throughout the foraging to get an exploitation–exploration balance. It has three steps of evolution: random distribution (finding food), exploration and adaptive transformation (approaching food), and exploitation (wrapping food). An evaluation using 33 benchmark functions shows that, for some of those benchmark functions, SMA is stable and guarantees global optima. As with those recent DE variants, EO, and MPA, the SMA is unscalable and unstable for most benchmark functions.

Until now, none of the metaheuristic optimizers guarantees to get the global optima for all problems and dimensionality sizes. It is logically proven by the theorem of no free lunch [43]. Therefore, many researchers are continuously encouraged to create a new algorithm to solve as many problems as possible, as scalable as possible, as stable as possible in giving the highest guarantee rate. Motivating by those facts, a novel scalable and stable optimizer named Komodo Mlipir Algorithm (KMA) is proposed here. Mimicking the foraging and reproduction of Komodo dragons, the population of a few individuals (candidate solutions) in KMA are split into three groups: high-quality big males, one middle-quality female, and low-quality small males. KMA introduces two new movements, namely high-exploitation low-exploration (HILE) to intensify high-quality solutions and *mlipir* movement that creates a low-exploitation high-exploration (LIHE) to avoid local optima and tackle broad flat (plateaus) areas.

This paper is written in the following structure. A detailed description of KMA is given in Section 2. Next, Section 3 discusses some experimental results for the 23 benchmark functions compared to six algorithms: GA, SHADE, LSHADE-cnEpSin, EO, MPA, and SMA. Furthermore, the convergence, scalability, and stability analysis for higher-dimensional functions are then provided for the proposed KMA and those five algorithms. The comprehensive discussion is then given in Section 4. The conclusion is finally provided in Section 5.

2. Proposed KMA

This section discusses two inspirations of the proposed KMA: the Komodo behavior and the typical Javanese gait "*mlipir*". The concept and mathematical model are then given in detail with a clear illustration and formulation. Finally, the pseudocode of KMA is provided.

2.1. Komodo in the wild nature

The Komodo dragon (Varanus komodoensis) is a monitor lizard of the family Varanidae [44] living on Komodo Island and a few neighboring islands of the Lesser Sunda Islands nearby Flores, the East Nusa Tenggara, Indonesia. It grows to 3 meters in length and about 135 kg (300 pounds) in weight [45]. It has unique behaviors in foraging and reproduction.

The big adult Komodo dragons eat large prey, such as deer, goats, boar, and carrion. Besides, they cannibalize the smaller young ones and sometimes even other adults [46]. When a big Komodo eats prey, some other Komodo dragons are attracted to join [47]. However, the big Komodo can be willing or refuses to share the prey.

The female Komodo dragons may produce offspring by mating the big adult male or sometimes through parthenogenesis (asexual reproduction) [44,48]. Some big males often do bloody combat for prey or female. The winner can keep the prey or mate the female (if she wants) for the sexual reproduction [47]. Suppose the female does not want to mate the winner. In that case, the female takes parthenogenesis [48].

Meanwhile, the small males do foraging by looking for leftovers left by the big males. However, they should keep their distance or will be cannibalized by the big males. Thus, they move aside from the big males. Once have an opportunity, they approach the leftovers.

2.2. Mlipir

"*Mlipir*" is a word in the Javanese language that can be defined as "walk on the side of the road to avoid danger" or "moving along the roadside carefully, without being noticed by anyone, and successfully reaching a certain destination safely". It is a typical Javanese gait (style of walking) in order to reach the destination safely.

As described in Section 2.1, the small male Komodo dragons use the movement strategy like *mlipir* during the foraging. They find leftovers left by the big males and avoid being cannibalized by them. Thus, they sneaked around the big males. As soon as the opportunity arises, they grab the leftovers safely.

2.3. Concept of KMA

The concept of KMA can be simply illustrated in Fig. 1. Given a simple optimization problem of two-dimensional function $f(x_1, x_2) = x_1^2 + x_2^2$, where x_1 and x_2 are the horizontal and vertical axes, respectively, and the global optimum solution (target) is $f = 0$ at $x_1 = 0$ and $x_2 = 0$. First, let an initial population containing six Komodo individuals (candidate solutions): $\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_6$ is randomly generated, where each Komodo only represents a vector of position (with no velocity) of a candidate solution in the problem space. Next, the population is split into three groups: high-quality big males, middle-quality female, and low-quality small males. Here, the highest-quality big male mimics the strongest Komodo that catches the only prey, which represents the best-so-far solution.

The evolution is then performed in a particular order. First, the big males interact with each other to do a high-exploitation low-exploration (HILE). Next, using the normally distributed probability of 0.5, the female either mates the best big male or parthenogenesis (asexual reproduction), doing medium-exploitation or medium-exploration (MIME). Finally, the small males do the proposed novel *mlipir* movement to surround the search space to do a low-exploitation high-exploration (LIHE).

As the big males, two Komodo dragons $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$ fight with each other. It is adopted in this model as attraction and distraction using a particular rule. A lower-quality big male should move forward to the higher-quality big males. In contrast, a higher-quality big male can either move forward or backward to the lower-quality ones with the probability of 0.5. Hence, some big males do a high-exploitation (intensification) in a narrow area around the global optimum while the others do a low-exploration (diversification) in a wider area. As illustrated in Fig. 1, $\boldsymbol{k}_2$ is attracted to $\boldsymbol{k}_1$ (shown by $\boldsymbol{w}_{21}$) that makes it moves toward the global optimum (exploitation). This movement mimics the Komodo's behavior to fight for seizing the prey kept by another big male. In contrast, $\boldsymbol{k}_1$ is distracted from $\boldsymbol{k}_2$ (shown by $\boldsymbol{w}_{12}$) randomly with a probability of 0.5 and do exploration. This movement imitates the Komodo's behavior to keep the prey for itself. In addition, like most metaheuristic algorithms, KMA is designed to keep the highest-quality big males.

Next, the female either mates the winner Komodo individual (with the highest fitness) to generate two offsprings (shown by $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$) or do parthenogenesis using a random movement (shown by $\boldsymbol{h}$). Hence, as middle-quality, the female searches for the solution in a medium-exploitative medium-explorative manner.

Finally, as the low-quality small male, three Komodo dragons $\boldsymbol{k}_4, \boldsymbol{k}_5$, and $\boldsymbol{k}_6$ do a *mlipir* to move aside the big males by randomly select a part of their dimensions. The dimensional selection is implemented using the normal distribution with a fixed probability of 0.5. This movement is a new method in the field of metaheuristic optimization. Here, this movement is named "*mlipir*" and the probability of dimensional portion is called "*mlipir rate*". In Fig. 1, the "*mlipir rate*" is 0.5, which means the small males follow the big males only in half of the dimensions. For example, $\boldsymbol{k}_4$ moves aside $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$ using only the first dimension x_1 (horizontal), which are shown by $\boldsymbol{w}_{41}$ and $\boldsymbol{w}_{42}$, respectively. Hence, summing the two vectors $\boldsymbol{w}$ creates a movement that makes $\boldsymbol{k}_4$ walks aside and get a new position close to $\boldsymbol{k}_2$. In contrast, $\boldsymbol{k}_5$ moves aside $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$ using only the second dimension x_2 (vertical), which are shown by $\boldsymbol{w}_{51}$ and $\boldsymbol{w}_{52}$, respectively. Summing the three vectors $\boldsymbol{w}$ moves $\boldsymbol{k}_5$ to the above, surrounding the search space. Finally, $\boldsymbol{k}_6$ vertically follows $\boldsymbol{k}_1$ and horizontally follows $\boldsymbol{k}_2$ (shown by $\boldsymbol{w}_{61}$ and $\boldsymbol{w}_{62}$), which diagonally moves $\boldsymbol{k}_6$ forward to the global optimum. Interestingly, this *mlipir* movement is not only doing a high-exploration but may also do a low-exploitation. Hence, the novel *mlipir* movement plays a critical role in the proposed KMA. Since the population in the beginning generations has high diversity, this *mlipir* movement makes the small males do a high-exploration to cover the entire search space. In the last iterations, they do a low-exploration.

All the three movements of big males, females, and small males, can be extended to the high-dimensional spaces, such as m-dimensions. The big males attract and distract each other in the whole m-dimensions while the small males do a *mlipir* movement to the big males only in the dm dimensions, where d is the *mlipir* rate represented as a real number in the interval $(0, 1)$ to select a part of the dimensions: $1, 2, \ldots, (m-1)$. Besides, all new positions of the small males are kept without survivor selection since the big males have performed.

2.4. Pseudocode of KMA

The concept of KMA can be explained using pseudocode in Algorithm 1. First, three parameters of n, p, and d are set as the number of Komodo, the portion of the big males, and the *mlipir* rate, respectively. Next, n Komodo are created randomly. After that, the evolution is performed until a stopping criterion is reached. In each generation, all Komodo are evaluated based on the objective of the given problem. The qualities are then sorted to define their ranks. Based on the ranks, the population is split into three groups: q high-quality big males, one middle-quality female, and s low-quality small males based on Eqs. (1) and (2), respectively. Each big male is moved using Eq. (4), and the q best position is kept to survive in the next generation. Then, the female does either sexual or asexual reproduction based on Eq. (5) and (7), respectively. Next, each small male is moved using Eq. (9), and all their new positions are kept to survive in the next generation (with no survivor selection). Finally, the population is adaptively updated using Eq. (10). Once the stopping criterion is reached, the highest-quality Komodo $\boldsymbol{k}_{best}$ is returned as the global optimum solution.

The pseudocode of KMA is implemented in two phases to tackle unimodal and multimodal functions effectively and efficiently. In the first phase, KMA uses a low population size with a half portion of big males and a maximum *mlipir* rate. This phase is run for 1000 generations to examine the complexity of the given problem. If the given problem has a low complexity (unimodal), KMA should obtain the global optimum swiftly. Once the global optimum is reached, KMA returns the solution and stops the evolution. Otherwise, the second phase is applied until the maximum function evaluations are reached or KMA reaches the global optimum. In the second phase, KMA uses a higher population with a self-adaptation procedure to automatically online-tune the population size during the evolution process with a half portion of big males and a half *mlipir* rate. In this phase, KMA is expected can solve complex (multimodal) benchmarks.

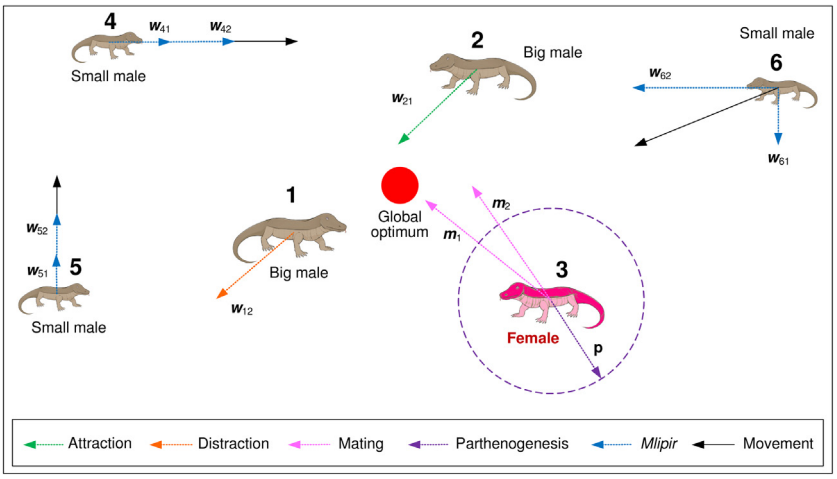


Fig. 1. Concept of Komodo Mlipir Algorithm.

Algorithm 1: Komodo Mlipir Algorithm

Result: $\boldsymbol{k}_{best}$ as the global optimum solution
Set n, p, and d as the number of Komodo individuals, the portion of big males, and the *mlipir* rate, respectively;
Initialization of n individuals with m dimensions;
while *StoppingCriterion* = *false* **do**
  for each Komodo, calculate its quality, and then rank them;
  Based on their ranks and the portion p, split the population into three groups: q highest-quality big males, 1 middle-quality female, and s low-quality small males using Eqs. (1) and (2);
  for each big male, move it using Eq. (4), and keep the q highest-quality big males (best positions) to survive in the next generation;
  Update the female by either mating the winner big male using Eq. (5) or doing parthenogenesis using Eq. (7);
  for each small male, move it using Eq. (9), and keep all their new positions to survive in the next generation;
  Update the population size n using Eq. (10);
  Select the highest-quality Komodo from the three groups as the best-so-far solution $\boldsymbol{k}_{best}$
end

2.5. Komodo individual representation

In KMA, a Komodo individual is expressed as a real-valued vector $\boldsymbol{k}$ with m dimensions, which determines a position in the problem space. In Fig. 1, the population contains of six Komodo, $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, …, $\boldsymbol{k}_6$ with dimension $m = 2$. Like a firefly in FA, each Komodo in KMA is designed to have only a position with no velocity. It more dynamically moves in the search space since the inertia is ignored in the movement.

2.6. Three groups of individuals

The population of n Komodo individuals are split into three groups: high-quality big males, middle-quality females, and low-quality small males with a portion p, which is in the interval (0, 1) and can be generally tuned as 0.5 for any given problem. Using a portion p, the population of n Komodo is split into q big males, one female, and s small males using the following formulas

$$q = \lfloor (p-1)n \rfloor, \tag{1}$$

$$s = n - q, \tag{2}$$

However, a too-small or a too-high portion p can produce zero for q or s. Hence, a simple procedure is applied to enforce both big males and small males with at least two Komodo interactions.

2.7. Movements of big males

The big males interact by attraction or distraction based on a simple rule introduced in this research. A low-quality big male should be attracted by the higher-quality one. In contrast, a high-quality big male can be attracted or distracted by the lower-quality ones randomly with the probability of 0.5 to get a new position. This scheme ensures the chance of exploitation is always higher than exploration. Therefore, this novel movement scheme is named high-exploitation low-exploration (HILE).

