Applied Soft Computing 36 (2015) 315–333

Contents lists available at ScienceDirect

Applied Soft Computing

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

Lightning search algorithm

Hussain Shareef*, Ahmad Asrul Ibrahim, Ammar Hussein Mutlag

Department of Electrical Engineering, United Arab Emirates University, P.O. Box 15551, 1 Al-Ain, United Arab Emirates

ARTICLE INFO

Article history:
Received 11 September 2014
Received in revised form 23 July 2015
Accepted 24 July 2015
Available online 4 August 2015

Keywords:
Benchmark functions
Constrained optimization
Lightning search algorithm
Nature-inspired algorithms

ABSTRACT

This paper introduces a novel metaheuristic optimization method called the lightning search algorithm (LSA) to solve constraint optimization problems. It is based on the natural phenomenon of lightning and the mechanism of step leader propagation using the concept of fast particles known as projectiles. Three projectile types are developed to represent the transition projectiles that create the first step leader population, the space projectiles that attempt to become the leader, and the lead projectile that represent the projectile fired from best positioned step leader. In contrast to that of the counterparts of the LSA, the major exploration feature of the proposed algorithm is modeled using the exponential random behavior of space projectile and the concurrent formation of two leader tips at fork points using opposition theory. To evaluate the reliability and efficiency of the proposed algorithm, the LSA is tested using a well-utilized set of 24 benchmark functions with various characteristics necessary to evaluate a new algorithm. An extensive comparative study with four other well-known methods is conducted to validate and compare the performance of the LSA. The result demonstrates that the LSA generally provides better results compared with the other tested methods with a high convergence rate.

1. Introduction

Optimization is a process of finding the best solution. Solutions are labeled good or bad after determining the objective function that states the relations between system parameters and constraints. The objective function is often formulated based on application, and it can be in the form of fabrication cost, process efficiency, and so on.

Numerous techniques have been used to deal with optimization problems. Most classical point-by-point methods (e.g., direct methods and gradient-based methods) use a deterministic procedure to reach the optimum solution [1]. However, finding the optimum solution for such problems using classical techniques becomes complicated as the size of the search space increases with the dimension of the optimization problem [2]. Recently, computational intelligence optimization algorithms have been extensively used to solve complex optimization problems in various domains, including science, commerce, and engineering, because of their ease of use, broad applicability, and global perspective.

Computational intelligence optimization algorithms are nature-inspired computational methodologies that address complex real-world problems. These algorithms can be further divided into swarm intelligence methods and evolutionary algorithms (EAs). Swarm intelligence optimization algorithms generally use reduced mathematical models of the complex social behavior of insect or animal groups. The most popular swarm intelligence methods are particle swarm optimization (PSO) [3], artificial bee colony (ABC) [4], and ant colony optimization (ACO) [5]. The PSO mimics the movements of bird flocking or fish schooling [6]. Inspired by the food-searching mechanism of honey bees, the ABC method uses the foraging behavior of these insects [7]. Meanwhile, ACO was developed based on the behavior of ants when seeking the optimal path between their colony and food source [5]. However, these swarm intelligence methods are limited by factors such as trapping in local minima and premature convergence [6–8]. To overcome these problems, variants of these algorithms have been developed with superior performance [6,8–10]. Other swarm intelligence methods, such as the gravitational search algorithm (GSA) [11], the harmony search algorithm (HSA) [12], biogeography-based optimization [13], and the grenade explosion method [14], have also been developed.

EAs derive their working principles from natural genetic evolution. At each generation, the best individuals of the current population survive and produce offspring resembling them; hence, the population gradually comprises enhanced individuals. Operations such as recombination, crossover, mutation, selection, and adaptation are involved in this process [15]. The renowned

* Corresponding author. Tel.: +60 12 7714885; fax: +60 3 89216146.
E-mail addresses: hussain_ln@yahoo.com (H. Shareef), asrul@eng.ukm.my (A.A. Ibrahim), ammar_alqiesy@yahoo.com (A.H. Mutlag).

http://dx.doi.org/10.1016/j.asoc.2015.07.028
1568-4946/© 2015 Elsevier B.V. All rights reserved.

paradigms of EAs are the genetic algorithm (GA) [15], evolutionary programming [16], differential evolution [17], evolutionary strategy [16], and genetic programming [18]. These algorithms are based on the principles of Darwinian theory and evolution theory of living beings. However, each algorithm follows specialized recombination, crossover, mutation, selection, and adaptation strategies. Similar to other metaheuristic algorithms, the aforementioned methods also have some drawbacks, such as slow convergence rate, difficulty in solving multimodal functions, and stagnation in local minima [19–21]. Advanced versions of EAs have been developed in recent years to improve the efficiency and performance of the aforementioned EAs; these advanced algorithms include stud genetic algorithm [21], fast evolutionary programming [20], adaptive differential evolution algorithm [22], and covariance matrix adaptation evolution strategy [23]. Not all algorithms and their variants provide superior solutions to some specific problems. Therefore, new heuristic optimization algorithms must be continuously searched to advance the field of computational intelligence optimization.

This study aims to introduce a novel metaheuristic optimization method called the lightning search algorithm (LSA) to solve constraint optimization problems. The LSA is based on the natural phenomenon of lightning. The proposed optimization algorithm is generalized from the mechanism of step leader propagation. It considers the involvement of fast particles known as projectiles in the formation of the binary tree structure of a step leader. Three projectile types are developed to represent the transition projectiles that create the first step leader population *N*, the space projectiles that attempt to become the leader, and the lead projectile that represent the best positioned projectile originated among *N* number of step leaders. The probabilistic nature and tortuous characteristics of lightning discharges, which depend on the type of projectile, are modeled using various random distribution functions. The LSA is explained in depth in Section 4.

2. Highlights of recent nature-inspired optimization algorithms

Many real-world optimization problems involve non-linearities and complex interactions among problem variables. Therefore, the capacity of nature-based algorithms to solve different optimization problems effectively and efficiently must be increased. Increasing the problem-solving capacity of these algorithms is generally achieved by modifying existing algorithms, hybridizing algorithms, and developing new algorithms. Several computational intelligence optimization algorithms have been proposed to overcome the limitations of their predecessors. The following descriptions highlight the recent methods published in the scientific literature.

2.1. Bat algorithm

The bat-inspired algorithm is a metaheuristic optimization algorithm developed by Xin-She Yang in 2010 [24]. The bat algorithm is based on the echolocation behavior of microbats with varying pulse rates of emission and loudness. Each virtual bat flies randomly with a velocity v_i at position (solution) x_i with a varying frequency or wavelength and loudness A_i. As the bat searches and finds its prey, the frequency, loudness, and pulse emission rate r are modified using a frequency-tuning technique to control the dynamic behavior of a swarm of bats. Search is intensified by a local random walk. The update of the velocities and positions of bats is similar to the standard updating procedure of PSO [24]. However, the bat algorithm features an intensive local search control, which is implemented by adjusting the loudness and pulse rate. The selection of the best continues until certain stop criteria are met. The main drawback of the standard bat algorithm is the abrupt switching to the exploitation stage by quickly varying A and r. This quick variation may lead to stagnation after the initial stages [25]. Many researchers have recognized this limitation of the bat algorithm and provided strategies to enhance the performance of this algorithm [26–28]. These strategies include using fuzzy logic [26], chaotic sequence [27], deferential operator, and Levy flight concepts [27,28].

2.2. Firefly algorithm (FFA)

The FA is a novel nature-inspired metaheuristic algorithm that is used to solve continuous multi-objective optimization problems based on the social behavior of fireflies [29]. The FA is an efficient technique for searching the Pareto optimal set, and its success rate and efficiency are better than those of PSO and the GA for both continuous and discrete problems [30]. The standard FA involves two important issues, namely, variation of light intensity *I* and formulation of attractiveness β. The attractiveness between fireflies is formulated as a function of the square of distance r between each other and the light absorption coefficient γ. As the fireflies search for the best solution, their movements are updated based on their current position, attractiveness, and a randomization term. When γ tends to be zero, the FA corresponds to the standard PSO [30]. However, the FA is superior to other algorithms because it is capable of automatic subdivision and dealing with multimodality [31].

2.3. Backtracking search algorithm (BSA)

Modeled based on the EA, the BSA is aimed at solving problems that are frequently encountered in EAs, such as excessive sensitivity to control parameters and premature convergence. Similar to the conventional EA, the BSA involves five processes, namely, initialization, initial selection, mutation, crossover, and second selection. In the initial selection, the BSA calculates the historical population as an indicator of the search direction. It can also redefine historical population at the beginning of each iteration in such a way that a population belonging to a randomly selected previous generation acts as a memory until it is changed. The mutation and crossover strategies of the BSA are different from those of the EA and its advanced versions. In generating trail population, only one parameter is used to control the amplitude of the search direction matrix in the mutation phase. However, crossover is complex. Two strategies are used to generate the final trial population. The first strategy uses mix rate to control the number of elements of individuals that will mutate in a trial, and the second strategy allows only one randomly chosen individual to mutate in each trial. In the second selection stage of the BSA, the population is updated using greedy selection, in which only individuals with high fitness values in the trail population are used. Despite its simple structure, the BSA may be memory and time consuming in computation because of the use of the dual population algorithm [32].

2.4. Krill herd algorithm (KHA)

The KHA is a newly developed swarm intelligence approach for optimization tasks based on the herding behavior of krill individuals [33]. It is a population-based method that consists of an idealized swarm of krills, each of which travels through a multidimensional search space to search for food. In the KHA, the positions of krill individuals are considered as different design variables, and the distance of the food from the krill individuals is equivalent to the fitness value of the objective function. In addition, krill individuals alter their position and travel to better positions. The position of each individual is affected by three principal processes: (i) movement affected by other krill individuals, (ii) foraging activity, and (iii) random diffusion [33,34]. The KHA is advantageous because it only requires a few control parameters to regulate. However, the KHA is principally based on random walks during search. It cannot consistently maintain the convergence area to the global optimum value. To improve this premature convergence problem, a krill migration operator was introduced in [35] during krill updating.

2.5. Teacher learning optimization (TLO)

The TLO algorithm was first introduced by Rao et al. [36]. Similar to other nature-inspired algorithms, the TLO is also a population-based method that uses a population of solutions to proceed to the global solution. The TLO is based on the interaction between students and a teacher in a class [36,37]. The population consists of the learners and the teacher. This algorithm aims to enhance the overall knowledge of the class. In this algorithm, the optimization process is based on two phases, namely, teacher and learner. The teacher phase simulates the learning process from the teacher, and the learner phase mimics learning from other students in the class. In the teacher phase, the students increase their understanding by learning from the teacher. Therefore, in this phase, a good teacher is one who can transmit his or her knowledge to the class [36]. The learner phase is based on the interaction among students. Students can learn from one another through interaction and sharing. These two phases should be applied to students. The principal drawback of the algorithm is that it uses a single teacher concept and a constant teaching factor. The use of a single teacher may cause premature convergence while a constant teaching factor may increase the computation time [37,38]. Many techniques such as the use of multiple teachers and adaptive teaching were suggested in [38,39].

2.6. Social spider optimization (SSO)

The SSO algorithm, which is based on the simulation of the cooperative behavior of social spiders, was first introduced by Erik Cuevas et al. in 2013 [40]. In the SSO algorithm, individuals imitate a swarm of spiders that interact based on the biological laws of the cooperative colony. The algorithm considers two search agents (spiders): males and females. Depending on sex, each individual is conducted by a set of different evolutionary operators that mimic different cooperative behaviors that are typically found in the colony. This procedure incorporates computational mechanisms to avoid critical faults typically present in the popular PSO and ABC algorithms; such faults include premature convergence and the incorrect exploration–exploitation balance [41].

2.7. Differential search algorithm (DSA)

The concept behind the development of the DSA is the seasonal migration of different species in search of fruitful living. All organisms form a superorganism and start moving together to discover efficient habitats. During their journey, the individuals of the superorganism check whether their randomly chosen locations meet their transitory criteria. If any location is appropriate for their temporary layover during the journey, the individuals of the superorganism that revealed the stopover immediately settle in that location and carry on their journey from that location. During a stopover, the superorganism tries to explore the sites in the area left between the organisms using a random process similar to the Brownian-like random walk. Then, donors are made by reshuffling all individuals of the superorganism. These donors are appropriate in discovering the stopover site. Randomly selected members of the superorganism move toward the target of the donors to successfully discover the stopover site. This change in site allows the superorganism to continue its migration toward the global minimum. The DSA is suitable for multimodal optimization problems because it has no inclination to correctly select the best possible solution for a given problem. However, similar to PSO, the DSA has two control variables for fine tuning based on the problem under consideration. Further details on the DSA can be found in [42].