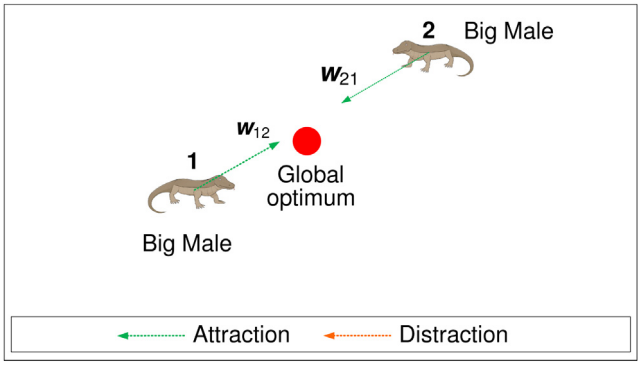
The movement of a big male $\boldsymbol{k}_i$ to produce a new position $\boldsymbol{k}'_i$ can be defined using two following formulas

$$\boldsymbol{w}_{ij} = \begin{cases} r_1(\boldsymbol{k}_j - \boldsymbol{k}_i), & \text{if } f(\boldsymbol{k}_j) < f(\boldsymbol{k}_i) \text{ or } r_2 < 0.5 \\ r_1(\boldsymbol{k}_i - \boldsymbol{k}_j), & \text{otherwise,} \end{cases} \tag{3}$$

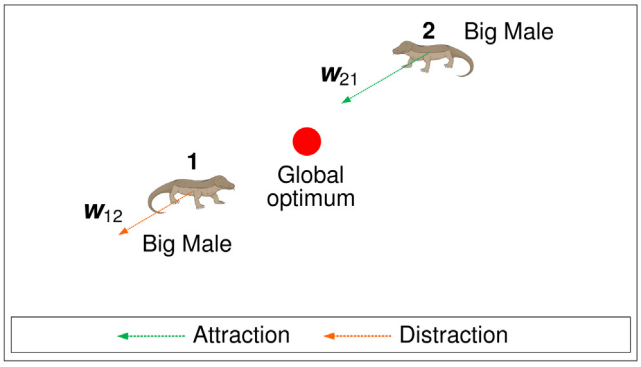
$$\boldsymbol{k}'_i = \boldsymbol{k}_i + \sum_{j=1}^{q} \boldsymbol{w}_{ij}, \text{ where } j \neq i, \tag{4}$$

where $f(\boldsymbol{k}_i)$ and $f(\boldsymbol{k}_j)$ are fitness (or qualities) of the ith and the jth big males, respectively, $\boldsymbol{k}_i$ and $\boldsymbol{k}_j$ are the ith and the jth big males (where $j \neq i$), respectively, r_1 and r_2 are random numbers in the interval [0, 1] in the normal distribution, and q is the number of big males.

If there are two big males, then the low-quality one should do exploitation, and the high-quality one may do either exploitation or exploration randomly with the probability of 0.5. Hence, there are two possible movements of those big males as illustrated in Fig. 2. In possible movement 1, both big males do an exploitative



(a) Possible movement 1 of two big males



(b) Possible movement 2 of two big males

Fig. 2. Two possible movements of two big males.

search in areas nearby the global solution. Meanwhile, in the possible movement 2, the low-quality Big Male 2 does exploitation by attracting the higher-quality Big Male 1. In contrast, Big Male 1 does exploration by distracting Big Male 2. Hence, in this case, the probability of exploitation is higher than the exploration.

If the population contains three big males, there are eight possible movements of the three big males, as illustrated in Fig. 3. In possible movement 1, all the three big males do exploitation close to the global optimum. Meanwhile, in the possible movement 2, 3, 4, and 5, two big males do exploitation, and the rest does an exploration. Finally, in the possible movements 6, 7, and 8, only one big male does exploitation, and the rests do exploration. Hence, in all possible movements, at least one big male is doing exploitation. There is no possibility that all the three big males do exploration. Therefore, it can be guaranteed that the probability of exploitation should be higher than the exploration.

However, there must be an optimum number of big males to do the interaction. Hypothetically, two or three big males will give an optimum interaction to balance exploitation and exploration. To solve the unimodal functions, two big males are enough. Nevertheless, to solve the multimodal functions, three big males will be better. The hypothesis is empirically proven based on several preliminary experiments. This interaction concept is different from the EO [36] that uses five particles as the equilibrium candidates to give optimum performance.

2.8. Reproduction of female

As a middle-quality individual, a female is designed to do either exploitation or exploration in the same chance. It is implemented using a fixed probability of 0.5. If the exploitation is selected, the female intensifies the potential area around the global optimum solution by mating the winner big male (the highest-quality) to generate two offsprings (new positions). Otherwise, it diversifies the solution in a wide area of search space by doing parthenogenesis. Eventually, the female is updated by the best-generated offspring. Furthermore, the mating procedure (sexual reproduction) of two Komodo is formulated as

$$\begin{aligned} k'_{il} &= r_l \cdot k_{il} + (1 - r_l) \cdot k_{jl} \\ k'_{jl} &= r_l \cdot k_{jl} + (1 - r_l) \cdot k_{il} \end{aligned} \tag{5}$$

where k_{il} and k_{jl} are the lth dimension of the ith the jth Komodo as the winner big male and the female, respectively, k'_{il} and k'_{jl} are the kth dimension of two offsprings produced by the mating process, and r_k is a random number in interval [0, 1] in the normal distribution for the lth dimension. Since the mating is performed randomly in each dimension, then many possible offsprings can be generated. For example, two possible offsprings are generated as illustrated in Fig. 4(a). Next, the female is updated by the best offspring as shown in Fig. 4(b).

Meanwhile, the parthenogenesis procedure (asexual reproduction) is implemented by appending a small value to each female dimension. The small number is randomly generated using the symmetric normal distribution, which is defined as

$$(k_{i1}, k_{i2}, \ldots, k_{im}) \rightarrow (k'_{i1}, k'_{i2}, \ldots, k'_{im}), \tag{6}$$

$$k'_{ij} = k_{ij} + (2r - 1)\alpha |ub_j - lb_j|, \tag{7}$$

where $k_{i1}, k_{i2}, \ldots, k_{im} \in [lb_j, ub_j]$ is the vector element of m dimensions of a Komodo individual $\boldsymbol{k}$, lb_j and ub_j are the lower and upper bounds of the jth dimension, r is a random value in the normal distribution, and α is the radius of parthenogenesis set to be a fixed value of 0.1, which means the new solution can be generated in the radius of 10% of the search space. Fig. 5(a) illustrates two of the many possible generated offsprings: $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ in the radius of 0.1. Next, let $\boldsymbol{p}_1$ is selected, and the female is then updated by the $\boldsymbol{p}_1$, as shown in Fig. 5(b).

2.9. Movements of small males

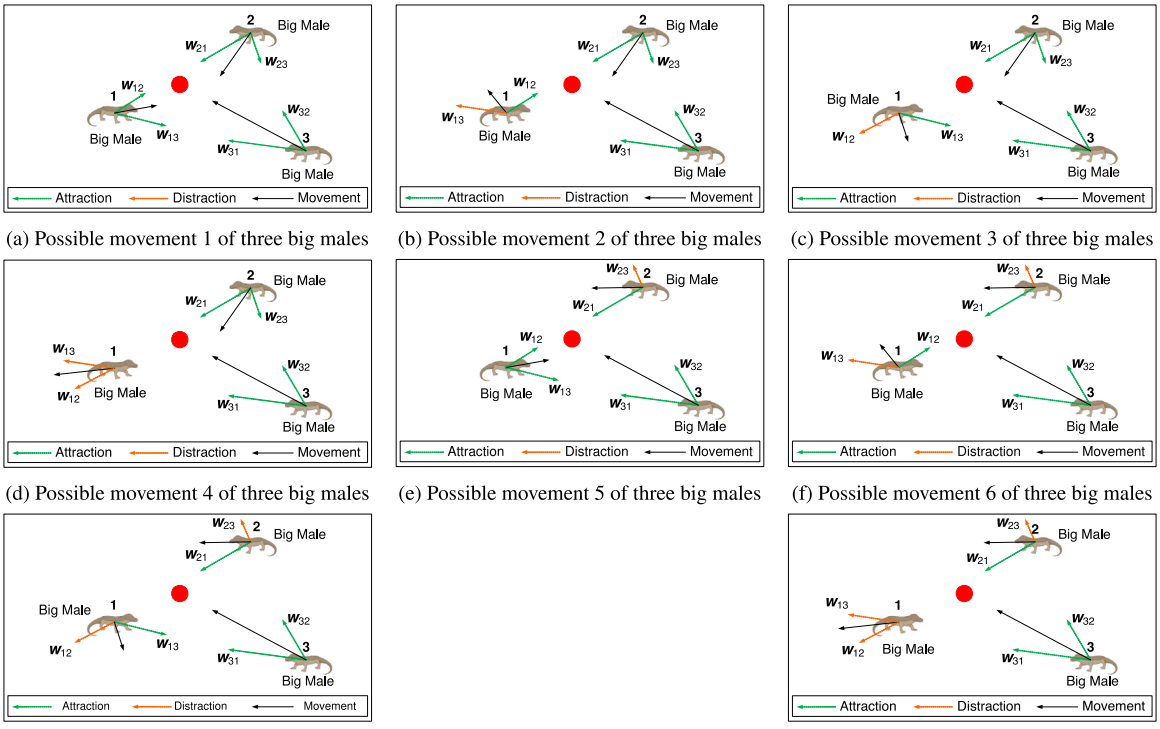
In wild nature, a small male Komodo has two objectives: (1) look for leftovers left by the big males; and (2) avoid being cannibalized by them by moving aside (*mlipir*). This behavior is adopted in KMA by creating a novel movement called *mlipir*. As low-quality individuals, the small males are designed to search for the solution in a broad area but sometimes do a little exploitation. Each small male moves aside (*mlipir*) from the big males, which is implemented by randomly selecting a part of their dimensions with a particular probability (*mlipir rate*). With a *mlipir rate* d in the interval (0, 1), the *mlipir* movement of the ith Komodo to follow the jth one is formulated as

$$w_{ij} = \begin{cases} \sum_{l=1}^{m} r_1(k_{jl} - k_{il}), & \text{if } r_2 < d \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

$$\boldsymbol{k}'_i = \boldsymbol{k}_i + \sum_{j=1}^{q} w_{ij}, \quad \text{where } j \neq i, \tag{9}$$

where r_1 and r_2 are randomly generated numbers in [0, 1] using the normal distribution: r_1 defines the velocity of the movement while r_2 selects the dimension to follow, m is the dimension, k_{il} and k_{jl} is the lth dimension of the ith small male and the jth big male, respectively, l is the randomly selected dimension, which based on the normal distribution with a probability equals to the *mlipir* rate d to select $1, 2, \ldots, (m-1)$ dimensions of the big male, and q is the number of big males.

If there are two small males and two big males, then there are nine possible combinations of movements of the small males, as



(a) Possible movement 1 of three big males (b) Possible movement 2 of three big males (c) Possible movement 3 of three big males

(d) Possible movement 4 of three big males (e) Possible movement 5 of three big males (f) Possible movement 6 of three big males

(g) Possible movement 7 of three big males (h) Possible movement 8 of three big males

Fig. 3. Eight possible movements of three big males.

illustrated in Fig. 6. In the four possible movements 1, 2, 4, and 5, both small males do an explorative search by moving horizontally and vertically to surround the big males: on the above, below, left, and right. Meanwhile, in the other possible movements 3, 6, 7, 8, and 9, at least one small male does exploitation by moving diagonally closer to the big male. If the movements of both small males are calculated separately, then 12 out of 18 (66.67%) possible movements are exploration, and the rests are exploitation. Hence, in this *mlipir* movement, the probability of exploration is guaranteed to be higher than exploitation.

2.10. Population adaptation scheme

As described above, KMA has three parameters: population size n, portion p, and mlipir rate d. Hypothetically, n is more sensitive than p and d since it strongly controls the exploration and exploitation strategies, while p and d can be defined as the fixed values of 0.5 since the characteristics of the given problems are unknown. Therefore, an adaptation scheme is proposed to tune n adaptively during the evolution. If two successive best-so-far fitness improvements, the population size n is decreased by deleting five individuals. Otherwise, if they show stagnations, n increases by generating five new individuals; here, the new individual is created from the best-so-far big male that is moved randomly. The new population size n' is calculated as

$$n' = \begin{cases} n - a, & \text{if } \delta f_1 > 0 \text{ and } \delta f_2 > 0 \\ n + a, & \text{if } \delta f_1 = 0 \text{ and } \delta f_2 = 0 \end{cases} \tag{10}$$

where a is the number of individuals to delete or generate, $\delta f_1 = \frac{|f_1 - f_2|}{f_1}$ and $\delta f_2 = \frac{|f_2 - f_3|}{f_2}$ are the fitness differences of two successive ith and $(i-1)$th generations, respectively. Here, the value of a is set to five, which represents a small population size used in the first phase. Besides, the initial, minimum, and maximum values of n should be consequently defined. Some preliminary observations inform that the optimum values are 200, 20, and 200, respectively.

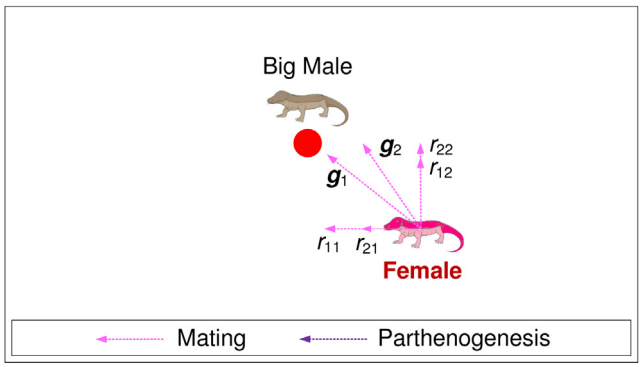
2.11. Difference of KMA and other algorithms

The proposed KMA is different from the existing metaheuristic algorithms since it combines three searching strategies: HILE, MIME, and LIHE to find a global optimum swiftly and avoid the local ones. Both HILE and LIHE movements are entirely different from those used in the existing algorithms, while MIME is the same as crossover and mutation in the GA.