Fig. 1. Step leaders descending from a storm cloud.

2.8. Cuckoo search algorithm (CSA)

The CSA is a metaheuristic optimization algorithm first developed by Yang and Deb in 2009 [43] and further developed by Rajabioun [44]. It was inspired by the lifestyle of cuckoo birds. The egg laying and breeding properties of cuckoos are the main concepts used in this algorithm. Each individual cuckoo in the algorithm has a habitat around which it starts to lay eggs. Surviving eggs grow and become mature cuckoos. Then, cuckoos move toward the best habitat to mate. The change during the travel toward the target habitat motivates the population to search for other areas. After some immigration events, all cuckoo populations gather in the same habitat, that is, the area's best position [44]. However, the performance of the CSA is affected by algorithm-dependent control parameters, such as egg laying radius [45].

3. Mechanism of lightning

Lightning is a fascinating and impressive natural phenomenon. The probabilistic nature and tortuous characteristics of lightning discharges originate from a thunderstorm. Among the common displays of lightning, downward negative cloud-to-ground lightning flashes (Fig. 1) are the most studied event in lightning research [46].

During a thunderstorm, charge separation occurs within the cloud, usually with a positive charge above and a negative charge below. This process creates a strong electric field. Free electrons generated by cosmic radiation or natural radioactivity are generally attached to oxygen molecules to form negative ions [47]. However, under high electric fields, a fraction of the electron velocity distribution possesses enough energy to ionize air, thereby generating additional electrons along with the original ones.

When one of these free electrons is accelerated by the electric field in the region where ionization probability is higher than attachment probability, an electron avalanche occurs. This phenomenon eventually causes a negative corona streamer. Nonetheless, the streamers are weak and become electrically isolated from their point of origin when the ionization probability is lower than the attachment probability.

The so-called streamer-to-leader transition occurs at high electron densities and temperatures when narrow channels are thermalized. The negative or the step leader's movement from cloud to ground is not continuous but progresses through regular and discrete steps. For the leader to progress, a leader section in the vicinity in front of the old leader channel, called a space leader,

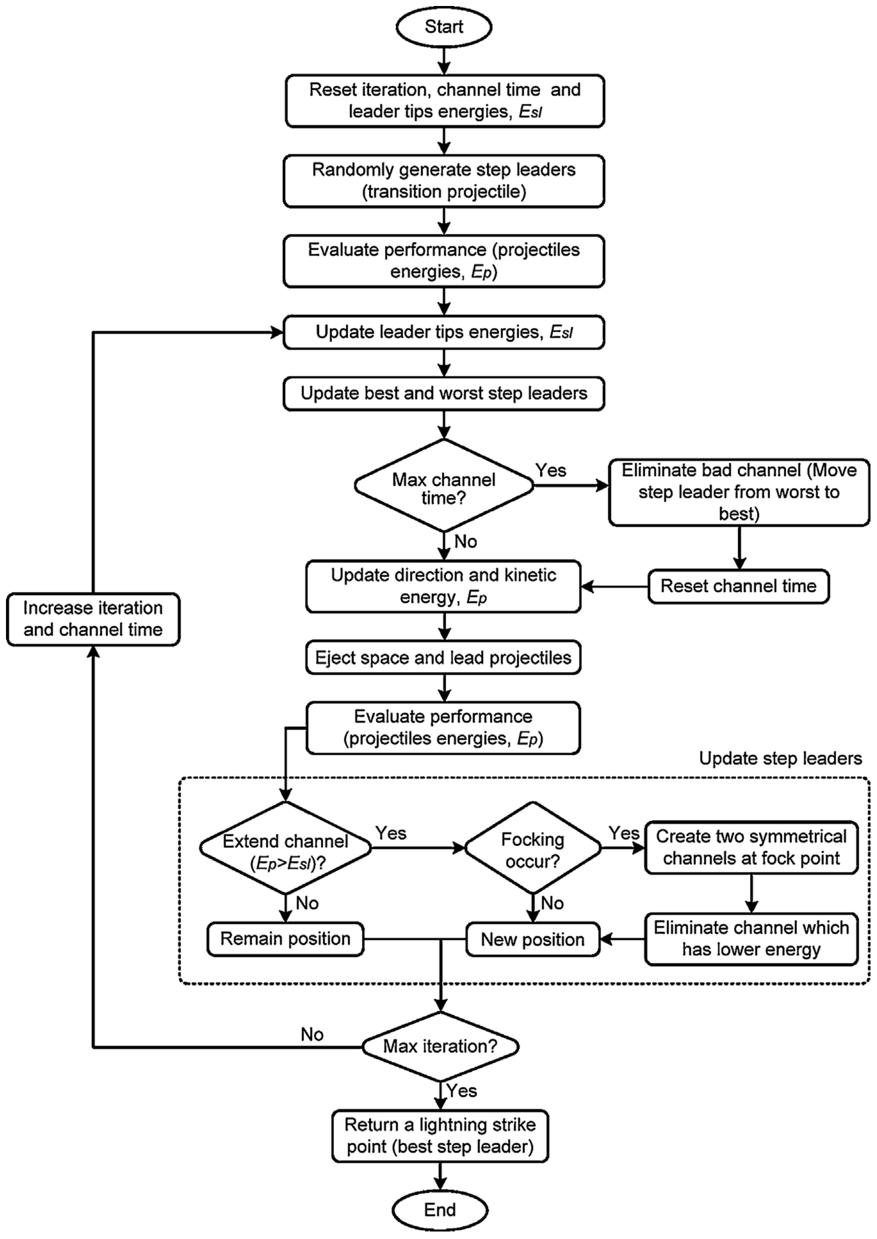
Start
Reset iteration, channel time and leader tips energies, Esl
Randomly generate step leaders (transition projectile)
Evaluate performance (projectiles energies, Ep)
Update leader tips energies, Esl
Update best and worst step leaders
Max channel time?
Yes
Eliminate bad channel (Move step leader from worst to best)
Reset channel time
No
Update direction and kinetic energy, Ep
Increase iteration and channel time
Eject space and lead projectiles
Evaluate performance (projectiles energies, Ep)
Update step leaders
Extend channel (Ep>Esl)?
Yes
Focking occur?
Yes
Create two symmetrical channels at fock point
No
Remain position
No
New position
Eliminate channel which has lower energy
No
Max iteration?
Yes
Return a lightning strike point (best step leader)
End

Fig. 2. Flowchart of the LSA.

develops from the preceding corona streamers. The space leader propagates backward until it connects to the old leader channel forming a long channel with a tip having the same electric potential [48]. A current wave is generated during this process.

When this current wave reaches the tip of the new leader, a burst of corona streamers again propagates out in front of the new tip. Afterward, a new space leader originates in front of the current leader tip. The process is then repeated. The development is a stochastic growth process in which the probability of a step to grow in a certain direction is determined by the state of the medium and the projection of the local electric field in this direction [46,49].

As the step leaders approach the earth, high-density negative charges at the leader tip stimulate a concentrated positive charge on Earth. When the electric field intensity resulting from the positive charges near the ground becomes large enough, upward positive streamers from the ground are initiated, and the step leader bridges the gap as they move their way down. This phenomenon determines the lightning strike point and the main lightning current path between the cloud and the ground. The step leader is not the lightning strike; it only maps the optimal course that the strike will follow.

4. Lightning search algorithm (LSA)

The proposed optimization algorithm is generalized from the hypothesis presented in [50] as a mechanism of step leader propagation. It considers the involvement of fast particles known as projectiles in the formation of the binary tree structure of the step leader and in the concurrent formation of two leader tips at fork points instead of a conventional step leader mechanism that uses the concept of streamers described above.

4.1. Projectile and step leader propagation

Hydrogen, nitrogen, and oxygen atoms can be found near the region of thunderclouds. During the intensive freezing of water molecules within a thundercloud, a portion of the water molecules unable to fit in the ice structure are squeezed out of the forming

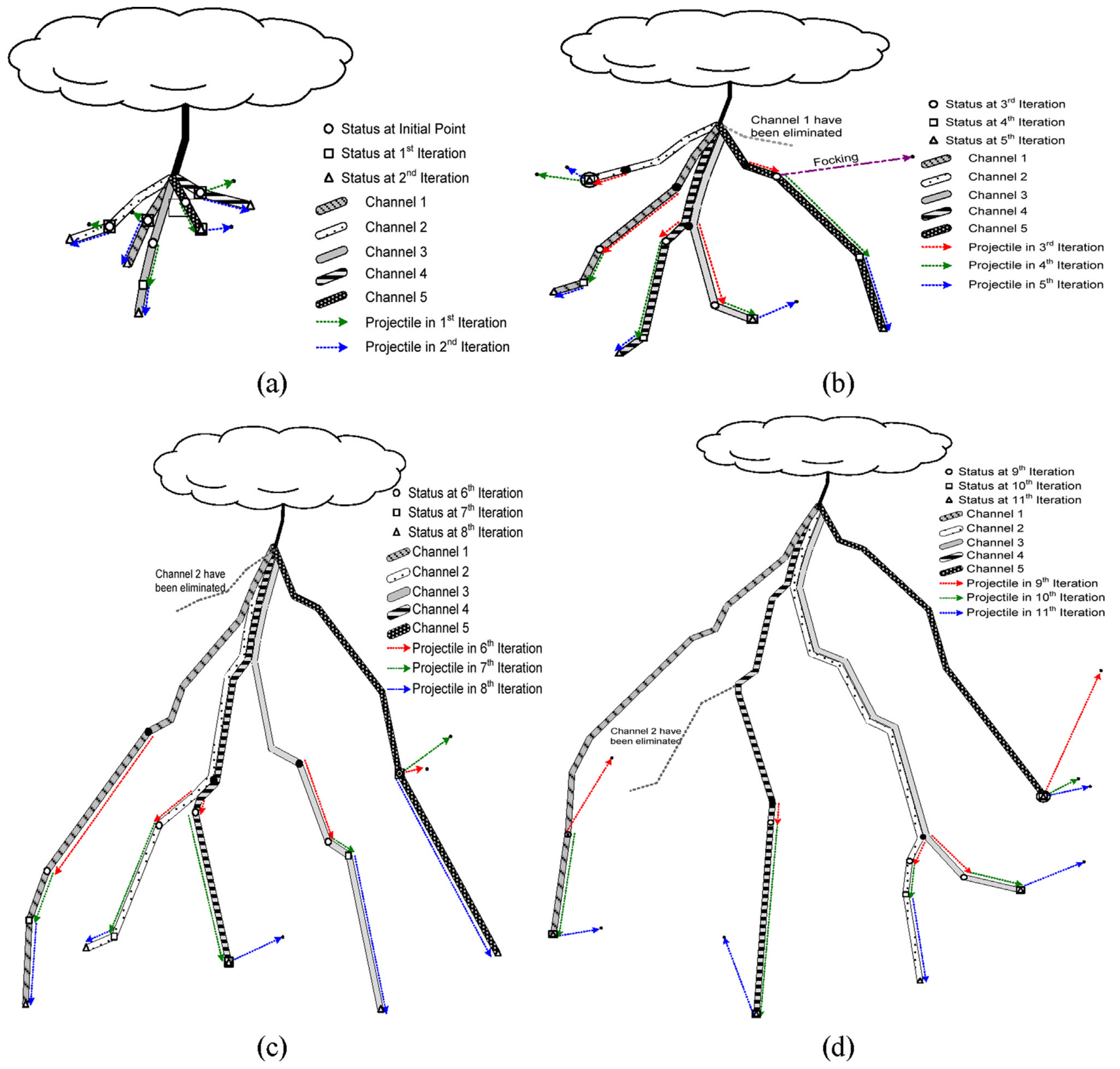
Status at Initial Point
Status at 1st Iteration
Status at 2nd Iteration
Channel 1
Channel 2
Channel 3
Channel 4
Channel 5
Projectile in 1st Iteration
Projectile in 2nd Iteration
(a)
Channel 1 have been eliminated
Focking
Status at 3rd Iteration
Status at 4th Iteration
Status at 5th Iteration
Projectile in 3rd Iteration
Projectile in 4th Iteration
Projectile in 5th Iteration
(b)
Channel 2 have been eliminated
Status at 6th Iteration
Status at 7th Iteration
Status at 8th Iteration
Projectile in 6th Iteration
Projectile in 7th Iteration
Projectile in 8th Iteration
(c)
Status at 9th Iteration
Status at 10th Iteration
Status at 11th Iteration
Projectile in 9th Iteration
Projectile in 10th Iteration
Projectile in 11th Iteration
(d)

Fig. 3. Illustration of LSA functionality in finding the global minimum of the Sphere function. Position of the step leaders until the (a) 2nd, (b) 5th, (c) 8th, and (d) 11th iteration.

ice at intense speeds. The oxygen and hydrogen atoms are separated during the process and ejected in a random direction as projectiles. After being ejected from the thunder cell, the projectile travels through the atmosphere and provides an initial ionization path or channel through collision and transition to the step leader. In the proposed algorithm, each projectile from the thunder cell is assumed to create a step leader and a channel. In other words, the projectile represents the initial population size. The concept of projectile is highly similar to the term "particle" or "agent" used in PSO and the GSA, respectively. The projectiles suggest random solutions for corresponding problems to be solved by the LSA. In the present study, the solution refers to the tip of the current step leader's energy Ec.

4.2. Properties of the projectile

A projectile that travels through the atmosphere under normal conditions loses its kinetic energy during elastic collisions with molecules and atoms in the air. The velocity of a projectile is given as

$$v_p = \left[1 - \left(1/\sqrt{1-(v_0/c)^2} - sF_i/mc^2\right)^{-2}\right]^{-1/2} \quad (1)$$

where v_p and v_0 are the current velocity and initial velocity, respectively, of the projectile; c is the speed of light; F_i is the constant ionization rate; m is the mass of the projectile; and s is the length of the path traveled.

Eq. (1) clearly shows that velocity is a function of leader tip position and projectile mass. When the mass is small or when the path traveled is long, the projectile has little potential to ionize or explore a large space. It can only ionize or exploit the nearby space. Therefore, the exploration and exploitation abilities of the algorithm can be controlled by using the relative energies of the step leaders.

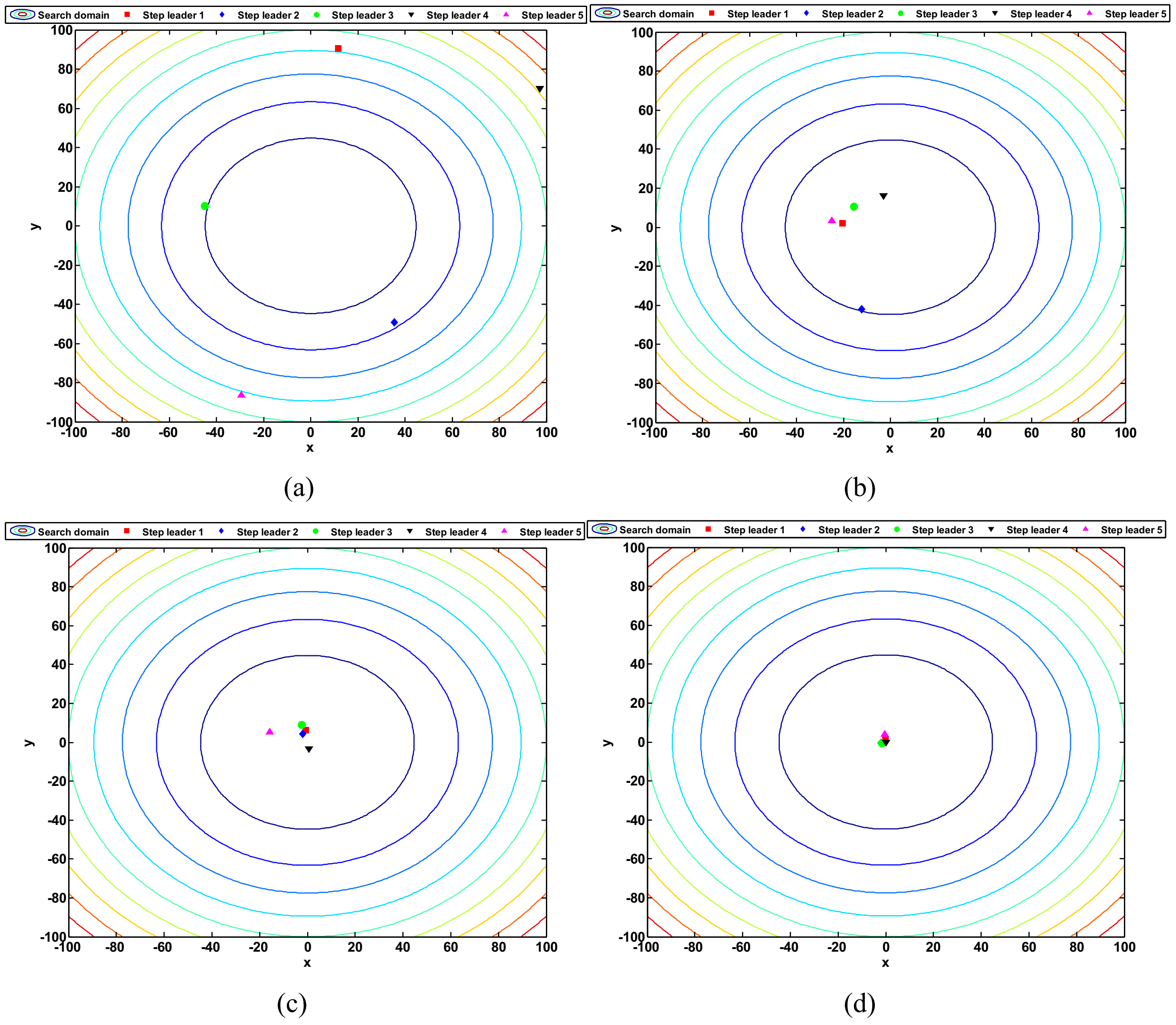

Fig. 4. Numerical positions of individuals after the (a) 2nd, (b) 5th, (c) 8th, and 11th iteration.

Another important property of a stepped leader is forking, in which two simultaneous and symmetrical branches emerge. This phenomenon rarely occurs because of nuclei collision. Any additional channel created during forking increases the number of projectiles by one and hence the population size. In the proposed algorithm, forking is realized in two ways. First, symmetrical channels are created because the nuclei collision of the projectile is realized by using the opposite number as in Eq. (2) [51].

$$\bar{p}_i = a + b - p_i \tag{2}$$

where $\bar{p}_i$ and p_i are the opposite and original projectiles, respectively, in a one-dimensional system; a and b are the boundary limits. This adaptation may improve some of the bad solutions in the population. If forking does not improve channel propagation in the LSA, one of the channels at the forking point is illuminated to maintain the population size.

In the second type of forking, a channel is assumed to appear at a successful step leader tip because of the energy redistribution of the most unsuccessful leader after several propagation trials. The unsuccessful leader can be redistributed by defining the maximum allowable number of trials as channel time. In this case, the population size of step leaders does not increase.

4.3. *Projectile modeling and step leader movement*

Three projectile types are developed to represent the transition projectiles that create the first-step leader population *N*, the space projectiles that attempt to reach the best leader position, and the lead projectiles that represent the best position among *N* numbers of step leaders. In this case, a one-dimensional projectile type is illustrated for clarity.

4.3.1. *Transition projectile*

As mentioned previously, a leader tip is formed at an early stage because the transition forms an ejected projectile from the thunder cell in a random direction. Therefore, it can be modeled as a random number drawn from the standard uniform probability distribution on the open interval representing the solution space. The

Table 1
Parameter settings used in LSA, DSA, BSA, FFA, and PSO.

Parameter	LSA	DSA	BSA	FFA	PSO	HSA
Population size	50	50	50	50	50	50
Max. iteration	500	500	500	500	500	500
c1 and c2	–	–	–	–	2	
p1 and p2	–	0.3.rand	–	–	–	
F	–	–	3.rand	–	–	
γ	–	–	–	1	–	
β	–	–	–	1	–	
α	–	–	–	0.2	–	
Channel time	10	–	–	–	–	
Harmony consideration rate	–	–	–	–	–	0.9
Minimum pitch adjusting rate	–	–	–	–	–	0.4
Maximum pitch adjusting rate	–	–	–	–	–	0.9
Minimum bandwidth	–	–	–	–	–	0.0001
Maximum bandwidth	–	–	–	–	–	1

Table 2
Unimodal and separable test functions.

Function ID	Name	Expression	n	Search space	Function minimum
F1	Sphere	$f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	30	$[-100,100]^n$	0
F2	Step	$f_2(\mathbf{x}) = \sum_{i=1}^{n} \left(\lfloor x_i + 0.5 \rfloor \right)^2$	30	$[-100,100]^n$	0
F3	Quartic	$f_3(\mathbf{x}) = \sum_{i=1}^{n} ix_i^4 + rand(0,1)$	30	$[-1.28,1.28]^n$	0

Table 3
Test 1 global optimization results for benchmark functions in Table 2.

Function ID	Statistics	LSA	DSA	BSA	FFA	PSO	HSA
F1	Best	**1.06220E−19**	0.939999414	1.432970945	0.005027795	1.76307E−06	11.2864678276
	Worst	**2.40506E−06**	37.18006813	53.56753696	0.026468336	0.000281909	39.1564122176
	Average	**4.81067E−08**	11.58475565	9.967361777	0.011961069	2.76248E−05	24.7111807765
	Median	**1.86102E−15**	9.501717692	7.029279930	0.011538451	1.67426E−05	23.9089005983
	Standard deviation	**3.40126E−07**	6.938443045	9.812246890	0.004295896	4.32130E−05	6.67110350060
F2	Best	**0.000000000**	2.000000000	2.000000000	**0.000000000**	**0.00000000**	12.0000000000
	Worst	8.000000000	49.00000000	115.0000000	**0.000000000**	1.00000000	47.0000000000
	Average	3.340000000	15.74000000	13.94000000	**0.000000000**	0.10000000	25.1000000000
	Median	3.000000000	12.00000000	10.50000000	**0.000000000**	0.00000000	24.0000000000
	Standard deviation	2.086007800	11.29531240	17.30094960	**0.000000000**	0.30304580	7.53292094470
F3	Best	0.016268885	0.044973120	0.022657369	**0.008299594**	95.86581258	0.24844627016
	Worst	**0.040825354**	0.359368482	0.102683537	0.116080642	186.9633517	0.76142384317
	Average	**0.024079674**	0.123075150	0.054498487	0.035228916	138.8343114	0.46324111883
	Median	**0.022692902**	0.104513507	0.052128033	0.031611623	135.5998742	0.43320090602
	Standard deviation	**0.005726198**	0.065346271	0.016118308	0.023983202	22.07744869	0.11272240271

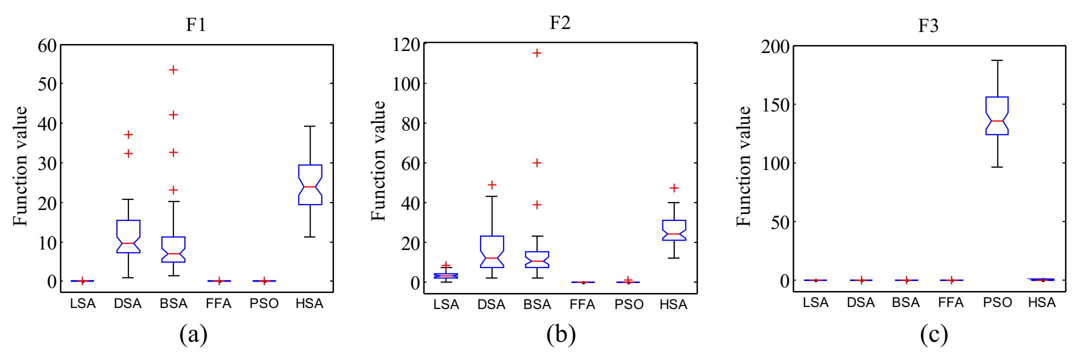

Fig. 5. Variation in global optimization results for benchmark functions in Table 2. (a) F1, (b) F2, and (c) F3.