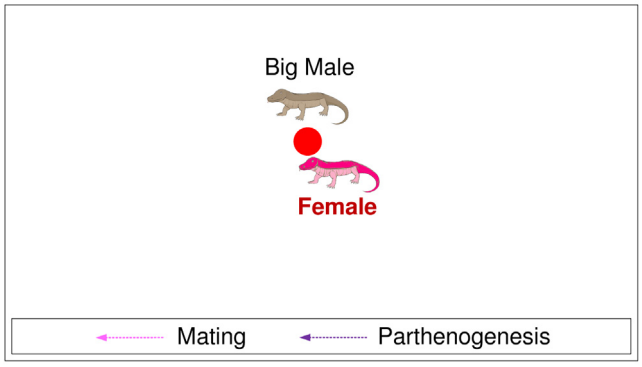
HILE is designed by attraction and distraction between the high-quality individuals with a probability of 0.5, which is different from the attraction and distraction in DFA [25] that are always (with a probability of 1.0) performed based on two rules: a dragonfly attracts to the better ones, and it distracts to the worst. Meanwhile, LIHE performs low-intensification high-diversification using *mlipir* movements of low-quality individuals to avoid local optima. Unlike differential mutations in DE calculated from indirect vectors of two to four best or randomly selected individuals [49], the *mlipir* movement is composed of direct attraction vectors of partial dimensions of one to three randomly selected higher-quality individuals. As illustrated in Fig. 6, this *mlipir* movement logically provides a higher exploitation strategy than the differential mutations in DE.

2.12. Computational costs of KMA

The computational costs of KMA can be easily estimated based on complexity analysis. The time complexity of KMA for one generation (iteration) is $O(nm+nc+\log n)$, where n, m, c, and $\log n$ are the number of individuals in the population, the dimension, the objective function calculation, and the sorting process of fitness values, respectively. Compared to GA, EO, MPA, and SMA, the proposed KMA is a little more complicated. It has an additional complexity of $\log p$ come from the sorting process. Besides, it also has another complexity from the population adaptation scheme, although it is quite low.

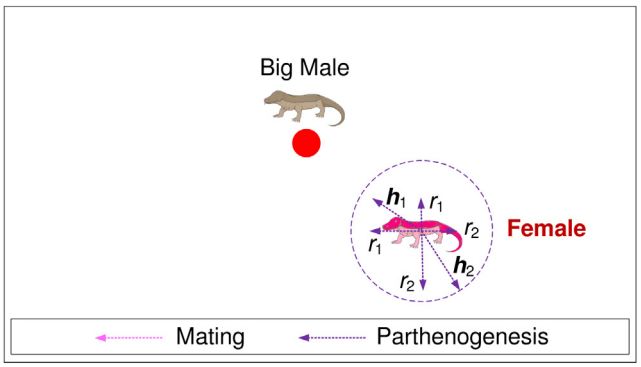


(a) Possible matings of female and the winning big male

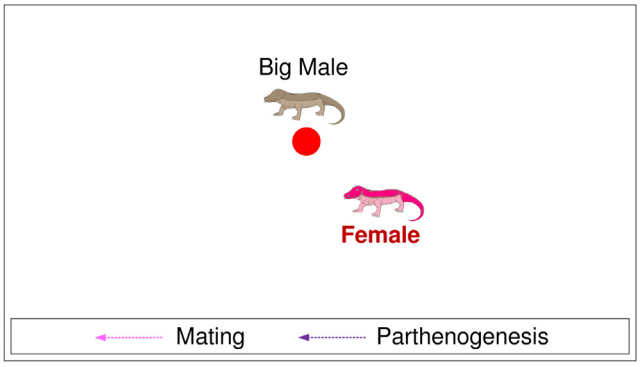


(b) New female offspring after mating

Fig. 4. With a normally distributed probability of 0.5, the female mates the winner big male, then the female is updated by the best offspring (if it is better than the female).



(a) Possible parthenogenesis



(b) New female offspring after doing parthenogenesis

Fig. 5. With a normally distributed probability of 0.5, the female does a parthenogenesis to generate an offspring, then the female is updated if the offspring is better than itself.

3. Experimental results

The commonly used 23 classic benchmark functions: 7 high-dimension unimodal (HDU), 6 high-dimension multimodal (HDM), and 10 fixed-dimension multimodal (FDM) as described in [34] are used here to examine the three capabilities of KMA: exploitation, exploration, and scalability. The detailed descriptions of the 23 classic benchmark functions and their plots in two-dimensions are depicted in Table 6 and Fig. 8 in Appendix.

The seven unimodal benchmark functions: F1, F2, …, F7 (with only one optimum value) examine the exploitation strategy. In contrast, the multimodal functions: F8, F9, …, F13 (with many local optima) will check the exploration. The ten fixed low-dimension multimodal functions: F14, F15, …, F23 evaluate low-dimension exploration ability. Moreover, the functions F1 to F13 will be scaled up into 50, 100, 500, and 1000 dimensions to assess the scalability.

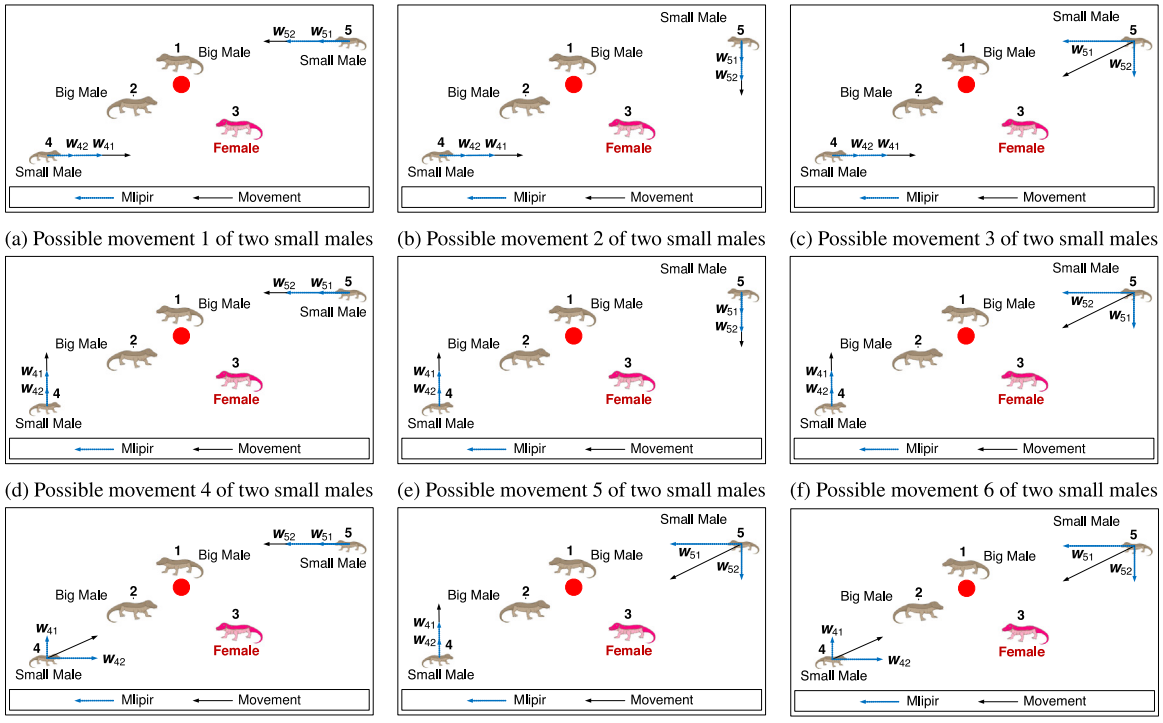
3.1. Parameter settings

The KMA is designed to has three easy-to-adjust parameters: population size n, portion p, and *mlipir* rate d. In the first phase of evolution, KMA uses a low population of five individuals ($n_1 = 5$), a half portion of big males ($p_1 = 0.5$), and a maximum *mlipir* rate ($d_1 = \frac{(m-1)}{m}$). This first phase is run for 1000 generations (or 5000 evaluations) to quickly obtain the global optimum solution to a simple (unimodal) problem. In the second phase, KMA is tuned to use a high population of two hundred individuals ($n_2 = 200$) with a self-adaptation procedure to automatically online-tune the population size in the interval between 20 and 200 during the evolution process ($n_{2,min} = 20$, $n_{2,max} = 200$), a half portion of big males ($p_2 = 0.5$), and a half *mlipir* rate ($d_2 = 0.5$). In the second phase, KMA is expected can solve more complex (multimodal) benchmark functions, which contain many local optima and flat areas. All those parameter settings are based on some preliminary experiments.

The optimum parameters for KMA and the competitors are summarized in Table 1. The first six settings are adopted from the previous researches in [9,11,36,37,37], and [39]. Here, all the parameter settings are fixed for all the benchmark functions to examine the scalability with a limited number of function evaluations of 25 000. GA is designed to use a high population $n = 100$ of real-encoding chromosomes, tournament-based parent selection with a tournament size of 10, a whole arithmetic crossover with probability $p_c = 0.8$, and a creep mutation with probability $p_m = 0.05$ to make it simple for thousand dimensions. SHADE is also designed to exploit a high population $n = 100$, with the probability to use the best individual of 0.1 and archive rate of 2 (meaning that the maximum size of the archive is $2n$). As described in [11], LSHADE-cnEpSin is recommended to use a variable population size of $18 \times m$. However, its population size is fixed to $18 \times 5 = 90$ since a big population size performs worse for the limited function evaluations.

3.2. Comparison to other optimizers

Next, KMA is compared to six algorithms: GA, SHADE, LSHADE-cnEpSin, EO, MPA, and SMA using the 23 benchmark functions illustrated in Fig. 8, where their detailed descriptions can be found in [36]. Here, the allowed number of evaluations (generated



(a) Possible movement 1 of two small males (b) Possible movement 2 of two small males (c) Possible movement 3 of two small males

(d) Possible movement 4 of two small males (e) Possible movement 5 of two small males (f) Possible movement 6 of two small males

(g) Possible movement 7 of two small males (h) Possible movement 8 of two small males (i) Possible movement 9 of two small males

Fig. 6. Nine possible movements of two small males.

Table 1
Parameter settings of KMA and the competitors.

Algorithm	Parameter
GA [37]	$n = 100$, tournament selection size = 10, whole arithmetic crossover $p_c = 0.8$, creep mutation $p_m = 0.05$
SHADE [9]	$n = 100$, Pbest $= 0.1$, Arc rate $= 2$
LSHADE-cnEpSin [11]	$n = 90$, $n_{min} = 4$, Pbest rate $= 0.11$, Arc rate $= 1.4$, Memory $= 5$, $p_s = 0.5$, $p_c = 0.4$
EO [36]	$n = 30$, $a_1 = 2$, $a_2 = 1$, GP $= 0.5$
MPA [37]	$n = 50$, FADs $= 0.2$, P $= 0.5$
SMA [39]	$n = 30$, $z = 0.03$
KMA	$n_1 = 5$, $n_2 = 200$, $n_{2,min} = 20$, $n_{2,max} = 200$, $p_1 = 0.5$, $p_2 = 0.5$, $d_1 = \frac{(m-1)}{m}$, $d_2 = 0.5$

individuals) for each benchmark function is limited to 25 000 with 30 independent runs to provide two meaningful statistical results: average solution (Avg) and standard deviation (Std).

Table 2 shows the findings of the evaluation using both metrics (Met): Avg and Std. The bold number shows the best result among all the competitors, the underscored one informs the global optimum solution is always reached by the algorithm with the Std equals to zero or less than the decimal digits of the global solution, and the bold-underscored one shows that it achieves the best and the global optimum solution as well.

Based on the Avg and Std, the proposed KMA guarantees the global optimum solutions for the most (17 of 23) functions: five unimodal, two high-dimension multimodal, and all the ten fixed low-dimension multimodal. It almost reaches the global optima for F7, F8, F10, F12, and F13. Impressively, although F7 has a high-level noise, KMA can reach a near-global optimum. KMA gives a solution significantly different from the global optimum only for F5. The function F5 is quite hard to solve by KMA and all the competitors since it has vast flat areas (plateaus) that make all the algorithms trapped in stagnation.

Meanwhile, MPA guarantees to get the global optimum solutions for thirteen functions: one unimodal, two high-dimension multimodal, and all the ten fixed low-dimension multimodal. It produces the near-global optima for eight functions and the solutions far from the global optima for two functions (F5 and F8). SMA guarantees to find the global optima for ten functions. It gives solutions close to the global optima for the twelve functions and worse solutions only for F5. Next, EO guarantees to reach the global optima only for six functions. It produces solutions close to the global optima for twelve functions and much worse solutions for five functions. The three other algorithms SHADE, LSHADE-cnEpSin, and GA produce much worse results.

Based on a statistical measure named Friedman Mean Rank (FMR), GA gives the worst FMR of 5.13 following by LSHADE-cnEpSin and SHADE that achieve 4.57 and 3.43, respectively. EO, MPA, and SMA produce better ranks of 3.17, 2.35, and 2.13, respectively. Finally, KMA achieves the best rank of 1.52, as illustrated in Table 2. In general, KMA achieves better (or equal) results than all the others for 18 out of 23 functions. It produces a slightly worse solution than SMA only for F7 and F8 and much worse than the competitors only for F5, F12, and F13. These results show the superiority of KMA.

In terms of the global optimum guaranty (Gua), which is defined as the certainty of finding the global optimum with Std equals zero or more negligible than the decimal digits of the global optimum, KMA is also much better than the competitors. It guarantees to reach the global optimum solutions for 17 of 23 (73.91%) benchmark functions, which is much higher than GA, SHADE, LSHADE-cnEpSin, EO, MPA, and SMA that provides guaranties of 5 of 23 (21.74%), 9 of 23 (39.13%), 5 of 23 (21.74%), 7 of 23 (30.43%), 13 of 23 (56.52%), and 10 of 23 (43.48%).