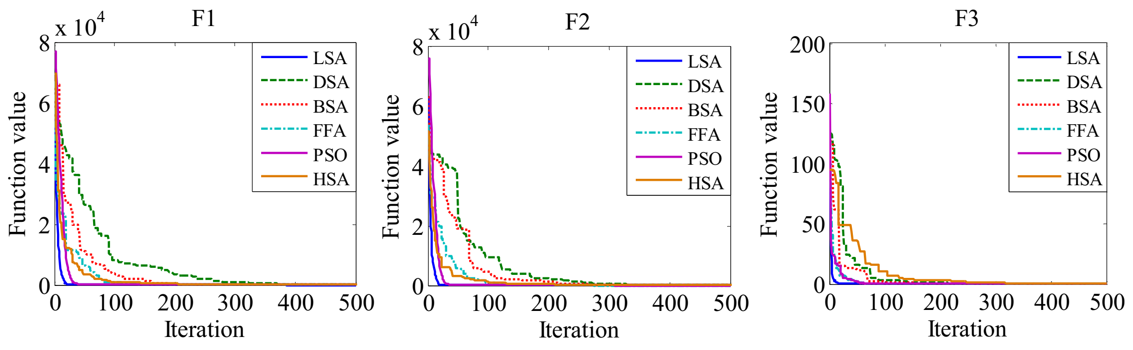

Fig. 6. Convergence characteristic curves for LSA, DSA, BSA, FFA, PSO and HSA in solving F1, F2, and F3.

Table 4
Unimodal and non-separable test functions.

Function ID	Name	Expression	n	Search space	Function minimum
F4	Schwefel 2.22	$f_4(\mathbf{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10,10]^n$	0
F5	Schwefel 1.2	$f_5(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	30	$[-100,100]^n$	0
F6	Schwefel 2.21	$f_6(\mathbf{x}) = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	30	$[-100,100]^n$	0
F7	Rosenbrock	$f_7(\mathbf{x}) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	30	$[-30,30]^n$	0

Table 5
Test 2 global optimization results for benchmark functions in Table 4.

Function ID	Statistics	LSA	DSA	BSA	FFA	PSO	HSA
F4	Best	**2.21758E−07**	0.420804245	0.455148845	0.172154000	0.001456467	0.953321668
	Worst	0.970135549	1.939300935	2.881032922	0.653905407	**0.018902868**	2.248271341
	Average	0.036806544	1.006036631	1.197150810	0.373326823	**0.004923274**	1.457490046
	Median	**0.001069055**	0.921312778	1.140432900	0.370450000	0.004072683	1.427328377
	Standard deviation	0.156233023	0.357910793	0.528510424	0.101431051	**0.003334754**	0.268102707
F5	Best	**9.201365586**	7177.369543	718.2100731	716.4716944	9.488262443	2981.570007
	Worst	125.5038929	35,609.52576	7428.974777	4064.686905	**67.65392364**	12,550.05280
	Average	43.24080402	20,888.93319	2720.310537	1808.806377	**27.86396546**	6878.654840
	Median	34.78963155	21,559.5700	2537.242277	1867.269224	**27.14963995**	6697.852052
	Standard deviation	29.92194448	6907.30897	1182.196483	659.6539690	**9.579498184**	1943.088569
F6	Best	0.118431936	11.97471441	5.864411668	**0.055032609**	0.311741713	7.385350000
	Worst	5.983329378	43.83060890	15.33345411	**0.109483015**	0.935978857	11.97597349
	Average	1.493275733	27.81032820	9.834751443	**0.076694719**	0.610237030	9.385431891
	Median	0.885785820	27.07984848	9.233009358	**0.073437506**	0.585417929	9.129318591
	Standard deviation	1.302827039	7.077083103	2.273287400	**0.014606062**	0.146018388	1.226512199
F7	Best	**0.560036507**	353.5219172	208.3005084	27.85966457	14.36215241	337.8836744
	Worst	**201.6439055**	2347.035592	1242.75774	1850.94738	307.2355336	2811.421866
	Average	**64.28160301**	1108.180713	471.5485416	128.2896097	68.72292596	830.0325560
	Median	73.53003347	1002.19241	394.1795853	**29.38136765**	67.42734810	698.8990872
	Standard deviation	**43.75576111**	572.4209417	231.1419788	278.6344835	57.81076978	474.1998851

Table 6
Multimodal and separable test functions.

Function ID	Name	Expression	n	Search space	Function minimum
F8	Schwefel	$f_8(\mathbf{x}) = -\sum_{i=1}^{n} \left(x_i \sin \left(\sqrt{\lvert x_i \rvert} \right) \right)$	30	$[-500,500]^n$	−12,569.5
F9	Rastrigin	$f_9(\mathbf{x}) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]$	30	$[-5.12,5.12]^n$	0
F10	Foxholes	$f_{10}(\mathbf{x}) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \right]^{-1}$	2	$[-65.53,65.53]^n$	1
F11	Branin	$f_{11}(\mathbf{x}) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10$	2	$[-5,10] \times [0,15]$	0.398

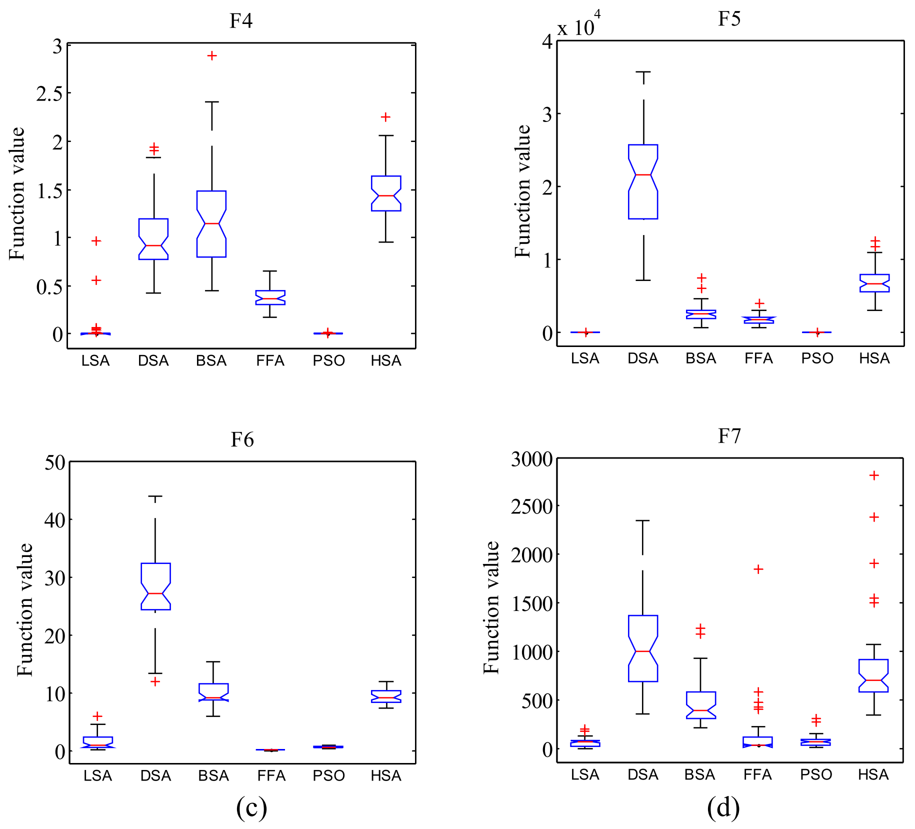

Fig. 7. Variation in global optimization results for benchmark functions in Table 4. (a) F4, (b) F5, (c) F6, and (d) F7.

probability density function $f(x^T)$ of the standard uniform distribution can be represented as

$$f(x^T) = \begin{cases} 1/b-a & \text{for} \quad a \le x^T \le b \\ 0 \quad \text{for} & x < a \, or \, x^T > b \end{cases} \tag{3}$$

where x^T is a random number that may provide a solution or the initial tip energy E_{sl_i} of the step leader sl_i ; a and b are the lower and upper bounds, respectively, of the solution space. For a population of N step leaders $SL = [sl_1, sl_2, sl_3, \ldots, sl_N]$, N random projectiles $P^T = [p_1^T, p_2^T, p_3^T, \ldots, p_N^T]$ that satisfy the solution dimension are required.

4.3.2. Space projectile

Once the N step leader tips are evolved, the leaders need to move using energetic projectiles by ionizing the section in the vicinity of the old leader tip in the next step *step+1*. The position of the space projectile $P^S = [p_1^S, p_2^S, p_3^S, \ldots, p_N^S]$ at *step+1* can be partially modeled as a random number generated from the exponential

Table 7
Test 3 global optimization results for benchmark functions in Table 6.

Function ID	Statistics	LSA	DSA	BSA	FFA	PSO	HSA
F8	Best	−9193.9122	**−10,563.3588**	−10,357.6423	−7397.28009	−6932.44144	−8180.83229
	Worst	−6488.5802	**−9443.46858**	−9101.11690	−4336.23426	−3125.63430	−6607.59209
	Average	−8001.3887	**−10,037.5543**	−9615.76912	−5900.32722	−4231.37873	−7395.27011
	Median	−8058.7179	**−10,005.1120**	−9611.69924	−5867.31737	−3979.33993	−7388.96021
	Standard deviation	669.159310	278.960960	**253.0050960**	655.5192820	869.1484330	350.4224603
F9	Best	40.7932709	33.4849279	53.1810068	**11.3594362**	17.18796618	59.70960631
	Worst	105.465281	63.5140164	82.5563962	**56.5153988**	65.86114792	107.1312960
	Average	62.7618960	47.0645809	65.7590797	**26.2570093**	43.05418116	87.03621984
	Median	59.6974248	46.6551322	66.3643674	**24.6150801**	42.61976479	85.55923249
	Standard deviation	14.9153021	**7.19733732**	8.02990073	9.14924055	10.62865813	10.78576220
F10	Best	**0.998003838**	**0.998003838**	**0.998003838**	**0.998003838**	**0.998003838**	**0.998003838**
	Worst	2.982105157	**0.998003838**	**0.998003838**	3.068711200	6.903335694	5.988897936
	Average	1.077446947	**0.998003838**	**0.998003838**	1.875598153	1.790398060	1.605937747
	Median	**0.998003838**	**0.998003838**	**0.998003838**	1.992087793	1.992030900	**0.998003838**
	Standard deviation	0.337979509	**3.36448E−16**	**3.36448E−16**	0.668873415	1.259895845	1.371648196
F11	Best	**0.397887358**	**0.397887358**	**0.397887358**	**0.397887358**	**0.397887358**	0.397887358
	Worst	**0.397887358**	**0.397887358**	**0.397887358**	**0.397887375**	**0.397887358**	0.495020685
	Average	**0.397887358**	**0.397887358**	**0.397887358**	**0.397887360**	**0.397887358**	0.407772879
	Median	**0.397887358**	**0.397887358**	**0.397887358**	**0.397887360**	**0.397887358**	0.398339833
	Standard deviation	**1.68224E−16**	7.79967E−12	1.43316E−12	3.13056E−09	**1.68224E−16**	0.021637622

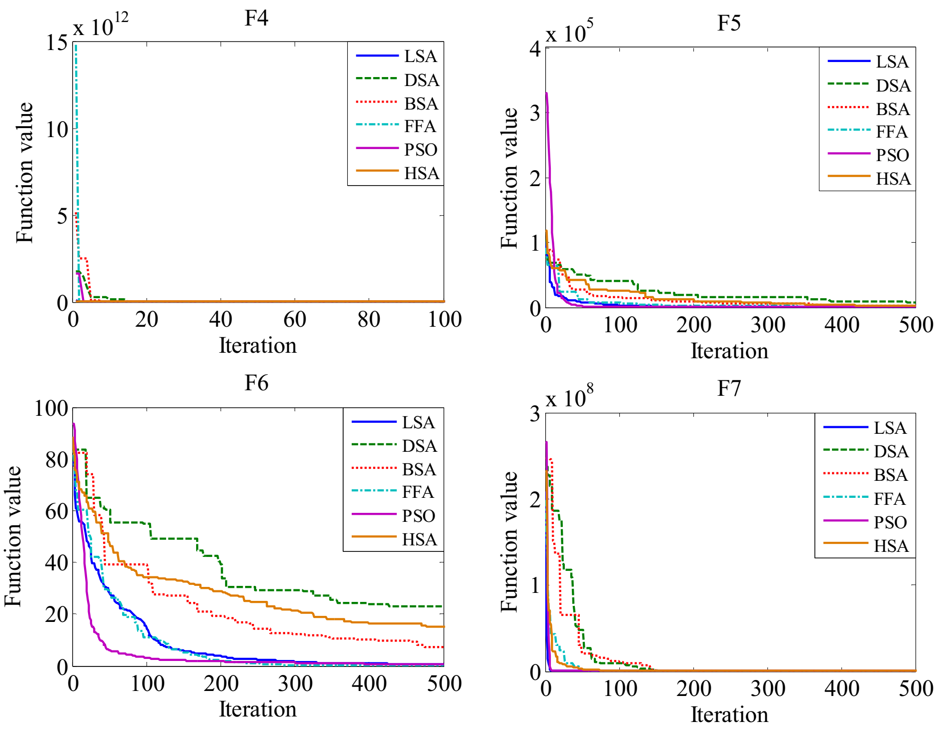

Fig. 8. Convergence characteristic curves for LSA, DSA, BSA, FFA, PSO and HSA in solving F4, F5, F6, and F7.

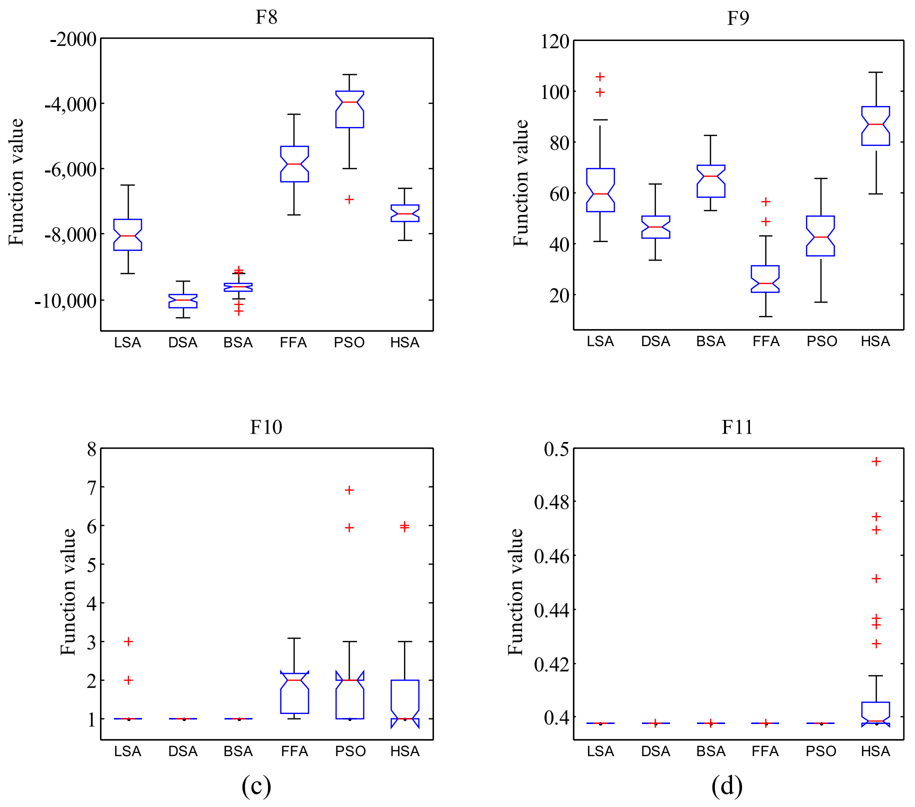

Fig. 9. Variation in global optimization results for benchmark functions in Table 6. (a) F8, (b) F9, (c) F10, and (d) F11.

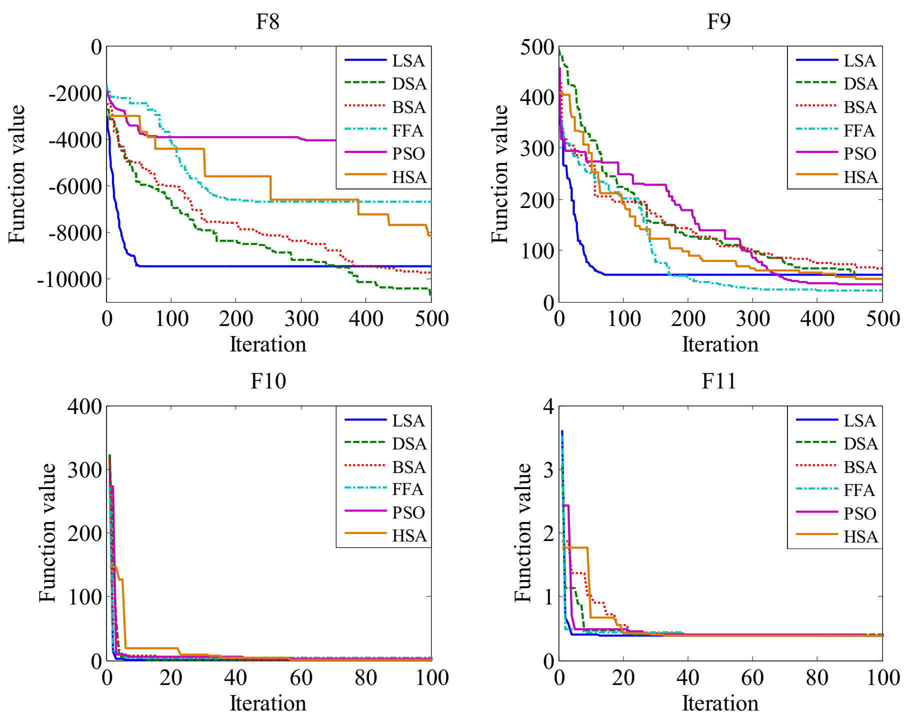

Fig. 10. Convergence characteristic curves for LSA, DSA, BSA, FFA, PSO and HSA in solving F8, F9, F10, and F11.

distribution with shaping parameter μ. The probability density function $f(x^S)$ of an exponential distribution is given by

$$f(x^S) = \begin{cases} \frac{1}{\mu} e^{-x^S/\mu} & \text{for} \quad x^S \geq 0 \\ 0 & \text{for} \quad x^S \leq 0 \end{cases} \tag{4}$$

Eq. (4) shows that the space projectile position or the direction in the next step can be controlled by shaping parameter μ. In the LSA, μ_i for a specific space projectile p_i^S is taken as the distance between the lead projectile p^L and the space projectile p_i^S under consideration. With this definition, the position of p_i^S at *step+1* can be written as

$$p_{i\_new}^S = p_i^S \pm \exp rand(\mu_i) \tag{5}$$

where *exprand* is an exponential random number. If p_i^S is negative, then the generated random number should be subtracted because Eq. (4) only provides positive values. However, the new position $p_{i\_new}^S$ does not guarantee stepped leader propagation or channel formation unless the projectile energy $E_{p\_i}^S$ is greater than the step leader E_{sl_i} to extend the channel or until a good solution is found. If $p_{i\_new}^S$ provides a good solution at *step+1*, then the corresponding stepped leader sl_i is extended to a new position sl_{i_new}, and p_i^S is updated to $p_{i\_new}^S$. Otherwise, they remain unchanged until the next step. If $p_{i\_new}^S$ extends sl_{i_new} beyond the recent, most extended leader during this process, then it becomes the lead projectile.

4.3.3. *Lead projectile*

Presumably, the step leader that has traveled nearest to the ground and the projectile associated with it do not have enough potential to ionize large sections in front of the leader tip. Therefore, the lead projectile can be modeled as a random number drawn from the standard normal distribution with the shape parameter μ and the scale parameter σ. The normal probability density function $f(x^L)$ is expressed as

$$f(x^L) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x^L-\mu)^2/2\sigma^2} \tag{6}$$

Eq. (6) shows that the randomly generated lead projectile can search in all directions from the current position defined by the shape parameter. This projectile also has an exploitation ability defined by the scale parameter. In the LSA, μ_L for the lead projectile p^L is taken as p^L, and the scale parameter σ_L exponentially decreases as it progresses toward the Earth or as it finds the best solution. With this definition, the position of p^L at *step+1* can be written as

$$p_{new}^L = p^L + normrand(\mu_L, \sigma_L) \tag{7}$$

where *normrand* is a random number generated by the normal distribution function. Similarly, the new lead projectile position p_{new}^L does not guarantee step leader propagation unless the projectile lead projectile energy $E_{p\_i}^L$ is greater than the step leader E_{sl_i} to extend to a satisfactory solution. If p_{new}^L provides a good solution at *step+1*, then the corresponding step leader sl_i is extended to a new position sl_{L_new}, and p^L is updated to p_{new}^L. Otherwise, they remain unchanged until the next step, as in the case of the space projectile.

The whole procedure of the LSA is summarized as a flowchart in Fig. 2.

5. Illustrative example of LSA functionality

The flowchart and the details given in Section 4 can be used as bases to develop a program code for executing the LSA in any programming language. In this section, the benchmark Sphere function is used to visualize and show the functionality of the LSA. The Sphere function in this illustrative example has two variables, and the number of step leaders created by the transition projectiles in this example is assumed to be 5. Thus, population size is 5×2. The LSA aims to find the global minimum of the two-dimensional Sphere function given in Eq. (8). In this case, the global minimum occurs when x_1 and x_2 are equal to zero. The search domain is defined as $-100 \leq x, y \leq 100$.

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \quad (8)$$

Five locations of the individual step leaders and the propagations of their channels from its creation using transition projectile (initialization) until the 11th iteration are graphically exhibited in Fig. 3. The status, channel elimination steps, forking events, and projectile movements are clearly shown in the figure. The numerical position of each step leader in the search domain after the 2nd, 5th, 8th, and 11th iteration are shown in Fig. 4(a)–(d). The location of the step leaders moves toward the global minimum and converges as the number of integration increases. From this example and the procedure given in Fig. 2, it should be noted that the main complexity of the LSA lies in the forking mechanism and channel elimination procedure. Furthermore, it is also important to observe that the channel time is the only algorithm dependent parameter of the LSA that requires tuning. Experiments shows that channel time counter with 1–5 percent of the predefined maximum iteration number provide excellent results for most of the benchmark problems.

6. Test results and validation

The proposed LSA is first tested and validated using a well-utilized set of 23 benchmark functions created by Rashedi et al. [11], Rao et al. [38], and Yao [20] while developing their optimization algorithms. These functions are characterized as linear, non-linear, continuous, discontinuous, unimodal, multimodal, convex, non-convex, separable, and non-separable. However, functional characteristics such as modality, separability, and dimensionality are considerably important when testing and validating a new algorithm. The modality of a function refers to the number of vague peaks in the function surface. A function is multimodal if it has two or more vague peaks. An algorithm that encounters these peaks while searching may be trapped in one of the local minima. Meanwhile, separability indicates the difficulty level of various benchmark functions. Typically, separable functions are easier to solve than non-separable functions because each variable of a function is independent of the other variables. The difficulty of a problem also increases with function dimensionality. For highly non-linear problems, dimensionality may be a significant barrier for almost all optimization algorithms. In this study, the first 23 benchmark functions are categorized based on modality and separability, and four tests are conducted to evaluate the reliability and efficiency of the proposed algorithm. Secondly, one of the most deeply investigated problems in computational mathematics known as the traveling salesman problem (TSP) [52] involving several constrains (considered as 24th benchmark) is given to LSA as a challenging problem in fifth test. In each test, the performance of the proposed algorithm is compared with other standard heuristic optimization methods, namely, DSA, BSA, FFA, PSO and HSA. For a fair comparison, the population size and maximum iteration are set to 50 and 500, respectively, in all cases. The algorithm-dependent parameter settings for each algorithm in the comparison are listed in Table 1 as recommended in the literature [3,12,29,32,42].