Table 2
Comparison of KMA and the competitors for thirteen 50-dimensional (F1 to F13) and ten fixed low-dimensional (F14 to F23) benchmark functions for 30 runs, each 25 000 function evaluations. Both symbols − and + (in the parentheses) inform that the current result is significantly worse and better than the KMA's result based on the Wilcoxon's rank sum test (Wil) with a significance level of 0.05, respectively, while the symbol ≈ denotes an insignificant result.

Func	Met	GA	SHADE	LSHADE-cnEpSin	EO	MPA	SMA	KMA
F1	Avg	4.380E+ 01 (−)	5.484E−05 (−)	3.136E−06 (−)	2.334E−86 (−)	4.138E−21 (−)	0 (≈)	0
	Std	5.257E+ 00	2.497E−05	3.016E−06	7.436E−86	4.086E−21	0	0
F2	Avg	3.063E+ 00 (−)	1.263E−02 (−)	1.314E−02 (−)	7.094E−49 (−)	3.256E−12 (−)	5.945E−174 (−)	0
	Std	2.347E−01	3.780E−03	1.202E−02	1.583E−48	2.494E−12	0.000E+ 00	0
F3	Avg	4.114E+ 03 (−)	4.717E+ 02 (−)	6.163E+ 01 (−)	6.547E−13 (−)	4.149E−02 (−)	0 (≈)	0
	Std	8.527E+ 02	1.674E+ 02	2.887E+ 01	3.408E−12	8.361E−02	0	0
F4	Avg	2.708E+ 00 (−)	1.066E+ 00 (−)	4.544E+ 00 (−)	2.193E−24 (−)	2.787E−08 (−)	6.137E−178 (−)	0
	Std	3.018E−01	3.224E−01	1.151E+ 00	2.657E−24	1.409E−08	0.000E+ 00	0
F5	Avg	6.347E+ 02 (−)	5.217E+ 01 (−)	6.054E+ 01 (−)	4.480E+ 01 (+)	4.532E+ 01 (+)	**7.075E+ 00** (+)	4.831E+ 01
	Std	1.300E+ 02	1.958E+ 01	2.401E+ 01	2.024E−01	4.179E−01	1.401E+ 01	1.820E−01
F6	Avg	4.690E+ 01 (−)	0 (≈)	1.230E+ 01 (−)	0 (≈)	0 (≈)	0 (≈)	0
	Std	9.159E+ 00	0	7.666E+ 00	0	0	0	0
F7	Avg	8.233E−02 (−)	2.189E−02 (−)	1.903E−02 (−)	1.135E−03 (−)	1.377E−03 (−)	**1.456E−04** (+)	1.715E−04
	Std	3.496E−02	4.702E−03	7.689E−03	4.641E−04	7.427E−04	1.583E−04	1.188E−04
F8	Avg	−1.538E+ 04 (−)	−1.059E+ 04 (−)	−1.877E+ 04 (+)	−1.491E+ 04 (−)	−1.455E+ 04 (−)	−2.095E+ 04 (+)	−1.701E+ 04
	Std	7.506E+ 02	5.241E+ 02	3.637E+ 02	8.352E+ 02	6.191E+ 02	6.814E−01	2.453E+ 03
F9	Avg	1.213E+ 02 (−)	2.617E+ 02 (−)	2.686E+ 01 (−)	0 (≈)	0 (≈)	0 (≈)	0
	Std	1.509E+ 01	1.230E+ 01	5.255E+ 00	0	0	0	0
F10	Avg	2.588E+ 00 (−)	1.605E−03 (−)	1.499E+ 00 (−)	4.441E−15 (−)	1.060E−11 (−)	**8.882E−16** (≈)	**8.882E−16**
	Std	1.595E−01	5.147E−04	4.012E−01	0.000E+ 00	5.925E−12	0.000E+ 00	0
F11	Avg	1.391E+ 00 (−)	3.601E−04 (−)	5.434E−03 (−)	0 (≈)	0 (≈)	0 (≈)	0
	Std	7.608E−02	1.366E−03	7.676E−03	0	0	0	0
F12	Avg	1.057E−01 (−)	2.077E−03 (+)	2.281E−02 (−)	**2.083E−07** (+)	9.452E−04 (+)	4.413E−03 (−)	2.799E−03
	Std	3.319E−02	1.136E−02	5.030E−02	2.463E−07	1.241E−03	6.600E−03	1.815E−03
F13	Avg	2.253E+ 00 (−)	**4.633E−05** (+)	2.558E−02 (+)	6.054E−02 (−)	7.316E−02 (−)	5.307E−03 (+)	5.270E−02
	Std	3.612E−01	2.131E−05	4.703E−02	5.858E−02	6.265E−02	4.593E−03	4.689E−02
F14	Avg	9.980E−01 (≈)	9.980E−01 (≈)	3.925E+ 00 (−)	1.064E+ 00 (−)	9.980E−01 (≈)	9.980E−01 (≈)	9.980E−01
	Std	3.011E−05	4.123E−17	4.659E+ 00	3.622E−01	1.515E−16	3.680E−13	4.382E−16
F15	Avg	1.815E−03 (−)	3.075E−04 (≈)	5.947E−04 (−)	1.076E−03 (−)	3.075E−04 (≈)	5.400E−04 (−)	3.075E−04
	Std	2.472E−03	1.135E−15	1.859E−04	3.650E−03	5.890E−15	3.337E−04	5.878E−15
F16	Avg	−1.032E+ 00 (≈)	−1.032E+ 00 (≈)	−1.004E+ 00 (−)	−1.032E+ 00 (≈)	−1.032E+ 00 (≈)	−1.032E+ 00 (≈)	−1.032E+ 00
	Std	8.681E−06	8.192E−06	1.490E−01	9.888E−06	9.302E−06	9.342E−06	9.779E−06
F17	Avg	3.979E−01 (≈)	3.979E−01 (≈)	3.979E−01 (≈)	3.979E−01 (≈)	3.979E−01 (≈)	3.979E−01 (≈)	3.979E−01
	Std	2.863E−05	3.269E−05	2.967E−05	3.401E−05	3.393E−05	3.024E−05	3.514E−05
F18	Avg	3.000E+ 00 (≈)	3.000E+ 00 (≈)	3.989E+ 00 (−)	3.000E+ 00 (≈)	3.000E+ 00 (≈)	3.000E+ 00 (≈)	3.000E+ 00
	Std	4.252E−05	1.327E−14	4.937E+ 00	1.473E−14	1.714E−14	1.601E−11	4.103E−10
F19	Avg	−3.861E+ 00 (≈)	−3.861E+ 00 (≈)	−3.861E+ 00 (≈)	−3.861E+ 00 (≈)	−3.861E+ 00 (≈)	−3.861E+ 00 (≈)	−3.861E+ 00
	Std	7.195E−04	6.576E−04	6.727E−04	1.273E−03	6.650E−04	7.448E−04	8.201E−04
F20	Avg	−3.289E+ 00 (−)	−3.317E+ 00 (+)	−3.320E+ 00 (≈)	−3.271E+ 00 (−)	−3.320E+ 00 (≈)	−3.238E+ 00 (−)	−3.320E+ 00
	Std	5.283E−02	2.145E−02	3.996E−04	6.234E−02	3.105E−04	5.478E−02	3.165E−04
F21	Avg	−7.326E+ 00 (−)	−9.9848 (−)	−10.1532 (≈)	−9.9618 (−)	−10.1532 (≈)	−10.1530 (−)	−10.1532
	Std	3.569E+ 00	9.224E−01	6.224E−15	9.236E−01	4.687E−11	1.528E−04	6.452E−15
F22	Avg	−8.507E+ 00 (−)	−10.4029 (≈)	−10.4029 (≈)	−9.3141 (−)	−10.4029 (≈)	−10.4027 (−)	−10.4029
	Std	3.233E+ 00	8.105E−06	1.117E−05	2.219E+ 00	9.030E−06	1.298E−04	1.014E−05
F23	Avg	−8.165E+ 00 (−)	−10.5364 (≈)	−10.0985 (−)	−10.5294 (−)	−10.5364 (≈)	−10.5362 (−)	−10.5364
	Std	3.460E+ 00	2.130E−06	1.694E+ 00	3.847E−02	2.172E−06	1.321E−04	2.538E−06
Friedman		5.13	3.43	4.57	3.17	2.35	2.13	**1.52**
Rank		7	5	6	4	3	2	**1**
Wil (−)		18	11	16	14	8	8	
Wil (≈)		5	9	5	7	13	11	
Wil (+)		0	3	2	2	2	4	
Gua (%)		21.74	39.13	21.74	30.43	56.52	43.48	**73.91**

3.3. Convergence curves analysis

A detailed evaluation is then performed on the convergence curves. First, the convergence curves are analyzed to the unimodal functions (F1 to F7). Fig. 7(a) shows the convergence curves of the seven algorithms: GA, SHADE, LSHADE-cnEpSin, EO, MPA, SMA, and KMA for the function F1. The vertical and horizontal axes represent the *log* of the average solution and the generation (iteration), respectively. Since those algorithms have various population sizes, they use different step sizes of 9, 9, 10, 30, 18, 30, and 180, respectively, on the horizontal axis to get fairness. It can be seen that KMA converges most quickly at the beginning generation. This result also happens on the five other unimodal functions: F2, F3, F4, F6, and F7. KMA also converges swiftly for F5 in the beginning generations. It shows that the three proposed movements work very well in the first phase of evolution. Unfortunately, it gets a stagnation until the end of evolution and obtains a worse local optimum at around 1.68 (*log* 48.31), as illustrated in Fig. 7(b). The function F5 is challenging due to the flat areas, making KMA (and all other algorithms) are trapped in stagnation. This curve indicates that the high population in the second phase of KMA evolution cannot do proper coordination in the flat fitness landscape. It happens since most (or even all) individuals have the same fitness values (qualities), making KMA challenging to split the population into big males, females, and small males based on their qualities.

The convergence curves are then analyzed to the multimodal functions (F8 to F13). Fig. 7(c) illustrates the convergence curves of all the algorithms for F8. Similar to SMA, the proposed KMA converges quickly than the others in the beginning generations. Unfortunately, it gets some stagnations and reaches a worse solution than SMA that obtains the best one. In contrast, the

other algorithms converge much slower and give much worse solutions. Meanwhile, Fig. 7(d) shows the convergence curves for F9. In this case, KMA converges most swiftly compared to the competitors. It reaches the minimum solution of 0 (global optimum) in the beginning generations, with the lowest mean function evaluations of 145.33, where the red-dotted line is disappeared since $log(0)$ is infinite. In contrast, EO, MPA, and SMA converge more slowly while GA, SHADE, LSHADE-cnEpSin cannot reach the global optimum. This convergence curve also occurs on F11.

The convergence analysis is finally performed for the ten fixed-dimension multimodal functions. Fig. 7(e) plots the convergence curves for F17. All the algorithms can converge to the global solution. In this case, KMA converges more slowly in the beginning generations since it only employs five individuals in the first phase of evolution. The five individuals have difficulty coordinating in some flat areas in F17. Once it changes the population size into 200 individuals in the second phase of evolution, it gets the global optimum swiftly. A similar result also happens on F15, F16, and F18. Meanwhile, Fig. 7(f) shows the convergence curves for F19. Similar to the curve of F17, KMA also converges more slowly in the early generations since, in the first phase of evolution, it only uses five individuals that have difficulty coordinating in the wide-flat fitness landscapes in F19. This result also applies to F14, F20, F21, F22, and F23.

Hence, it can be highlighted that all those convergence curves indicate that KMA is the most efficient metaheuristic algorithm among the competitors. These results prove that all the three movements (HILE, MIME, and LIHE strategies) and the population adaptation scheme proposed in KMA can provide an exploitation–exploration balance and highly guarantee global optimum solutions to various benchmark functions.

3.4. Scalability analysis

The scalability is analyzed by examining KMA performance compared to the competitors to optimize the high-dimensional versions of the thirteen functions: F1 to F13, where their dimensions (Dim) are scaled up from 50 into 100, 500, and 1000. The allowed number of generated individuals (function evaluations) is also limited to 25 000 with 30 independent runs to get the average (Avg), standard deviation (Std), and mean function evaluations (MFE). Tables 3, 4, and 5 illustrate the results. The bold number represents the best result among the competitors. The underscored one shows the global optimum solution is always reached with the Std equals to zero or less than the decimal digits of the global optimum. The bold-underscored one informs that it achieves the best and the global optimum solution as well.

In all high-dimensions (100, 500, and 1000), KMA guarantees to get the global optimum for 7 out of 13 (53.85%) benchmark functions, which is much higher than SMA, EO, MPA, SHADE, LSHADE-cnEpSin, and GA that give guarantees 38.46%, 23.08%, 23.08%, 0%, 0%, and 0%, respectively. Moreover, KMA impressively requires a tiny MFE. For instance, it needs only 55.83, 62.33, and 65.17 MFE to get the global optimum solution for the 100, 500, and 1000-dimensions unimodal function F6, which is much fewer than EO, MPA, and SMA that require more than 1000 MFE.

For 5 out of 7 unimodal functions, KMA is stable and guarantees the global optimum solutions (with Std $= 0$). Impressively, it is stable with low MFE (from 55.83 to 4221.67) for all the dimensions, much lower than all the competitors. For F7, KMA reaches a near-global solution with a low Std, which is much better than GA, SHADE, LSHADE-cnEpSin, EO, and MPA. It is competitive to the SMA. It is slightly unstable only for F5. Fortunately, this result is better than SHADE and LSHADE-cnEpSin and competitive to EO and MPA. Hence, these results imply that KMA is much better than the competitors in terms of exploitation ability and stability in optimizing the high-dimensional unimodal functions.