6.1. Test 1

This test is conducted to evaluate the reliability and efficiency of the LSA in searching the global minimum value when it is subjected to benchmark functions with unimodal and separable characteristics. Table 2 presents the details of these functions. This test also compares the LSA with four other methods, namely, DSA, BSA, FFA,

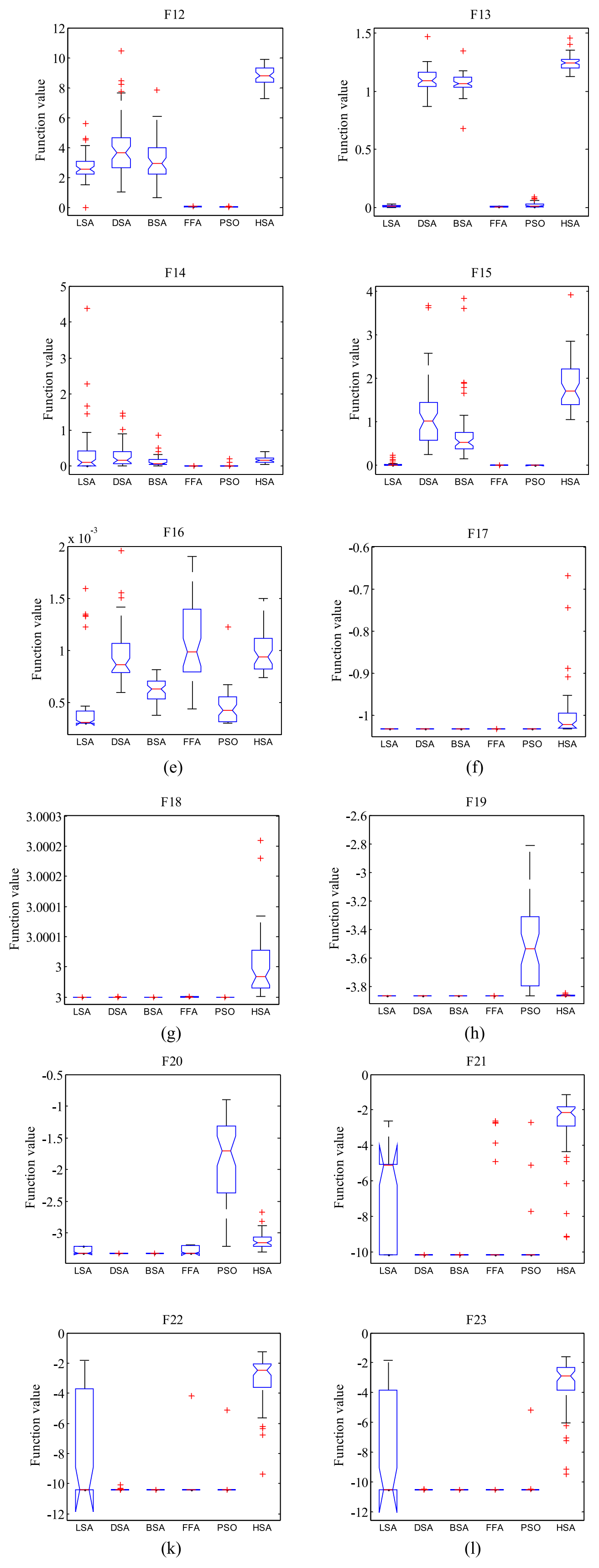

Fig. 11. Variation in global optimization results for benchmark functions in Table 6. (a) F12, (b) F13, (c) F14, (d) F15, (e) F16, (f) F17, (g) F18, (h) F19, (i) F20, (j) F21, (k) F22, and (l) F23.

Table 8
Multimodal and non-separable test functions.

Function ID	Name	Expression	n	Search space	Function minimum
F12	Ackley	$f_{12}(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	$[-32,32]^n$	0
F13	Griewank	$f_{13}(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600,600]^n$	0
F14	Penalized	$f_{14}(\mathbf{x}) = \frac{\pi}{n}\left\{10\sin^2(\pi y_i) + \sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right] + (y_n-1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i+1}{4}, \quad u(x_i, a, k, m) = \begin{cases} k(x_i-a)^m, & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i-a)^m, & x_i < -a \end{cases}$	30	$[-50,50]^n$	0
F15	Penalized 2	$f_{15}(\mathbf{x}) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right] + (x_n-1)^2[1+\sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$ $u(x_i, a, k, m) = \begin{cases} k(x_i-a)^m, & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i-a)^m, & x_i < -a \end{cases}$	30	$[-50,50]^n$	0
F16	Kowalik	$f_{16}(\mathbf{x}) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$	4	$[-5,5]^n$	0.000307
F17	6-Hump Camel Back	$f_{17}(\mathbf{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5,5]^n$	−1.0316285
F18	GoldStein–Price	$f_{18}(\mathbf{x}) = \left[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\right] \times \left[30+(2x_1-3x_2)^2 \times (18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\right]$	2	$[-2,2]^n]^n$	3
F19	Hartman 3	$f_{19}(\mathbf{x}) = -\sum_{i=1}^{4} c_i\exp\left[-\sum_{j=1}^{4} a_{ij}(x_j-p_{ij})^2\right]$	4	$[0,1]^n$	−3.86
F20	Hartman 6	$f_{20}(\mathbf{x}) = -\sum_{i=1}^{4} c_i\exp\left[-\sum_{j=1}^{6} a_{ij}(x_j-p_{ij})^2\right]$	6	$[0,1]^n$	−3.32
F21	Shekel 5	$f_{21}(\mathbf{x}) = -\sum_{i=1}^{5}\left[(x-a_i)(x-a_i)^T + c_i\right]^{-1}$	4	$[0,10]^n$	−10
F22	Shekel 7	$f_{22}(\mathbf{x}) = -\sum_{i=1}^{7}\left[(x-a_i)(x-a_i)^T + c_i\right]^{-1}$	4	$[0,10]^n$	−10
F23	Shekel 10	$f_{23}(\mathbf{x}) = -\sum_{i=1}^{10}\left[(x-a_i)(x-a_i)^T + c_i\right]^{-1}$	4	$[0,10]^n$	−10

Table 9
Test 3 global optimization results for benchmark functions in Table 8.

Function ID	Statistics	LSA	DSA	BSA	FFA	PSO	HSA
	Best	**8.73041E−08**	1.028135299	0.655100657	0.022344835	0.001075635	7.2791173817
	Worst	5.591537337	10.46621317	7.854634520	0.087412368	**0.080060126**	9.8869529647
F12	Average	2.686199995	4.047462107	3.153910594	0.051178370	**0.005497460**	8.8199175672
	Median	2.537270358	3.646161760	2.916464245	0.050700286	**0.002850224**	8.7968767855
	Standard deviation	0.910802774	1.984370448	1.491250651	0.013738065	**0.011087520**	0.6124944637
	Best	**2.22045E−16**	0.867850097	0.679371668	0.002458997	3.53885E−07	1.1252532007
	Worst	0.024590437	1.473488056	1.348418301	**0.008959060**	0.090389522	1.4554682369
F13	Average	0.007241875	1.107016779	1.068844500	**0.005840751**	0.019679948	1.2441128611
	Median	0.007396040	1.091219304	1.066875504	**0.005649562**	0.009879577	1.2436667371
	Standard deviation	0.006753356	0.093916433	0.086865408	**0.001434566**	0.024205903	0.0640798050
	Best	**2.23008E−15**	0.011463807	0.011257896	9.14932E−05	1.10504E−08	0.0456882410
	Worst	4.379867485	1.468316408	0.857216546	**0.000547669**	0.207337908	0.4108890986
F14	Average	0.358172550	0.291749952	0.135375902	**0.000237647**	0.018659838	0.1796597064
	Median	0.103669020	0.170607599	0.075928645	0.00023083	**1.90335E−07**	0.1594349815
	Standard deviation	0.743960008	0.336701290	0.153384387	**0.000100091**	0.054169508	0.0974110156
	Best	**2.75229E−19**	0.257978073	0.149026090	0.000727556	1.83924E−07	1.0491062235
	Worst	0.233414464	3.681026161	3.847811408	**0.005349947**	0.010996712	3.9303745643
F15	Average	0.024148546	1.170006171	0.756466028	**0.002240514**	0.004442821	1.8325821059
	Median	0.010987366	1.015913093	0.526940255	0.001904127	**1.14847E−05**	1.7006483170
	Standard deviation	0.047279168	0.784576410	0.749431707	**0.001040353**	0.005406675	0.5498874505
	Best	**0.000307486**	0.000601711	0.000378591	0.000443132	0.000307541	0.0007385823
	Worst	0.001594050	0.001954465	**0.000817061**	0.001900659	0.001223173	0.0015003604
F16	Average	0.000534843	0.000960175	0.000626118	0.001062472	**0.000475754**	0.0009634515
	Median	**0.000309116**	0.000864618	0.000632434	0.000984346	0.000430698	0.0009339154
	Standard deviation	0.000424113	0.000268150	**0.000108003**	0.000359353	0.000194548	0.0001757872
	Best	−**1.031628453**	−**1.031628453**	−**1.031628453**	−**1.031628453**	−**1.031628453**	−1.031627142
	Worst	−**1.031628453**	−**1.031628453**	−**1.031628453**	−**1.031628442**	−**1.031628453**	−0.668652888
F17	Average	−**1.031628453**	−**1.031628453**	−**1.031628453**	−**1.031628451**	−**1.031628453**	−0.996514478
	Median	−**1.031628453**	−**1.031628453**	−**1.031628453**	−**1.031628452**	−**1.031628453**	−1.022330219
	Standard deviation	**0.000000000**	**0.000000000**	**0.000000000**	2.82695E−09	**0.000000000**	0.0681095561
	Best	**3.000000000**	**3.000000000**	**3.000000000**	3.000000001	**3.000000000**	3.0000004574
	Worst	**3.000000000**	3.000000380	**3.000000000**	3.000000111	**3.000000000**	3.0002588975
F18	Average	**3.000000000**	3.000000008	**3.000000000**	3.000000029	**3.000000000**	3.0000523753
	Median	**3.000000000**	**3.000000000**	**3.000000000**	3.000000024	**3.000000000**	3.0000290691
	Standard deviation	**3.34499E−15**	5.38203E−08	3.50940E−15	2.56076E−08	4.11342E−15	0.0000538472
	Best	−**3.862782148**	−**3.862782148**	−**3.862782148**	−**3.862782148**	−**3.862782148**	−3.862720257
	Worst	−**3.862782148**	−**3.862782148**	−**3.862782148**	−3.862782143	−2.810142824	−3.843936930
F19	Average	−**3.862782148**	−**3.862782148**	−**3.862782148**	−3.862782147	−3.508607946	−3.860210514
	Median	−**3.862782148**	−**3.862782148**	−**3.862782148**	−3.862782147	−3.535082985	−3.861372004
	Standard deviation	**0.000000000**	**0.000000000**	**0.000000000**	9.16371E−10	0.307782201	0.0032566497
	Best	−**3.321995172**	−**3.321995172**	−**3.321995172**	−3.321995168	−3.20310205	−3.303125207
	Worst	−3.203102050	−**3.321995016**	−3.321977602	−3.185406645	−0.89430411	−2.664423570
F20	Average	−3.272060061	−**3.321995167**	−3.321994590	−3.267673897	−1.85270547	−3.121636655
	Median	−3.321995172	−**3.321995172**	−3.321995164	−3.321995148	−1.70210590	−3.155957708
	Standard deviation	0.059276470	**2.32293E−08**	2.62866E−060	0.061982679	0.655286473	0.1334711000
	Best	−**10.15319968**	−**10.15319968**	−**10.15319968**	−10.15319938	−**10.15319968**	−9.127235482
	Worst	−2.630471668	−10.14497223	−**10.14912772**	−2.63047156	−2.682860396	−1.141856571
F21	Average	−7.027319823	−10.15283380	−**10.15309398**	−8.42709131	−8.653837241	−2.782671474
	Median	−5.100772140	−10.15319541	−10.15319937	−10.15319694	−**10.15319968**	−2.167447446
	Standard deviation	3.156152099	0.001254219	**0.000583586**	3.120294072	2.808248893	1.8136031039
	Best	−**10.40294057**	−**10.40294057**	−**10.40294057**	−10.40294030	−**10.40294057**	−9.367837367
	Worst	−1.837592971	−10.10111461	−**10.40224208**	−4.180110796	−5.128822797	−1.259142499
F22	Average	−7.136702131	−10.39358640	−**10.40291730**	−10.27848169	−10.08649321	−3.045778915
	Median	−10.40294057	−10.40294007	−10.40294049	−10.40293826	−**10.40294057**	−2.485769122
	Standard deviation	3.514977367	0.044169900	**0.000106808**	0.880040697	1.265249900	1.6454997595
	Best	−**10.53640982**	−**10.53640982**	−**10.53640982**	−10.53640934	−**10.53640982**	−10.15193804
	Worst	−1.859480301	−10.45819454	−10.53597889	−**10.53640384**	−5.175646740	−2.629257871
F23	Average	−7.910438367	−10.53316951	−10.53639356	−**10.53640769**	−10.32051223	−4.204341501
	Median	−**10.53640982**	−10.53639613	−10.53640963	−10.53640783	−10.53640982	−2.682462288
	Standard deviation	3.596042666	0.012504559	6.41563E−050	**1.11512E−06**	1.0609040600	3.0085418461

PSO and HSA, for validation. Each benchmark function is tested 50 times. The results considering the best, worst, average, median, and standard deviation of the objective function are shown in Table 3. The best performance for each function is boldfaced. The LSA reaches the best global minimum or near global minimum for F1 and F2. For F3, the LSA fails to reach the best solution. However, its performance is acceptable because it has the lowest standard deviation and average global minimum value. This result is also clearly observed in the box plot (Fig. 5) constructed from the data obtained from 50 runs. Fig. 5 shows that the performance of the LSA is