Meanwhile, for two multimodal functions: F9 and F11, KMA guarantees the global optimum with a minimum Std (zero) for all dimensions. Moreover, it needs extremely low MFEs (from 150.50 to 183.00). For the function F10, KMA is also relatively stable to find the near-global solutions with a low Std, which is better than (or the same as) the competitors. For F12 and F13, although stable for all the high-dimensions, KMA is a little worse than SMA.

Statistically, for both 500 and 1000 dimensions, KMA gives the best Friedman Mean Rank (FMR), as illustrated in Tables 4 and 5. Meanwhile, for 100 dimensions, KMA gives a lower FMR than SMA, as illustrated in Table 3. Interestingly, it can be highlighted that in terms of the global optimum guaranty, KMA is much better than all the competitors. For 50, 100, 500, and 1000 dimensions, KMA is stable to give a guaranty of the global optimum solutions for 7 of 13 (53.85%) benchmark functions, which is much higher than SMA, EO, and MPA that are unstable: the guaranties decrease from 38.46% to 23.08%, from 23.08% to 15.38%, and from 23.08% to 15.38%, respectively, and is much better than GA, SHADE, and LSHADE-cnEpSin that do not provide any guaranty (0%). These results empirically prove that KMA has high scalability and robustness in the exploitation–exploration balance.

4. Discussion

As with other swarm-based algorithms, it is hard to provide mathematical proof to show that KMA is capable of solving both unimodal and multimodal problems effectively and efficiently since it uses real number encoded individuals. Until now, only GA with binary-encoding individuals can be mathematically explained using Holland's schema theorem, and the building block hypothesis [50]. However, the binary-encoding is not scalable to optimize the thousands-dimensional problems needing high precision due to some reasons: managing long binary chromosomes is too complex in space, decoding long chromosomes and then evaluating fitness is time-consuming, mating pairs of long parent chromosomes based on a single-point crossover only gives low exploitation but creating a better multi-point crossover or other schemes is too hard, and mutating a few binary genes will produce a low exploration. Hence, the best individual representation for GA is the real number encoding, like used in KMA and other algorithms. Consequently, the performances of GA, KMA, and all the others can only be explained in this paper using qualitative (subjective) analysis of exploitation and exploration balance.

First, the qualitative analysis is performed for GA that produces the lowest performance. A tournament selection creates a proportional selection of parents based on their fitness regardless of negative, zero, or positive values. Our experiments show that the tournament size of 10 individuals, which is 10% of the population size of 100, gives the optimum parent selection. Next, since GA uses the real number encoding chromosomes, the whole arithmetic crossover and creep mutation can be considered to give medium exploitation medium exploration, respectively. Our experimental results show that they are better than the single-point crossover and mutation in binary or integer number encodings. Furthermore, Table 2 shows that GA can guarantee global optima for 5 out of 10 (50%) fixed low-dimensional benchmark functions (F14 and F16 to F19) and reach near global optima for the five others. However, GA fails to optimize the thirteen high-dimensional functions, even for 50 dimensions. These results indicate that the whole arithmetic crossover is not able to create high exploitation to solve the seven unimodal functions (F1 to F7). Likewise, the creep mutation cannot provide a high exploration to optimize the six multimodal functions (F8 to F13). In other words, all those operators cannot provide scalability to GA.

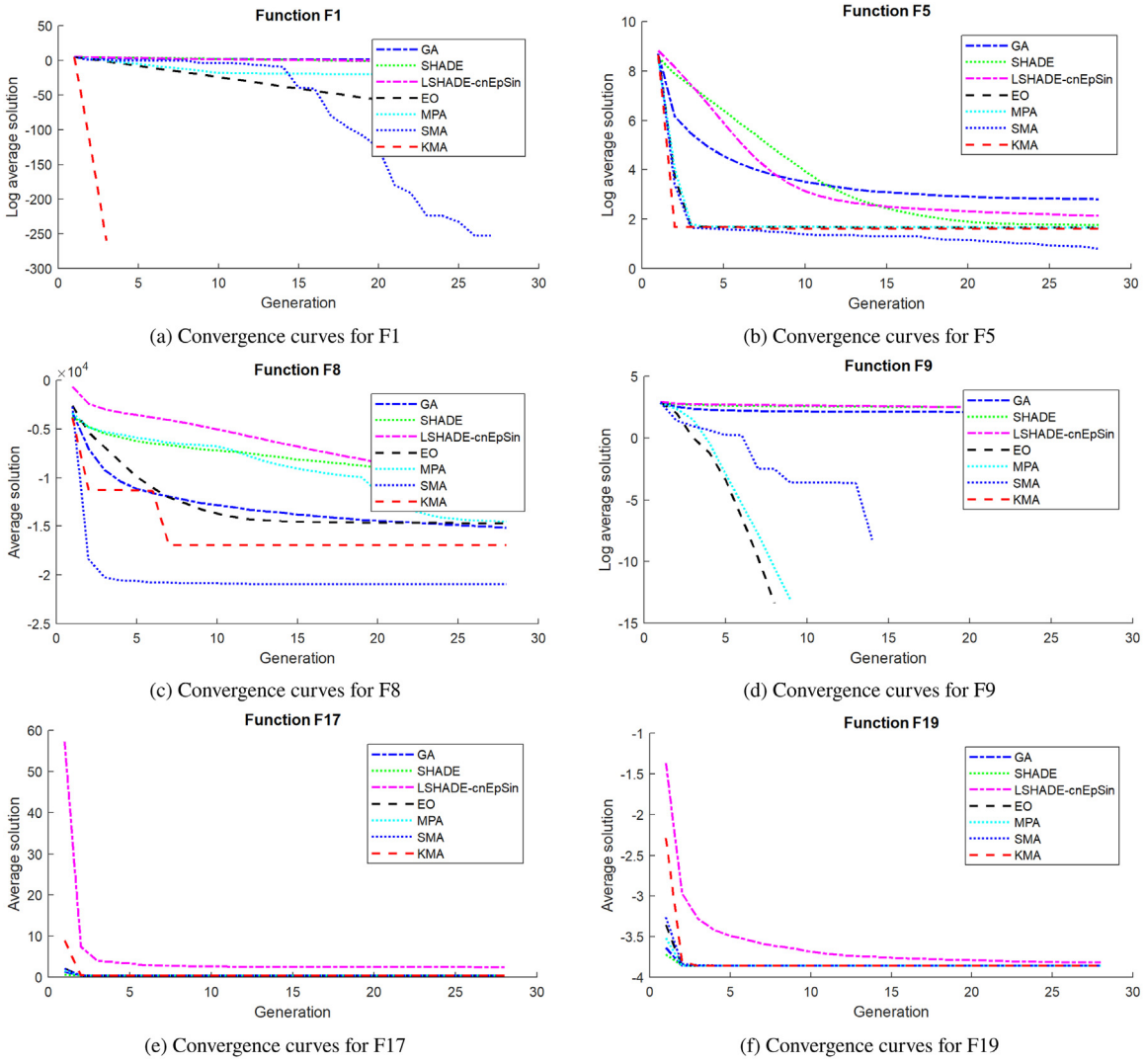


Fig. 7. Converge curves of KMA and the competitors for six representative functions for 30 independent runs, each 25 000 function evaluations.

Furthermore, SHADE guarantees global optima for 8 out of 10 (80%) fixed low-dimensional benchmark functions and reaches near global optima for the rests while LSHADE-cnEpSin only 50%, as shown in Table 2. However, both SHADE and LSHADE-cnEpSin also cannot give scalability. In 50-dimensional benchmarks, SHADE guarantees the global optimum only for F6 while LSHADE-cnEpSin does not give any guaranty, as shown in Table 2. In higher dimensions of 100, 500, and 1000, both algorithms do not provide any guaranty, as depicted in Tables 3, 4, and 5. These facts indicate the differential mutations in DE, the base model used in both algorithms, cannot give high exploitation and high exploration. They are calculated from indirect vectors of two to four best or randomly selected individuals [49] so that the low-quality individuals cannot directly follow the best one.

Meanwhile, EO obtains lower FMR than GA, SHADE, and LSHADE-cnEpSin. Although it guarantees global optima only for 40% of the fixed low-dimensional functions (F16 to F19), it interestingly gives scalability for 2 out of 13 (15.38%) high-dimensional benchmarks (F6 and F9), as can be seen in Table 5. These results can be subjectively explained as follow. EO splits the population of thirty particles (candidate solutions) into two groups with different search strategies. First, four high-quality particles (as equilibrium candidates) with their arithmetic mean do high exploitation. Second, twenty-six low-quality particles do a high exploration. For the unimodal functions, the equilibrium candidates quickly find the global optimum of F6, even for 1000 dimensions, but fail for the six others. Meanwhile, for the multimodal benchmarks, the low-quality particles can explore to cover the fitness landscape, and the equilibrium candidates can quickly find the global optimum of F9 for all the dimensions. However, for the five other multimodal functions, they fail. This issue arises because EO lacks an adaptive strategy for automatically tuning the number of equilibrium candidates. As explained in [36], three equilibrium candidates make EO more exploitative to quickly find the global optima of unimodal functions. In contrast, five or more equilibrium candidates make EO more explorative to find the global optima of multimodal functions rapidly. Unfortunately, the type of the given function is unknown, and EO has no procedure to automatically detect the function type. Creating such a procedure is quite challenging. The other problem probably comes from the arithmetic mean calculated from the four equilibrium candidates. It can make EO converge too early (premature) and trapped in local optima in the cases of multimodal benchmarks.

Next, MPA achieves better performance than EO. It guarantees global optima for all fixed low-dimensional functions (F14 to F23) and also provides scalability for 2 out of 13 (15.38%) high-dimensional benchmarks (F6 and F9). This achievement can be subjectively explained as follow. First, MPA employs 50 individuals: 25 high-quality predators and 25 low-quality prey with

Table 3
Comparison of the six algorithms for thirteen 100-dimensional benchmark functions for 30 runs, each 25 000 function evaluations. Both symbols − and + (in the parentheses) show that the current result is significantly worse and better than the KMA's result based on the Wilcoxon's rank sum test with a significance level of 0.05, respectively, while ≈ denotes an insignificant result.

Func	Met	GA	SHADE	LSHADE-cnEpSin	EO	MPA	SMA	KMA
F1	Avg	7.638E+ 02 (−)	2.432E−01 (−)	2.697E+ 00 (−)	1.166E−74 (−)	4.929E−19 (−)	0 (≈)	0
	Std	6.477E+ 01	8.656E−02	1.040E+ 00	1.535E−74	3.687E−19	0	0
	MFE	25000	25000	25001	25020	25000	21696	2087.17
F2	Avg	1.950E+ 01 (−)	8.981E−01 (−)	3.658E+ 00 (−)	7.151E−43 (−)	2.030E−11 (−)	4.393E−178 (−)	0
	Std	1.131E+ 00	3.456E−01	8.951E−01	8.133E−43	1.244E−11	0.000E+ 00	0
	MFE	25000	25000	25001	25020	25000	25007	3922.5
F3	Avg	2.453E+ 04 (−)	1.257E+ 04 (−)	2.629E+ 03 (−)	6.080E−05 (−)	4.333E+ 00 (−)	0 (≈)	0
	Std	5.323E+ 03	2.378E+ 03	6.532E+ 02	2.100E−04	5.581E+ 00	0	0
	MFE	25000	25000	25001	25020	25000	22382	2464.33
F4	Avg	8.272E+ 00 (−)	1.052E+ 01 (−)	1.242E+ 01 (−)	3.241E−07 (−)	2.702E−07 (−)	9.698E−160 (−)	0
	Std	5.292E−01	1.221E+ 00	1.539E+ 00	1.774E−06	1.468E−07	5.312E−159	0
	MFE	25000	25000	25001	25020	25000	25015	4144.17
F5	Avg	2.932E+ 04 (−)	3.233E+ 02 (−)	6.299E+ 02 (−)	9.531E+ 01 (≈)	9.626E+ 01 (−)	**9.883E+ 00**(+)	9.530E+ 01
	Std	6.765E+ 03	8.492E+ 01	2.270E+ 02	6.363E−01	7.919E−01	1.878E+ 01	1.467E+ 01
	MFE	25000	25000	25001	25020	25000	25020	25011.33
F6	Avg	7.714E+ 02 (−)	4.300E+ 00 (−)	8.920E+ 01 (−)	0 (≈)	0 (≈)	0 (≈)	0
	Std	6.764E+ 01	2.718E+ 00	2.830E+ 01	0	0	0	0
	MFE	25000	24910	25001	1785	2060	1222	55.83
F7	Avg	2.641E−01 (−)	1.005E−01 (−)	1.037E−01 (−)	1.675E−03 (−)	1.483E−03 (−)	**2.025E−04** (+)	2.122E−04
	Std	9.003E−02	2.357E−02	2.673E−02	7.879E−04	6.133E−04	1.722E−04	1.393E−04
	MFE	25000	25000	25001	25020	25000	25020	25149.67
F8	Avg	−2.413E+ 04 (−)	−1.334E+ 04 (−)	−2.321E+ 04 (−)	−2.891E+ 04 (−)	−2.615E+ 04 (−)	−4.189E+ 04 (+)	−3.259E+ 04
	Std	1.268E+ 03	6.594E+ 02	6.562E+ 02	1.592E+ 03	9.166E+ 02	7.066E+ 00	4.765E+ 03
	MFE	25000	25000	25001	25020	25000	25020	25045.83
F9	Avg	5.109E+ 02 (−)	7.234E+ 02 (−)	3.496E+ 02 (−)	0 (≈)	0 (≈)	0 (≈)	0
	Std	3.177E+ 01	1.963E+ 01	2.985E+ 01	0	0	0	0
	MFE	25000	25000	25001	5895	6891.67	5210	150.5
F10	Avg	4.960E+ 00 (−)	7.185E−01 (−)	2.672E+ 00 (−)	5.033E−15 (−)	6.508E−11 (−)	**8.882E−16** (≈)	**8.882E−16**
	Std	1.284E−01	4.472E−01	3.237E−01	1.347E−15	2.833E−11	0.000E+ 00	0.000E+ 00
	MFE	25000	25000	25001	25020	25000	25020	25085
F11	Avg	8.017E+ 00 (−)	1.672E−01 (−)	7.733E−01 (−)	0 (≈)	0 (≈)	0 (≈)	0
	Std	4.267E−01	4.487E−02	1.330E−01	0	0	0	0
	MFE	25000	25000	25001	6611	7825	6700	169.83
F12	Avg	2.651E+ 00 (−)	3.780E−01 (−)	8.041E−01 (−)	5.759E−03 (−)	2.800E−02 (−)	**1.216E−03** (+)	2.407E−03
	Std	3.679E−01	2.931E−01	3.086E−01	5.622E−03	7.451E−03	2.333E−03	2.832E−03
	MFE	25000	25000	25001	25020	25000	25020	25012.33
F13	Avg	5.324E+ 01 (−)	8.230E−01 (−)	1.171E+ 01 (−)	3.255E+ 00 (−)	5.419E+ 00 (−)	**8.191E−02** (+)	1.741E−01
	Std	5.268E+ 00	7.507E−01	8.421E+ 00	1.135E+ 00	2.842E+ 00	2.222E−01	3.133E−01
	MFE	25000	25000	25001	25020	25000	25020	25009.17
Friedman		6.61	5.31	5.92	2.77	3.23	1.15	**1.38**
Rank		7	5	6	3	4	1	**2**
Wil (−)		13	13	13	9	10	2	
Wil (≈)		0	0	0	4	3	6	
Wil (+)		0	0	0	0	0	5	
Gua (%)		0	0	0	23.08	23.08	38.46	**53.85**

different movement strategies of exploitation and exploration, respectively. It uses the probability of the Fish Aggregating Devices (FADs) effects of 0.2, making predators take longer jumps (explorations) in different dimensions randomly in 20% of the number of iterations. This scheme can avoid trapping in local optima. However, since MPA equally divides the evolution into three phases: high exploration, equal exploration–exploitation, and high exploitation, it cannot guarantee global optima for the most unimodal functions. Additionally, it is susceptible to being stuck in the local optima for higher-dimensional benchmarks.

Meanwhile, SMA obtains a lower FMR than MPA. Although it guarantees global optima only for 5 out of 10 (50%) fixed low-dimensional functions (as shown in Table 2) but it gives scalability for 3 out of 13 (23.08%) high-dimensional benchmarks (F6, F9, and F11), as shown in Table 5. These results can be achieved since SMA uses an adaptive weight W to manage a particular disturbance rate while converging quickly. Hence, it can avoid local optima during the fast convergence. Unfortunately, it is not scalable for some benchmarks since it uses a vibration parameter to efficiently explore in early generation and highly exploit in the later iteration.

Finally, KMA significantly outperforms all the competitors. It guarantees global optima for all (100%) fixed low-dimensional functions, as shown in Table 2. Additionally, it guarantees global optima and also scalability for 7 out of 13 (53.85%) high-dimensional benchmarks, as depicted in Table 5. This result can be achieved since KMA employs three following features.

First, different from EO, MPA, and SMA that utilize high exploration low exploitation in early generations (first stage) and the opposite strategy in the end (second stage), KMA uses three searching strategies: HILE, MIME, and LIHE in the whole evolution (the first and the second stages). HILE is designed based on a much different scheme used in EO regarding the movement formula, probability, and less (one, two, or three) fittest individuals. It can perform higher exploitation and converge to the global optima more swiftly for the unimodal functions. Meanwhile, MIME is implemented using similar schemes to the operators used in GA. It employs the same whole arithmetic crossover as in GA but a slightly different mutation by introducing a mutation radius of 0.5 to make longer jumps. In contrast, LIHE is implemented using *mlipir* movement, which is composed of direct attraction vectors of partial dimensions of one to three randomly selected higher-quality individuals. As illustrated in Fig. 6, this *mlipir* movement logically provides a higher exploitation strategy than the differential mutations in DE.

Table 4
Comparison of the six algorithms for thirteen 500-dimensional benchmark functions for 30 runs, each 25 000 function evaluations. Both symbols − and + (in the parentheses) show that the current result is significantly worse and better than the KMA's result based on the Wilcoxon's rank sum test with a significance level of 0.05, respectively, while ≈ denotes an insignificant result.

Func	Met	GA	SHADE	LSHADE-cnEpSin	EO	MPA	SMA	KMA
F1	Avg	3.057E+ 04 (−)	1.498E+ 04 (−)	2.072E+ 04 (−)	1.587E−62 (−)	3.273E−16 (−)	0 (≈)	0
	Std	1.267E+ 03	1.718E+ 03	2.374E+ 03	3.301E−62	2.702E−16	0	0
	MFE	25000	25000	25000	25020	25000	22423	2139
F2	Avg	3.069E+ 02 (−)	2.112E+ 02 (−)	3.253E+ 02 (−)	5.498E−37 (−)	7.210E−10 (−)	5.355E−01 (−)	0
	Std	7.687E+ 00	1.482E+ 01	6.976E+ 01	4.340E−37	6.922E−10	2.104E+ 00	0
	MFE	25000	25000	25000	25020	25000	25020	3982.83
F3	Avg	9.780E+ 05 (−)	1.448E+ 06 (−)	1.508E+ 05 (−)	5.591E+ 04 (−)	5.933E+ 03 (−)	3.812E−303 (−)	0
	Std	1.362E+ 05	1.682E+ 05	2.820E+ 04	7.208E+ 04	3.310E+ 03	0.000E+ 00	0
	MFE	25000	25000	25000	25020	25000	23750	2539.83
F4	Avg	2.977E+ 01 (−)	3.982E+ 01 (−)	1.847E+ 01 (−)	9.501E+ 01 (−)	4.364E−05 (−)	6.143E−141 (−)	0
	Std	7.773E−01	1.579E+ 00	2.791E+ 00	6.427E+ 00	4.672E−05	2.412E−140	0
	MFE	25000	25000	25000	25020	25000	25020	4191.67
F5	Avg	6.899E+ 06 (−)	2.730E+ 06 (−)	2.238E+ 06 (−)	4.965E+ 02 (−)	4.965E+ 02 (−)	**9.652E+ 01** (+)	4.715E+ 02
	Std	6.357E+ 05	5.643E+ 05	9.571E+ 05	5.158E−01	3.831E−01	1.261E+ 02	7.375E+ 01
	MFE	25000	25000	25000	25020	25000	25020	25045.33
F6	Avg	3.069E+ 04 (−)	1.475E+ 04 (−)	2.005E+ 04 (−)	0 (≈)	0 (≈)	0 (≈)	0
	Std	1.397E+ 03	1.405E+ 03	2.008E+ 03	0	0	0	0
	MFE	25000	25000	25000	2437	2451.67	1312	62.33
F7	Avg	5.312E+ 01 (−)	1.950E+ 01 (−)	2.130E+ 01 (−)	2.747E−03 (−)	1.931E−03 (−)	2.725E−04 (−)	**2.643E−04**
	Std	5.177E+ 00	3.283E+ 00	2.040E+ 01	1.042E−03	6.748E−04	2.671E−04	1.781E−04
	MFE	25000	25000	25000	25020	25000	25020	25103.33
F8	Avg	−6.522E+ 04 (−)	−2.556E+ 04 (−)	−3.126E+ 04 (−)	−9.022E+ 04 (−)	−8.763E+ 04 (−)	−2.094E+ 05 (+)	−1.664E+ 05
	Std	4.664E+ 03	1.780E+ 03	1.991E+ 03	4.873E+ 03	3.222E+ 03	1.806E+ 02	2.339E+ 04
	MFE	25000	25000	25000	25020	25000	25020	25057
F9	Avg	4.469E+ 03 (−)	4.773E+ 03 (−)	4.404E+ 03 (−)	0 (≈)	0 (≈)	0 (≈)	0
	Std	1.133E+ 02	4.481E+ 01	1.162E+ 02	0	0	0	0
	MFE	25000	25000	25000	6792	7216.67	7546	157.67
F10	Avg	9.622E+ 00 (−)	7.980E+ 00 (−)	8.523E+ 00 (−)	7.046E−15 (−)	6.442E−10 (−)	**8.882E−16** (≈)	**8.882E−16**
	Std	1.319E−01	2.402E−01	3.125E−01	1.598E−15	1.770E−10	0.000E+ 00	0.000E+ 00
	MFE	25000	25000	25000	25020	25000	25020	25085
F11	Avg	2.763E+ 02 (−)	1.297E+ 02 (−)	1.829E+ 02 (−)	0 (≈)	0 (≈)	0 (≈)	0
	Std	1.158E+ 01	1.468E+ 01	2.420E+ 01	0	0	0	0
	MFE	25000	25000	25000	7996	10531.67	8790	179.5
F12	Avg	5.889E+ 04 (−)	3.252E+ 02 (−)	1.103E+ 02 (−)	3.622E−01 (−)	3.359E−01 (−)	**3.424E−03** (+)	1.197E−02
	Std	3.413E+ 04	6.120E+ 02	5.415E+ 02	1.788E−02	2.141E−02	5.155E−03	9.172E−03
	MFE	25000	25000	25000	25020	25000	25020	25010.67
F13	Avg	3.395E+ 06 (−)	5.172E+ 05 (−)	1.168E+ 05 (−)	4.885E+ 01 (−)	4.824E+ 01 (−)	**4.239E−01** (+)	2.263E+ 00
	Std	5.043E+ 05	2.135E+ 05	2.922E+ 05	3.293E−01	3.656E−01	7.088E−01	9.714E−01
	MFE	25000	25000	25000	25020	25000	25020	25011.67
Friedman		6.46	5.77	5.54	3.15	2.77	1.46	1.31
Rank		7	6	5	4	3	2	1
Wil (−)		13	13	13	10	10	4	
Wil (≈)		0	0	0	3	3	5	
Wil (+)		0	0	0	0	0	4	
Gua (%)		0	0	0	23.08	23.08	30.77	**53.85**

Second, KMA uses a procedure to detect the function type using a small population of 5 individuals in the first stage of evolution. This procedure is performed only for 100 generations (500 evaluations). If the fitness improvement rate is more than 0.5 (a simple function is detected), it continues the evolution process to the maximum of 1000 generations (5000 evaluations). This result indicates HILE and MIME can effectively exploit the bowl-like landscape and quickly go to the global optimum. Interestingly, it also shows that LIHE can help HILE and MIME to highly explore the multimodal landscape (contained in F9 and F11) so that KMA finds the global optimum in less than 40 generations (200 evaluations). Hence, it can be said that those three searching strategies are performed very well in this stage to guarantee global optima and scalability for 53.85% of high-dimensional benchmarks efficiently in less than 1000 generations (5000 evaluations). In contrast, if the fitness improvement rate is less than 0.5 (a complex function is detected), it jumps to the second stage.

Third, in the second stage of evolution, KMA utilizes a self-adaptation of a big population (20 to 200 individuals) based on their fitness to provide an exploitation–exploration balance. Adding five individuals, randomly generated from the best-so-far solution, to the population increases the exploration rate to avoid the local optima or escape from the flat fitness landscapes. In contrast, removing five individuals increase the exploitation rate to find the global optimum swiftly. Therefore, this second stage can guarantee global optima for all (100%) fixed low-dimensional benchmarks.

However, KMA has a limitation for the problems with wide flat areas, such as the function F5. It is caused by the two fixed parameters in the first stage of evolution: a half big male portion $p_1 = 0.5$ and a high *mlipir* rate $d_1 = \frac{(m-1)}{m}$. The wide flat areas make KMA has difficulty splitting the five individuals in the first stage into big males, female, and small males since they may have very similar (even the same) fitness. Hence, HILE cannot effectively do the high exploitation. Instead, it works randomly and selects any big male as a winner. Consequently, with the randomly selected winner, the female cannot do the MIME strategy. In contrast, small males are not affected by this situation since they do LIHE by randomly selecting the big males to *mlipir*. Likewise, in the second stage, those three strategies also cannot work very well because of the two fixed parameters: a half big male portion $p_2 = 0.5$ and a high *mlipir* rate $d_2 = 0.5$ throughout the evolution, which make KMA cannot do higher exploration.

Table 5
Comparison of the six algorithms for thirteen 1000-dimensional benchmark functions for 30 runs, each 25 000 function evaluations. Both symbols − and + (in the parentheses) show that the current result is significantly worse and better than the KMA's result based on the Wilcoxon's rank sum test with a significance level of 0.05, respectively, while ≈ denotes an insignificant result.