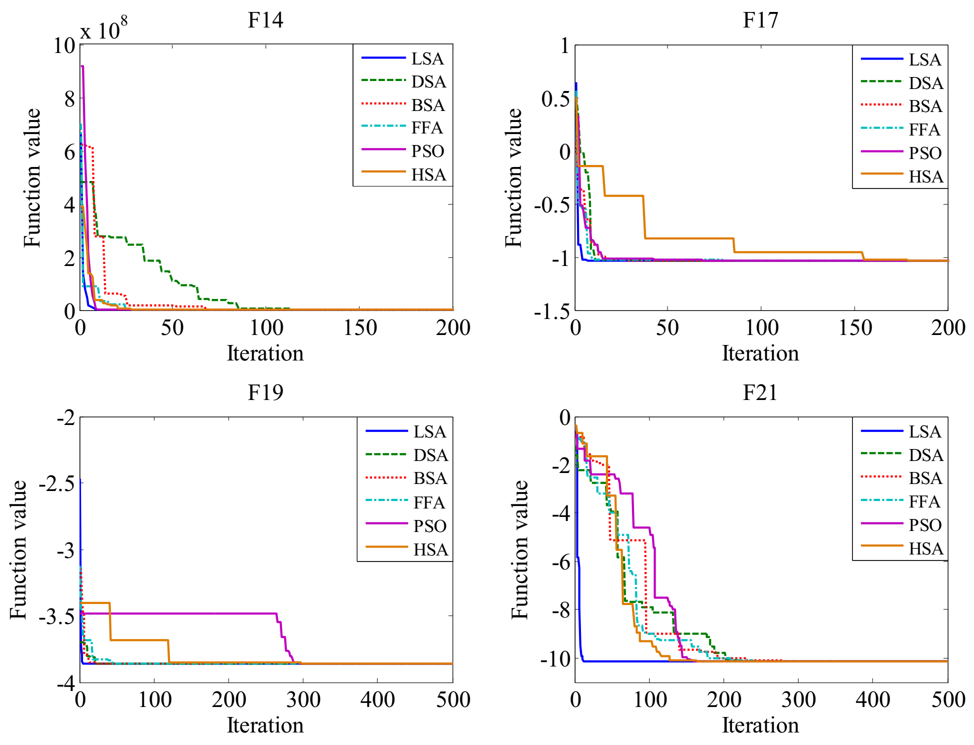

Fig. 12. Convergence characteristic curves for LSA, DSA, BSA, FFA, PSO and HSA in solving F14, F17, F19, and F21.

satisfactory for all three functions because the 25th and 75th percentiles of the samples collected for the LSA fall toward the minimum solution with a narrow interquartile range. Meanwhile, the median of the sample for the case of HSA and PSO (indicated by a red line) is very far from the best solution value for F1 and F3 respectively.

During unimodal function optimization, the performances of the algorithms in terms of computation time are almost similar. Therefore, the convergence characteristic curves shown in Fig. 6 are using to compare the performance. From the figure, it can be noted that the LSA converges faster than the other methods and thus possesses superior convergence characteristics for this type of function optimization.

6.2. Test 2

To observe the performance and consistency of the LSA in solving the unimodal and non-separable functions, four benchmark functions given in Table 4 are experimented on in this test: Schwefel 2.22 (F4), Schwefel 1.2 (F5), Schwefel 2.21 (F6), and Rosenbrock (F7). These functions have the same dimensions ($n = 30$) as those used in Test 1, but the difficulty level is higher because these functions are non-separable. The test results acquired using the LSA are compared with those obtained using the four optimization methods (Table 5). The best performance for each function is boldfaced. Table 5 shows that the LSA can find the best near-optimum solution for functions F4, F5, and F7. For F6, the FFA finds a better solution

Table 10
Symmetric traveling salesman problem considering 20 cities.

Function ID	Name	Expression	n	Cities coordinates			Function minimum
				u	x	y	
F24	TSP	$x_{ij} = \begin{cases} 1 & \text{if the path goes from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases}$ $f_{24}(\mathbf{u}) = \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij} x_{ij}$ where $c_{ij} = \sqrt{\left(u_i(x) - u_j(x)\right)^2 + \left(u_i(y) - u_j(y)\right)^2}$ u_i = artificial variable Subjected to : $\sum_{i=0, i \neq j}^{n} x_{ij} = 1 \quad j = 0, \ldots, n$ $\sum_{j=0, j \neq i}^{n} x_{ij} = 1 \quad i = 0, \ldots, n$ $1 \leq u_i \leq n$ $u_i - u_j + n x_{ij} \leq n - 1 \quad 1 \leq i \neq j \leq n$	20	1	4.38	7.51	Unknown
				2	3.81	2.55	
				3	7.65	5.05	
				4	7.9	6.99	
				5	1.86	8.90	
				6	4.89	9.59	
				7	4.45	5.47	
				8	6.46	1.38	
				9	7.09	1.49	
				10	7.54	2.57	
				11	2.76	8.40	
				12	6.79	2.54	
				13	6.55	8.14	
				14	1.62	2.43	
				15	1.19	9.29	
				16	4.98	3.49	
				17	9.59	1.96	
				18	3.40	2.51	
				19	5.85	6.16	
				20	2.23	4.73	

Table 11
Test 5 global optimization results for TSP benchmark function in Table 10.

Function ID	Statistics	LSA	DSA	BSA	FFA	PSO	HSA
F24	Best	**43.82008497**	44.86438626	45.48283621	50.71531388	47.0055046	46.95036608
	Worst	59.49844699	**53.26309528**	53.82535584	66.32757381	64.5403155	63.09636493
	Average	51.69388504	49.80562551	**49.38430033**	56.01030563	57.58814409	57.34555458
	Median	52.07472982	50.27784898	**49.58248118**	55.25919862	57.82057792	58.32401400
	Standard deviation	3.897472510	2.17282475	**2.022100734**	3.150618474	4.12618208	3.912720090

compared with the LSA. To determine the consistency and overall performance of the LSA, the data obtained from 50 runs are plotted (Fig. 7). The performance of the LSA is relatively satisfactory for all the four tested functions because the 25th and 75th percentiles of the samples collected for the LSA fall toward the minimum solution with a narrow interquartile range. Fig. 8 shows the convergence characteristic curves of the various optimization methods obtained while solving benchmark functions F4, F5, F6, and F7. Among the various algorithms, the LSA finds the best solutions in less than 200 iterations on average for F4, F5, and F6. The DSA, BSA, HSA and FFA need additional iterations. For F6, the convergence characteristics of the LSA are similar to those of the FFA.

6.3. Test 3

In this test, the difficulty level of the optimization problem is increased by using multimodal and separable functions (Table 6). F8 and F9 are high-dimensional problems while F10 and F11 are low-dimensional problems. Each benchmark function is tested again for 50 times. The results considering the best, worst, average, median, and standard deviation of the objective functions are shown in Table 7. The best performance for each function is boldfaced. All algorithms obtain the best global minimum or near-global minimum for F10 and F11. For F8 and F9, the LSA fails to determine the best solution. For F8, the DSA obtains the best solution. For F9, the FFA outperforms the other algorithms in finding the best solution. However, the comparative results shown in Fig. 9 show that the performance of the LSA is acceptable because its results are comparable to those of the other algorithms even when finding the solution for the Rastrigin function (F9). These results prove the capability of the LSA to escape from any local minimum.

Fig. 10 shows that the convergence rate of the LSA is much faster at the initial stages than at the later stages. Here gain the time taken by various algorithms to find the optimum solutions of test functions are comparable and thus the convergence rate of the LSA highlights its advantages.

6.4. Test 4

The exploration and exploitation abilities of the proposed LSA are also tested with 12 multimodal and non-separable high- and low-dimensional benchmark functions. Table 8 presents the details of these functions. Similar to the previous test analysis, this test also compares the LSA with five other methods, namely, DSA, BSA, FFA, PSO and HSA, for validation using the same statistical indices (Table 9). The best performance for each function is boldfaced. The LSA reaches the best global minimum or near-global minimum for all tested functions. This result confirms the exploration and exploitation abilities of the proposed LSA. Moreover, the box plots shown in Fig. 11 show that the performance of the LSA is comparatively satisfactory for most of the tested functions. Nonetheless, for the Shekel 5 test function (F21), the LSA cannot find satisfactory solutions in most of the test runs because the median and the 25th and 75th percentiles of the samples collected for the LSA extend from the true solution. However, for the other Shekel functions (F22 and F23), the performance of the LSA is acceptable because its median is close to the actual solution. The convergence characteristics of selected functions, namely, F14, F17, F19, and F21, are depicted in Fig. 12.

6.5. Test 5

The aim of this test is to find the capability of the proposed LSA in finding the solution for one of the most intensely investigated benchmark problems in computational mathematics known as the traveling salesman problem (TSP). In TSP, for a given a list of cities and the distances between each pair of cities, the objective is to find the shortest possible route that visits each city exactly once and returns to the origin city. The way of visiting all the cities is simply the order in which the cities are visited and the ordering is called a tour. It is an NP-hard problem. Table 10 presents the details of TSP functions.

Like the previous tests, TSP benchmark function is also evaluated again for 50 times with all the tested algorithms. The results considering the best, worst, average, median, and standard deviation of the objective function obtained by LSA, DSA, BSA, FFA, PSO and HAS are shown in Table 11. The LSA again finds the best minimum distance tour for TSP (F24). Moreover, in terms of consistency, LSA performance is acceptable because it has similar standard deviation and lower average global minimum value compared to FFA, PSO and HSA. This can also be clearly seen in the boxplot shown in Fig. 13. Finally for the completeness of the analysis, the optimum tours obtained by various optimization methods utilized in the comparison are given in Fig. 14 together with distance matrix and cities locations. From the figure it can be observed that none of the optimization method gives the same tour (results) like what is experienced in solving other benchmark functions. LSA find the best route in the order $4 \to 13 \to 3 \to 12 \to 8 \to 9 \to 17 \to 10 \to 16 \to 7 \to 20 \to 14 \to 18 \to 2 \to 11 \to 5 \to 15 \to 6 \to 1 \to 19$ with minimum tour distance of 43.82.

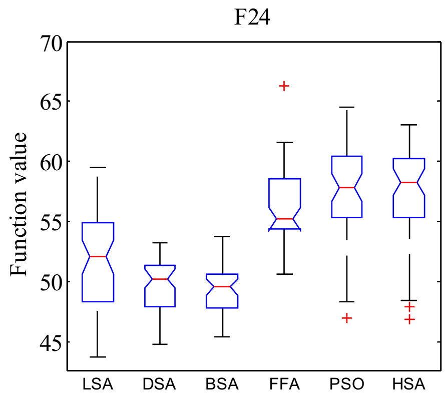

Fig. 13. Variation in global optimization results for TSP benchmark function (F24) in Table 9.