Func	Met	GA	SHADE	LSHADE-cnEpSin	EO	MPA	SMA	KMA
F1	Avg	1.104E+ 05 (−)	9.927E+ 04 (−)	5.959E+ 04 (−)	2.879E−59 (−)	1.718E−15 (−)	3.413E−292 (−)	0
	Std	4.031E+ 03	5.047E+ 03	9.258E+ 03	6.916E−59	1.340E−15	0.000E+ 00	0
	MFE	25000	25000	25001	25020	25000	23518	2171.17
F2	Avg	8.338E+ 02 (−)	6.554E+ 04 (−)	6.554E+ 04 (−)	1.644E−35 (−)	1.136E+ 03 (−)	1.223E+ 00 (−)	0
	Std	1.458E+ 01	6.554E+ 04	6.554E+ 04	1.931E−35	1.490E+ 02	4.522E+ 00	0
	MFE	25000	25000	25001	25020	25000	25020	4021
F3	Avg	4.055E+ 06 (−)	6.989E+ 06 (−)	1.431E+ 06 (−)	1.001E+ 05 (−)	3.510E+ 04 (−)	2.115E−240 (−)	0
	Std	6.387E+ 05	8.208E+ 05	3.607E+ 05	1.306E+ 05	1.980E+ 04	0.000E+ 00	0
	MFE	25000	25000	25001	25020	25000	2.398E+ 04	2650.5
F4	Avg	4.335E+ 01 (−)	4.903E+ 01 (−)	3.560E+ 01 (−)	9.943E+ 01 (−)	3.442E−04 (−)	7.211E−147 (−)	0
	Std	1.047E+ 00	2.192E+ 00	1.165E+ 01	2.702E−01	4.182E−04	3.950E−146	0
	MFE	25000	25000	25001	25020	25000	25020	4221.67
F5	Avg	4.296E+ 07 (−)	3.468E+ 07 (−)	1.271E+ 07 (−)	9.965E+ 02 (−)	9.962E+ 02 (−)	**1.374E+ 02** (+)	9.617E+ 02
	Std	2.898E+ 06	4.123E+ 06	1.760E+ 07	1.888E−01	2.049E−01	2.225E+ 02	8.468E+ 01
	MFE	25000	25000	25001	25020	25000	25020	25040.67
F6	Avg	1.087E+ 05 (−)	1.046E+ 05 (−)	5.698E+ 04 (−)	0 (≈)	0 (≈)	0 (≈)	0
	Std	3.581E+ 03	7.069E+ 03	8.936E+ 03	0	0	0	0
	MFE	25000	25000	25001	2673	2626.67	1919	65.17
F7	Avg	6.923E+ 02 (−)	4.583E+ 02 (−)	2.687E+ 02 (−)	3.268E−03 (−)	1.962E−03 (−)	4.483E−04 (−)	**2.731E−04**
	Std	5.411E+ 01	7.016E+ 01	4.823E+ 02	1.420E−03	9.409E−04	3.325E−04	1.411E−04
	MFE	25000	25000	25001	25020	25000	25020	25123.33
F8	Avg	−9.853E+ 04 (−)	−3.526E+ 04 (−)	−3.573E+ 04 (−)	−1.366E+ 05 (−)	−1.368E+ 05 (−)	−4.189E+ 05 (+)	−3.279E+ 05
	Std	7.907E+ 03	1.707E+ 03	2.873E+ 03	7.654E+ 03	5.031E+ 03	1.680E+ 02	4.762E+ 04
	MFE	25000	25000	25001	25020	25000	25020	25059.67
F9	Avg	9.707E+ 03 (−)	9.976E+ 03 (−)	9.485E+ 03 (−)	0 (≈)	0 (≈)	0 (≈)	0
	Std	1.339E+ 02	9.388E+ 01	1.358E+ 02	0	0	0	0
	MFE	25000	25000	25001	7169	7386.67	8253	157.83
F10	Avg	1.155E+ 01 (−)	1.152E+ 01 (−)	9.241E+ 00 (−)	7.638E−15 (−)	1.232E−09 (−)	**8.882E−16** (≈)	**8.882E−16**
	Std	1.381E−01	2.372E−01	3.882E−01	1.084E−15	4.100E−10	0.000E+ 00	0.000E+ 00
	MFE	25000	25000	25001	25020	25000	25020	25085
F11	Avg	9.932E+ 02 (−)	9.272E+ 02 (−)	5.022E+ 02 (−)	3.701E−18 (−)	9.992E−17 (−)	0 (≈)	0
	Std	3.161E+ 01	7.810E+ 01	8.310E+ 01	2.027E−17	3.388E−17	0	0
	MFE	25000	25000	25001	10828	24193.33	7146	183
F12	Avg	5.097E+ 06 (−)	4.692E+ 05 (−)	2.014E+ 05 (−)	6.439E−01 (−)	5.707E−01 (−)	**3.577E−03** (+)	1.333E−02
	Std	8.851E+ 05	2.195E+ 05	8.138E+ 05	2.257E−02	2.200E−02	1.080E−02	1.293E−02
	MFE	25000	25000	25001	25020	25000	25020	25010.67
F13	Avg	5.130E+ 07 (−)	2.395E+ 07 (−)	4.123E+ 06 (−)	9.914E+ 01 (−)	9.776E+ 01 (−)	**2.472E+ 00** (+)	4.218E+ 00
	Std	6.384E+ 06	5.036E+ 06	1.086E+ 07	2.294E−01	4.164E−01	6.227E+ 00	2.215E+ 00
	MFE	25000	25000	25001	25020	25000	25020	25013
Friedman		6.31	6.23	5.08	3.38	3.08	1.46	**1.31**
Rank		7	6	5	4	3	2	**1**
Wil (−)		13	13	13	11	11	5	
Wil (≈)		0	0	0	2	2	4	
Wil (+)		0	0	0	0	0	4	
Gua (%)		0	0	0	15.38	15.38	23.08	**53.85**

Finally, it can be summarized that KMA can give a guaranty and scalability to quickly find the global optima for unimodal (or bowl-like multimodal) functions during the first phase of evolution since it employs three different strategies: HILE, MIME, and LIHE with a small population and a high *mlipir* rate. KMA can also guarantee global optima for various complex fixed low-dimensional multimodal functions since, in the second phase of evolution, it uses an adaptively-tuned big population doing those three different strategies with a half *mlipir* rate. These qualitative analyses have been clearly visualized in the six convergence curves of representative benchmark functions in Section 3.3.

5. Conclusion

In this research, a novel metaheuristic optimizer named Komodo Mlipir Algorithm (KMA) has been successfully developed to provide a higher guaranty to reach global optima and more scalability to a thousand dimensions. Compared to recent algorithms (as competitors in this research), KMA has three advantages. First, during two stages of evolution, it employs three groups of individuals: high-quality big males, middle-quality females, and low-quality small males that work in different ways (HILE, MIME, and LIHE), providing a great adaptation for many types of functions. Second, in the first stage of evolution, it uses a fixed small population of five individuals (two big males, a female, and two small males), with a high *mlipir* rate. This characteristic gives the guarantee and scalability to reach the global optima of the simple bowl-like unimodal and multimodal functions. Third, in the second stage of evolution, it utilizes a self-adaptive big population of 20 to 200 individuals and a half *mlipir* rate to guarantee global optima of the complex fixed low-dimensional multimodal functions. However, KMA has one disadvantage. Since KMA utilizes two fixed parameters of big male portion and *mlipir* rate throughout the two stages of evolution, it cannot guarantee global optima of the functions containing wide flat areas.

Experimental results show that KMA significantly outperforms the competitors: GA, SHADE, LSHADE-cnEpSin, EO, MPA, and SMA in terms of average solution, mean function evaluations, guaranty rate, and scalability. An evaluation of 23 benchmark functions shows that all the proposed schemes give KMA an exploitation–exploration balance that guarantees global optima for most functions rapidly. It can guarantee global optima for 17 out of 23 (73.91%) functions, which is higher than MPA (56.52%), SMA (43.48%), SHADE (39.13%), and the other competitors. A

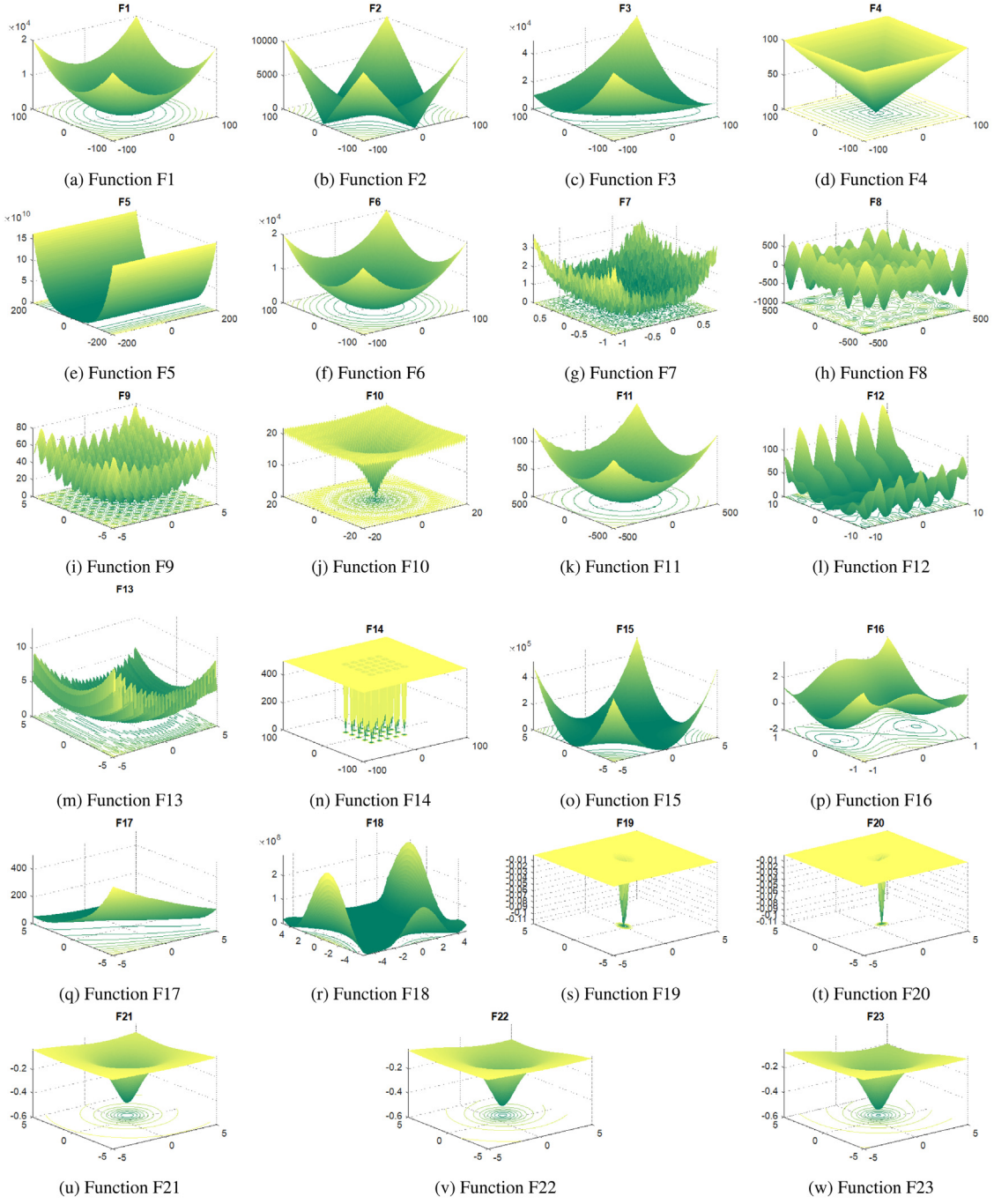


Fig. 8. Twenty three benchmark functions F1 to F23.

scalability evaluation informs that it can be scaled up to thousand dimensions. It guarantees global optima for 7 out of 13 (53.85%) functions efficiently in less than 5000 evaluations, which is significantly higher than SMA (23.08%), MPA (15.38%), EO (15.38%), and the other algorithms (0%). It is slightly unstable only for F5 (containing wide flat areas), but it is still better than GA, SHADE, and LSHADE-cnEpSin and competitive to EO and MPA. Hence, KMA is much better than all the competitors, making it a potential metaheuristic swarm-based optimizer.

In the future, an advanced self-adaptation scheme can be developed to dynamically update the two parameters: big male portion and *mlipir* rate, giving KMA an adaptive higher exploration to tackle the benchmarks with wide flat areas. Additionally, this research has one limitation. KMA is only evaluated using a small set of 13 simple high-dimensional and ten fixed low-dimensional benchmarks and is focused on a few evaluation metrics. Therefore, its performance should be investigated comprehensively using the more challenging benchmark functions and real-world applications.

Table 6
Twenty three benchmark functions: thirteen high-dimensional and ten fixed low-dimensional.

Func	Name	Type	Dimension	Range	f_{min}	x_i^*
F1	Sphere	HDU	{50, 100, 500, 1000}	[−100, 100]	0	(0, 0, …)
F2	Schwefel 2.22	HDU	{50, 100, 500, 1000}	[−100, 100]	0	(0, 0, …)
F3	Schwefel 1.2	HDU	{50, 100, 500, 1000}	[−100, 100]	0	(0, 0, …)
F4	Schwefel 2.21	HDU	{50, 100, 500, 1000}	[−100, 100]	0	(0, 0, …)
F5	Rosenbrock	HDU	{50, 100, 500, 1000}	[−30, 30]	0	(1, 1, …)
F6	Step	HDU	{50, 100, 500, 1000}	[−100, 100]	0	(0, 0, …)
F7	Quartic	HDU	{50, 100, 500, 1000}	[−1.28, 1.28]	0	(0, 0, …)
F8	Schwefel	HDM	{50, 100, 500, 1000}	[−500, 500]	−418.9829× Dim	(420.9687, 420.9687, …)
F9	Rastrigin	HDM	{50, 100, 500, 1000}	[−5.12, 5.12]	0	(0, 0, …)
F10	Ackley	HDM	{50, 100, 500, 1000}	[−32, 32]	0	(0, 0, …)
F11	Griewank	HDM	{50, 100, 500, 1000}	[−600, 600]	0	(0, 0, …)
F12	Penalized	HDM	{50, 100, 500, 1000}	[−50, 50]	0	(−1, −1, …)
F13	Penalized2	HDM	{50, 100, 500, 1000}	[−50, 50]	0	(1, 1, …)
F14	Foxholes	FDM	2	[−65, 65]	0.998	(−31.9783, −31.9783)
F15	Kowalik	FDM	4	[−5, 5]	0.0003	(0.1928, 0.1908, 0.1231,0.1358)
F16	Six Hump Camel	FDM	2	[−5, 5]	−1.0316	(0.0898, −0.7126) , (−0.0898, 0.7126)
F17	Branin	FDM	2	[−5, 5]	0.398	($-\pi$, 12.275), (π, 2.275), (9.425, 2.275)
F18	GoldStein–Price	FDM	2	[−2, 2]	3	(0, −1)
F19	Hartman 3	FDM	3	[0, 1]	−3.86278	(0.1146, 0.5557, 0.8526)
F20	Hartman 6	FDM	6	[0, 1]	−3.32	(0.202, 0.150, 0.477, 0.275, 0.312, 0.657)
F21	Shekel 5	FDM	4	[0, 10]	−10.1532	(4, 4, 4, 4)
F22	Shekel 7	FDM	4	[0, 10]	−10.4029	(4, 4, 4, 4)
F23	Shekel 10	FDM	4	[0, 10]	−10.5364	(4, 4, 4, 4)

CRediT authorship contribution statement

Suyanto Suyanto: Principal investigator, Conceptualization, Methodology, Data curation, Supervision, Writing – original draft. **Alifya Aisyah Ariyanto:** Investigation, Writing – review & editing. **Alifya Fatimah Ariyanto:** Validation, 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.