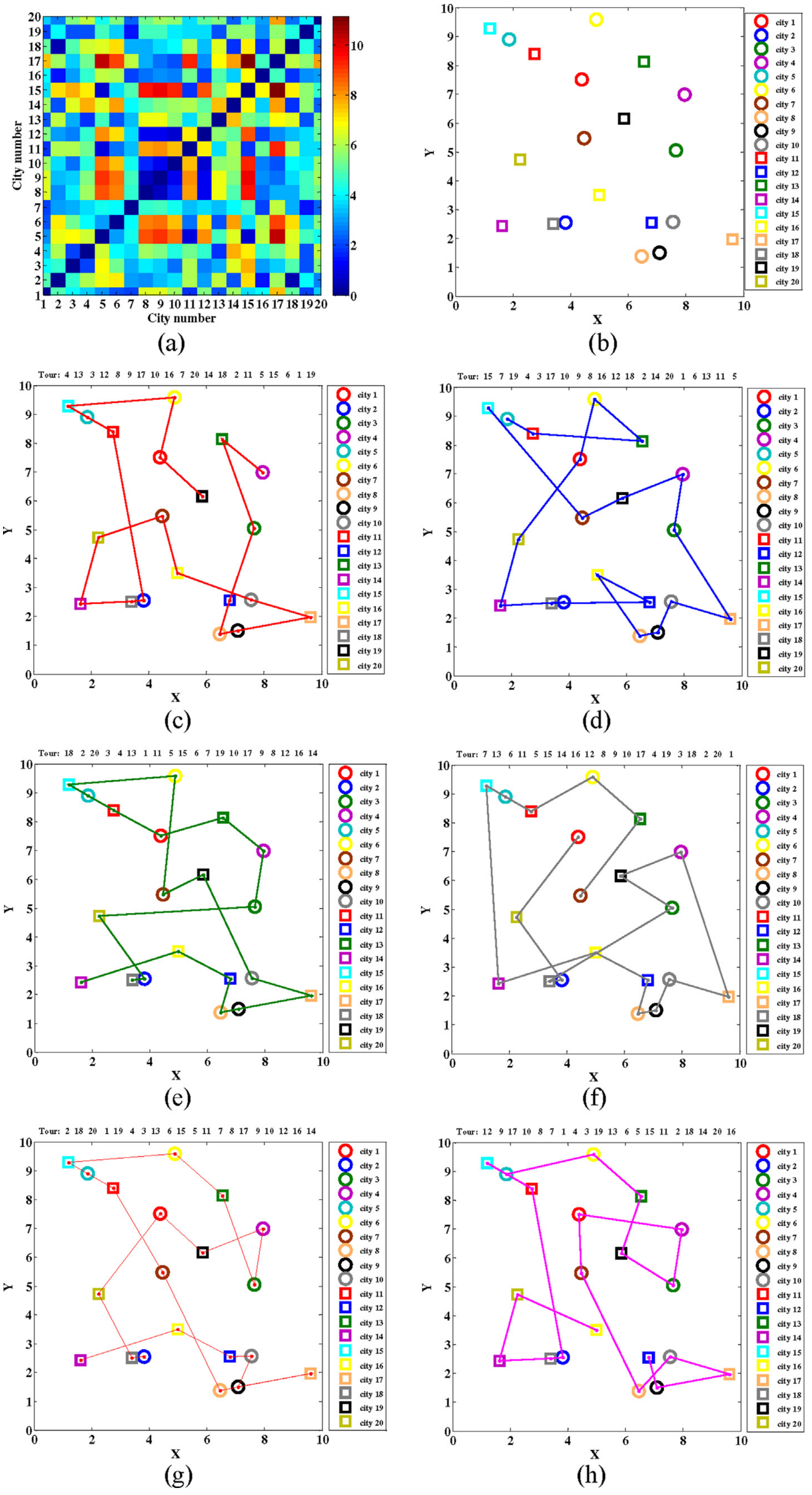

Fig. 14. Optimum tours for TSP benchmark function (F24) obtained by various optimization methods. (a) Distance matrix, (b) locations of cities, (c) tour by LSA, (d) tour by DSA, (e) tour by BSA, (f) tour by FFA, (g) tour by PSO, and (h) tour by HSA.

7. Conclusion

This study introduces a novel nature-inspired optimization method called the LSA. It is formulated based on lightning, which originates from thunderstorm. Similar to other metaheuristic algorithms, the LSA also needs a population to initiate a search. It considers the involvement of fast particles known as projectiles in the search. The process is divided into three projectile types, namely, transition, space, and lead projectile. The transition projectile initiates the population of solutions. The space and lead projectiles at the step leader tips perform unique exploration and exploitation activities to search for the optimum solution. In contrast to that of the counterparts of the LSA, the major exploration feature of the proposed algorithm is modeled using the exponential random behavior of space projectile and the concurrent formation of two leader tips at fork points using opposition theory. Meanwhile, the major exploitation process is designed to be handled by the lead projectile with a normal random search. To evaluate the reliability and efficiency of the proposed algorithm, the LSA is tested using a well-utilized set of 24 benchmark functions with various characteristics necessary to evaluate a new algorithm. An extensive comparative study with four other well-known methods is also conducted to validate and compare the performance of the LSA. The proposed LSA demonstrates satisfactory exploration, exploitation, and convergence characteristics. Furthermore, the LSA generally provides better results compared with the DSA, BSA, FFA, PSO and HSA with a high convergence rate. The projectile models can still be improved to optimize the performance of the LSA.

Conflict of interests

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

References

[1] K. Deb, A. Kumar, Real coded genetic algorithms with simulated binary crossover: studies on multi modal and multi objective problems, Complex Syst. 9 (1995) 431–454.
[2] C. Kanzow, N. Yamashita, T. Fukushima, Levenberg–Marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints, Comput. Appl. Math. 172 (2004) 375–397.
[3] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, New York, NY, 1995, pp. 39–43.
[4] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization. Computer Engineering Department, Engineering Faculty, Erciyes University, Tec. Rep.TR-06 (1-10), 2005, October.
[5] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 26 (1996) 29–41.
[6] M. Eslami, H. Shareef, M.R. Taha, M. Khajehzadeh, Adaptive particle swarm optimization for simultaneous design of UPFC damping controllers, Electr. Power Energy Syst. 57 (2014) 116–128.
[7] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (2008) 687–697.
[8] H. Li, S. Wang, M. Ji, An improved chaotic ant colony algorithm, in: Advances in Neural Networks – ISNN 2012, Springer, 2012, pp. 633–640.
[9] W. Gao, S. Liu, Improved artificial bee colony algorithm for global optimization, Inf. Process. Lett. 111 (2011) (2012) 871–882.
[10] Z.H. Zhan, J. Zhang, Y. Li, S.H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 39 (2009) 1362–1381.
[11] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 179 (2009) 2232–2248.
[12] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2001) 60–70.
[13] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (2008) 702–713.
[14] A. Ahrari, A.A. Atai, Grenade explosion method – a novel tool for optimization of multimodal functions, Appl. Soft Comput. 10 (2010) 1132–1140.
[15] D. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, New York, 1989.
[16] L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial Intelligence Through Simulated Evolution, Wiley, New York, 1966.
[17] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[18] J.R. Koza, Genetic Programming: On the Programming of Computers by Natural Selection, MIT Press, Cambridge, MA, 1992.
[19] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (2011) 4–31.
[20] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (1999) 82–102.
[21] W. Khatib, P. Fleming, The stud GA: a mini revolution? in: A. Eiben, T. Back, M. Schoenauer, H. Schwefel (Eds.), Parallel Problem Solving from Nature, Springer, New York, 1998.
[22] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems IEEE Trans. Evol. Comput. 10 (2006) 646–657.
[23] C. Igel, N. Hansen, S. Roth, Covariance matrix adaptation for multi-objective optimization, Evol. Comput. 15 (2007) 1–28.
[24] X.-S. Yang, A new metaheuristic bat-inspired algorithm, in: Nature inspired cooperative strategies for optimization (NICSO 2010), Springer, 2010, pp. 65–74.
[25] X.S. Yang, Bat algorithm: literature review and applications, Int. J. Bio-Inspired Comput. 5 (2013) 141–149.
[26] K. Khan, A. Nikov, A. Sahai, A fuzzy bat clustering method for ergonomic screening of office workplaces, Adv. Intell. Soft Comput. 101 (2011) 59–66.
[27] J.H. Lin, C.W. Chou, C.H. Yang, H.L. Tsai, A chaotic Levy flight bat algorithm for parameter estimation in nonlinear dynamic biological systems, J. Comput. Inf. Technol. 2 (2012) 56–63.
[28] J. Xie, Y.Q. Zhou, H. Chen, A novel bat algorithm based on deferential operator and Levy flights trajectory, Comput. Intell. Neurosci. 2013 (2013) 1–13.
[29] X.S. Yang, Firefly Algorithm in Engineering Optimization, John Wiley & Sons Inc., 2010.
[30] X.S. Yang, Firefly Algorithms for Multimodal Optimization. Stochastic Algorithms: Foundations and Applications, Springer, 2009, pp. 169–178.
[31] X.S. Yang, H. Xingshi, Firefly algorithm: recent advances and applications, Int. J. Swarm Intell. 1 (2013) 36–50.
[32] P. Civicioglu, Backtracking search optimization algorithm for numerical optimization problems, Appl. Math. Comput. 219 (2013) 8121–8144.
[33] A.H. Gandomia, A.H. Alavib, Krill herd: a new bio-inspired optimization algorithm, Commun. Nonlinear Sci. Numer. Simul. 17 (2012) 4831–4845.
[34] B. Mandal, P.K. Roy, S. Mandal, Economic load dispatch using krill herd algorithm, Int. J. Electr. Power Energy Syst. 57 (2014) 1–10.
[35] G.G. Wanga, A.H. Gandomib, A.H. Alavic, An effective krill herd algorithm with migration operator in biogeography-based optimization, Appl. Math. Model. 38 (2014) 2454–2462.
[36] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Comput.-Aided Des. 43 (2011) 303–315.
[37] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching–learning-based optimization: a novel optimization method for continuous non-linear large scale problems, Inf. Sci. 183 (2011) 1–15.
[38] R. Venkata, V. Patel, An improved teaching–learning-based optimization algorithm for solving unconstrained optimization problems, Sci. Iran 20 (2013) 710–720.
[39] A.K. Fard, T. Niknam, A. Khosravi, Multi-objective probabilistic distribution feeder reconfiguration considering wind power plants, Int. J. Electr. Power Energy Syst. 55 (2014) 680–691.
[40] E. Cuevas, M. Cienfuegos, D. Zaldívar, M. Pérez, A swarm optimization algorithm inspired in the behavior of the social-spider, Expert Syst. Appl. 40 (2013) 6374–6384.
[41] E. Cuevas, M. Cienfuegos, A new algorithm inspired in the behavior of the social-spider for constrained optimization, Expert Syst. Appl. 41 (2014) 412–425.
[42] P. Civicioglu, Transforming geocentric Cartesian coordinates to geodetic coordinates by using differential search algorithm, Comput. Geosci. 46 (2012) 229–247.
[43] X.S. Yang, S. Deb, Cuckoo search via Lévy flights, in: 2009 NaBIC 2009 World Congress on Nature & Biologically Inspired Computing, IEEE, 2009, pp. 210–214.
[44] R. Rajabioun, Cuckoo optimization algorithm, Appl. Soft Comput. 11 (2011) 5508–5518.
[45] H. Kahramanli, A modified cuckoo optimization algorithm for engineering optimization, Int. J. Future Comput. Commun. 1 (2012) 199–201.
[46] A.A. Dul'zon, V.V. Lopatin, M.D. Noskov, O.I. Pleshkov, Modeling the development of the stepped leader of a lightning discharge, Tech. Phys. 44 (1999) 394–398.
[47] I. Gallimberti, G. Bacchiega, A. Bondiou-Clergerie, P. Lalande, Fundamental processes in long air gap discharges, Appl. Phys. 3 (2002) 1–25.
[48] J.R. Dwyer, M.A. Uman, The physics of lightning, Phys. Rep. 534 (2014) 147–241.
[49] D. Nguyen, G. Deegan, F. D'Alessandro, Fractal nature of probabilistic model of lightning discharge, in: TENCON 2001 Proceedings of IEEE Region 10 International Conference on Electrical and Electronic Technology, IEEE, 2001, pp. 814–818.
[50] A. Berkopec, Fast particles as initiators of stepped leaders in CG and IC lightnings, J. Electrostat. 70 (2012) 462–467.
[51] H.R. Tizhoosh, Opposition-Based Learning: A New Scheme for Machine Intelligence, CIMCA/IAWTIC, 2005, pp. 695–701.

[52] D. Applegate, R. Bixby, V. Chvatal, W. Cook, The Traveling Salesman Problem: A Computational Study, Princeton University Press, 2011, ISBN 9780691129938.

Hussain Shareef received his B.Sc. with honors from IIT, Bangladesh, MS degree from METU, Turkey, and PhD degree from UTM, Malaysia, in 1999, 2002 and 2007, respectively. He is currently an Associate Professor at the Department of Electrical Engineering, United Arab Emirates University. His current research interests are power system deregulation, power quality, artificial intelligence and power system distribution automation. He is a member of IEEE.

Ahmad Asrul Ibrahim received his B.Eng. and M.S. degrees from Universiti Kebangsaan Malaysia, Malaysia in 2008 and 2012, respectively. He is currently a lecturer at Department of Electrical, Electronic and System Engineering, Universiti Kebangsaan Malaysia. His research interests include power quality assessment, artificial intelligence and power system automation.

Ammar Hussein Mutlag received his B.Sc. and M.S. degrees from University of Technology, Iraq in 2000 and 2005, respectively. He is currently pursuing the Ph.D. degree in electrical engineering at Universiti Kebangsaan Malaysia. His research interest is in the application of artificial intelligence to power converters.