Acknowledgments

As the first author, I would like to thank my parents, Saekan and Musripah, for inspiring the *mlipir* movement adopted in the Komodo Mlipir Algorithm and also my teenage daughter, Alima Tasnim Ariyanto, for creating the Komodo cartoon.

Appendix. Specification of 23 benchmark functions and their two-dimensional views

See Table 6 and Fig. 8.

References

[1] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1) (1992) 66–72, http://dx.doi.org/10.1038/scientificamerican0792-66, URL https://www.scientificamerican.com/article/genetic-algorithms/.
[2] Suyanto, An Informed Genetic Algorithm for University Course and Student Timetabling Problems, vol. 6114 LNAI, (PART 2) 2010, http://dx.doi.org/10.1007/978-3-642-13232-2_28.
[3] J. Cheng, Z. Pan, H. Liang, Z. Gao, J. Gao, Differential evolution algorithm with fitness and diversity ranking-based mutation operator, Swarm Evol. Comput. 61 (May 2020) (2021) 100816, http://dx.doi.org/10.1016/j.swevo.2020.100816, URL https://doi.org/10.1016/j.swevo.2020.100816.
[4] M.A. Bourouis, A. Zadjaoui, A. Djedid, Contribution of two artificial intelligence techniques in predicting the secondary compression index of fine-grained soils, Innov. Infrast. Solut. 5 (3) (2020) http://dx.doi.org/10.1007/s41062-020-00348-1.
[5] X. Yin, Z. Niu, Z. He, Z. Li, D. Lee, An integrated computational intelligence technique based operating parameters optimization scheme for quality improvement oriented process-manufacturing system, Comput. Ind. Eng. 140 (2020) http://dx.doi.org/10.1016/j.cie.2020.106284.
[6] V. Mp, B. Anand, Microprocessors and Microsystems Particle swarm optimization technique for multilevel inverters in solar harvesting micro grid system, Microprocess. Microsyst. 79 (August) (2020) 103288, http://dx.doi.org/10.1016/j.micpro.2020.103288.
[7] S. Palacios, S. Chiriboga, W. Montalvo, PID-2DOF-ACO speed controller for DC motor on ARM platform [Controlador de velocidad PID-2DOF-ACO para motor DC sobre plataforma ARM], RISTI - Revista Iberica de Sistemas E Tecnologias de Informacao 2020 (E30) (2020) 217–228.
[8] N. Hansen, S.D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), Genet. Evol. Comput. Ser. 11 (1) (2003) 1–18, http://dx.doi.org/10.1162/106365603321828970.
[9] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for Differential Evolution, in: 2013 IEEE Congress on Evolutionary Computation, (ISSN: 1941-0026) 2013, pp. 71–78, http://dx.doi.org/10.1109/CEC.2013.6557555.
[10] R. Tanabe, A.S. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in: IEEE Congress on Evolutionary Computation (CEC), 2014.
[11] N.H. Awad, M.Z. Ali, P.N. Suganthan, Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving CEC2017 benchmark problems, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 372–379, http://dx.doi.org/10.1109/CEC.2017.7969336.
[12] X.-S. Yang, Firefly algorithm, stochastic test functions and design optimization, Int. J. Bio-Inspired Comput. 2 (2) (2010) 78–84, http://dx.doi.org/10.1504/IJBIC.2010.032124, URL https://www.inderscienceonline.com/doi/pdf/10.1504/IJBIC.2010.032124.
[13] V. Kumar, D. Kumar, A systematic review on firefly algorithm: Past, present, and future, Arch. Comput. Methods Eng. (2020) http://dx.doi.org/10.1007/s11831-020-09498-y, URL https://www.x-mol.com/paper/1311351373626052608.
[14] X.-S. Yang, S. Deb, Cuckoo search via levy flights, 2010, arXiv:1003.1594.
[15] H. Ma, S. Li, E. Zhang, Z. Lv, J. Hu, X. Wei, Cooperative autonomous driving oriented MEC-Aided 5G-V2X: Prototype system design, field tests and AI-based optimization tools, IEEE Access 8 (2020) 54288–54302, http://dx.doi.org/10.1109/ACCESS.2020.2981463.
[16] X.-s. Yang, A new metaheuristic bat-inspired algorithm, in: Nature Inspired Coop- Erative Strategies for Optimization (NISCO), 2010, pp. 65–74, arXiv:arXiv:1004.4170v1, URL https://link.springer.com/chapter/10.1007/978-3-642-12538-6_6.
[17] A.A. Fadhil, R.G.H. Alsarraj, A.M. Altaie, Software cost estimation based on dolphin algorithm, IEEE Access 8 (2020) 75279–75287, http://dx.doi.org/10.1109/ACCESS.2020.2988867.
[18] K.M. Passino, Biomimicry of bacterial foraging for distributed optimization and control, IEEE Control Syst. 22 (3) (2002) 52–67, http://dx.doi.org/10.1109/MCS.2002.1004010, URL https://ieeexplore.ieee.org/abstract/document/1004010.
[19] X. Gan, B.X. B, Improved Bacterial Foraging Optimization Algorithm with Comprehensive Swarm, Springer International Publishing, 2020, pp. 325–334, http://dx.doi.org/10.1007/978-3-030-53956-6, URL http://dx.doi.org/10.1007/978-3-030-53956-6_29.

[20] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: A gravitational search algorithm, Inform. Sci. 179 (13) (2009) 2232–2248, http://dx.doi.org/10.1016/j.ins.2009.03.004, URL http://www.sciencedirect.com/science/article/pii/S0020025509001200, Special Section on High Order Fuzzy Sets.
[21] 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, http://dx.doi.org/10.1016/j.advengsoft.2017.07.002, URL http://www.sciencedirect.com/science/article/pii/S0965997816307736.
[22] R. Rao, V. Savsani, D. Vakharia, Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems, Comput. Aided Des. 43 (3) (2011) 303–315, http://dx.doi.org/10.1016/j.cad.2010.12.015, URL http://www.sciencedirect.com/science/article/pii/S0010448510002484.
[23] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61, http://dx.doi.org/10.1016/j.advengsoft.2013.12.007.
[24] S. Gupta, K. Deep, Enhanced leadership-inspired grey wolf optimizer for global optimization problems, Eng. Comput. 36 (4) (2020) 1777–1800, http://dx.doi.org/10.1007/s00366-019-00795-0.
[25] S. Mirjalili, Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, Neural Comput. Appl. 27 (4) (2016) 1053–1073, http://dx.doi.org/10.1007/s00521-015-1920-1, URL https://doi.org/10.1007/s00521-015-1920-1.
[26] A.I. Hammouri, M. Mafarja, M.A. Al-Betar, M.A. Awadallah, I. Abu-Doush, An improved Dragonfly Algorithm for feature selection, Knowl.-Based Syst. 203 (2020) 106131, http://dx.doi.org/10.1016/j.knosys.2020.106131, URL http://www.sciencedirect.com/science/article/pii/S0950705120303889.
[27] S. Mirjalili, Advances in engineering software the ant lion optimizer, Adv. Eng. Softw. 83 (2015) 80–98, http://dx.doi.org/10.1016/j.advengsoft.2015.01.010, URL http://dx.doi.org/10.1016/j.advengsoft.2015.01.010.
[28] A.S. Assiri, A.G. Hussien, M. Amin, Ant lion optimization: Variants, hybrids, and applications, IEEE Access 8 (2020) 77746–77764, http://dx.doi.org/10.1109/ACCESS.2020.2990338, URL https://ieeexplore.ieee.org/abstract/document/9078091.
[29] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, Knowl.-Based Syst. 89 (2015) 228–249, http://dx.doi.org/10.1016/j.knosys.2015.07.006, URL https://www.sciencedirect.com/science/article/abs/pii/S0950705115002580.
[30] R. Zhang, Z. Qiu, Optimizing hyper-parameters of neural networks with swarm intelligence: A novel framework for credit scoring, PLoS ONE 15 (6) (2020) http://dx.doi.org/10.1371/journal.pone.0234254.
[31] S.A. Uymaz, G. Tezel, E. Yel, Artificial algae algorithm (AAA) for nonlinear global optimization, Appl. Soft Comput. 31 (2015) 153–171, http://dx.doi.org/10.1016/j.asoc.2015.03.003, URL https://www.sciencedirect.com/science/article/pii/S1568494615001465.
[32] S. Korkmaz, M.S. Kiran, An artificial algae algorithm with stigmergic behavior for binary optimization, Appl. Soft Comput. 64 (2018) 627–640, http://dx.doi.org/10.1016/j.asoc.2018.01.001, URL https://www.sciencedirect.com/science/article/pii/S1568494618300061.
[33] W. Zhao, L. Wang, Z. Zhang, A novel atom search optimization for dispersion coefficient estimation in groundwater, Future Gener. Comput. Syst. (2018) http://dx.doi.org/10.1016/j.future.2018.05.037.
[34] R.V. Rao, Rao algorithms: Three metaphor-less simple algorithms for solving optimization problems, Int. J. Ind. Eng. Comput. 11 (1) (2020) 107–130, http://dx.doi.org/10.5267/j.ijiec.2019.6.002.
[35] S. Suyanto, A.T. Wibowo, S.A. Faraby, S. Saadah, R. Rismala, Evolutionary Rao algorithm, J. Comput. Sci. 53 (March) (2021) 101368, http://dx.doi.org/10.1016/j.jocs.2021.101368.
[36] A. Faramarzi, M. Heidarinejad, B. Stephens, S. Mirjalili, Equilibrium optimizer: A novel optimization algorithm, Knowl.-Based Syst. 191 (2020) http://dx.doi.org/10.1016/j.knosys.2019.105190, URL https://www.sciencedirect.com/science/article/abs/pii/S0950705119305295?via%3Dihub.
[37] A. Faramarzi, M. Heidarinejad, S. Mirjalili, A.H. Gandomi, Marine Predators Algorithm: A nature-inspired metaheuristic, Expert Syst. Appl. 152 (2020) http://dx.doi.org/10.1016/j.eswa.2020.113377.
[38] M.A. Elaziz, A.A. Ewees, D. Yousri, H.S.N. Alwerfali, Q.A. Awad, S. Lu, M.A.A. Al-Qaness, An improved marine predators algorithm with fuzzy entropy for multi-level thresholding: Real world example of COVID-19 CT image segmentation, IEEE Access 8 (2020) 125306–125330, http://dx.doi.org/10.1109/ACCESS.2020.3007928.
[39] S. Li, H. Chen, M. Wang, A.A. Heidari, S. Mirjalili, Slime mould algorithm: A new method for stochastic optimization, Future Gener. Comput. Syst. 111 (2020) 300–323, http://dx.doi.org/10.1016/j.future.2020.03.055, URL https://www.sciencedirect.com/science/article/pii/S0167739X19320941.
[40] R.-E. Precup, R.-C. David, R.-C. Roman, E.M. Petriu, A.-I. Szedlak-Stinean, Slime mould algorithm-based tuning of cost-effective fuzzy controllers for servo systems, Int. J. Comput. Intell. Syst. 14 (1) (2021) 1042–1052, http://dx.doi.org/10.2991/ijcis.d.210309.001.
[41] J. Brest, M.S. Maučec, The 100-digit challenge : Algorithm jDE100, in: 2019 IEEE Congress on Evolutionary Computation (CEC), 2019, http://dx.doi.org/10.1109/CEC.2019.8789904.
[42] K.M. Sallam, S.M. Elsayed, R.K. Chakrabortty, M.J. Ryan, Improved multi-operator differential evolution algorithm for solving unconstrained problems, in: 2020 IEEE Congress on Evolutionary Computation (CEC), 2020, pp. 1–8, http://dx.doi.org/10.1109/CEC48606.2020.9185577.
[43] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82, http://dx.doi.org/10.1109/4235.585893, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-0031118203&doi=10.1109%2f4235.585893&partnerID=40&md5=fac8c56be911367d556066800e863066, cited By 5993.
[44] T.E. of Encyclopaedia, Komodo dragon, in: Britannica, 2020, URL https://www.britannica.com/animal/Komodo-dragon.
[45] J. Sartore, Komodo dragon, in: National Geographic, 2021, URL https://www.nationalgeographic.com/animals/reptiles/k/komodo-dragon/.
[46] A.L. Lind, Y.Y. Lai, Y. Mostovoy, A.K. Holloway, A. Iannucci, A.C. Mak, M. Fondi, V. Orlandini, W.L. Eckalbar, M. Milan, et al., Genome of the Komodo dragon reveals adaptations in the cardiovascular and chemosensory systems of monitor lizards, Nat. Ecol. Evol. 3 (8) (2019) 1241–1252.
[47] C. Ciofi, The Komodo dragon, Sci. Am. 280 (3) (1999) 84–91.
[48] P.C. Watts, K.R. Buley, S. Sanderson, W. Boardman, C. Ciofi, R. Gibson, Parthenogenesis in Komodo dragons, Nature 444 (7122) (2006) 1021–1022.
[49] K. Opara, J. Arabas, Comparison of mutation strategies in Differential evolution – a probabilistic perspective, Swarm Evol. Comput. 39 (2018) 53–69, http://dx.doi.org/10.1016/j.swevo.2017.12.007, URL https://www.sciencedirect.com/science/article/pii/S2210650217303310.
[50] C. Stephens, H. Waelbroeck, Schemata evolution and building blocks, Genet. Evol. Comput. Ser. 7 (2) (1999) 109–124, http://dx.doi.org/10.1162/evco.1999.7.2.109.