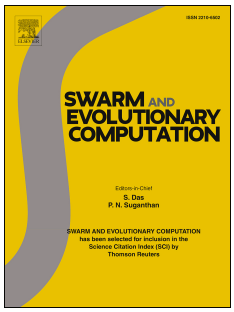
Accepted Manuscript



A novel nature-inspired algorithm for optimization: Squirrel search algorithm

Mohit Jain, Vijander Singh, Asha Rani

PII: S2210-6502(17)30522-9

DOI: 10.1016/j.swevo.2018.02.013

Reference: SWEVO 366

To appear in: *Swarm and Evolutionary Computation BASE DATA*

Received Date: 26 June 2017

Revised Date: 30 November 2017

Accepted Date: 23 February 2018

Please cite this article as: M. Jain, V. Singh, A. Rani, A novel nature-inspired algorithm for optimization: Squirrel search algorithm, *Swarm and Evolutionary Computation BASE DATA* (2018), doi: 10.1016/j.swevo.2018.02.013.

A novel nature-inspired algorithm for optimization: Squirrel search algorithm

Mohit Jain[a,*], Vijander Singh[a], Asha Rani[a]

[a]*Instrumentation and Control Engineering Division, Netaji Subhas Institute of Technology, University of Delhi, Delhi-110078, India*

Abstract

This paper presents a novel nature-inspired optimization paradigm, named as squirrel search algorithm (SSA). This optimizer imitates the dynamic foraging behaviour of southern flying squirrels and their efficient way of locomotion known as gliding. Gliding is an effective mechanism used by small mammals for travelling long distances. The present work mathematically models this behaviour to realize the process of optimization. The efficiency of the proposed SSA is evaluated using statistical analysis, convergence rate analysis, Wilcoxon's test and ANOVA on classical as well as modern CEC 2014 benchmark functions. An extensive comparative study is carried out to exhibit the effectiveness of SSA over other well-known optimizers in terms of optimization accuracy and convergence rate. The proposed algorithm is implemented on a real-time Heat Flow Experiment to check its applicability and robustness. The results demonstrate that SSA provides more accurate solutions with high convergence rate as compared to other existing optimizers.

Keywords: Nature-inspired algorithm, Unconstrained optimization, Squirrel search algorithm

1. Introduction

Optimization is the process of searching the best possible solution for a particular problem, under given circumstances. In recent years several real-world complex optimization problems have emerged in various fields of engineering [1, 2], business and economics [3] which cannot be solved within adequate time or accuracy by classical methods. There exist abundant mechanisms and principles in nature, which may be used to design computational intelligence methods so as to address such optimization problems. In the past few decades, researchers have developed several nature-inspired optimization algorithms, which imitate some biological behaviour or physical phenomena. For instance, Holand et al. [4] proposed genetic algorithm (GA) based on the concept of survival of fittest i.e. Darwinian theory of evolution. GA is one of the most renowned population based optimization technique. Kirkpatrick et al. [5] designed a single solution based optimization technique inspired from annealing phenomenon of metallurgy called as simulated annealing (SA). Particle swarm optimization (PSO) [6], ant colony optimization (ACO) [7] and artificial bee colony (ABC) [8] are other innovative distributed intelligent paradigms for solving the complex optimization problems inspired from the social behaviour of birds, colonizing species such as ants and honey bees respectively. These optimization methods are well appreciated by scientific community of computational intelligence, as they outperformed the classical heuristic methods, especially in case of multimodal, discrete and non-differential complex optimization problems. Moreover, these algorithms have been successfully employed in various fields of science such as process control [9], biomedical signal processing [10], image processing [11], flexible job shop scheduling [12] and many other engineering design problems [13, 14].

*Corresponding author

Email addresses: nsit.mohit@gmail.com (Mohit Jain), vijaydee@gmail.com (Vijander Singh), ashansit@gmail.com (Asha Rani)

Nature-inspired algorithms can be broadly classified into three main categories: evolutionary algorithms (EA), swarm intelligence (SI) and physics-based (PB) algorithms. EAs imitate the evolutionary behaviour of creatures found in nature. The search algorithms start with randomly generated solutions, generally termed as population, which further evolves over successive generations. Best individuals are combined to form new generation, which is the main strength of EAs as it promotes the improvement of population over the course of iterations. GA as well as differential evolution (DE) [15] algorithms can be considered as the most standard form of EAs. The second category is swarm intelligence based techniques, which mimic the intelligent social behaviour of groups of animals. Generally, SI based algorithms gather and utilize full information about search space with the progress of algorithm, while such information is abandoned by EAs from generation to generation. PSO, ACO and ABC can be described as representative algorithms in SIs. Some of the recent SIs are cuckoo search (CS) [16], grey wolf optimizer (GWO) [17], dragonfly algorithm (DA) [18] and many more [19]. The physics-based algorithms are inspired from basic physical laws that exist in universe. Some of the prevailing methods of this category are SA, gravitational search algorithm (GSA) [20], multi-verse optimizer (MVO) [21] and charged system search (CSS) [22]. A brief literature review on nature-inspired algorithms is presented in Table 1.

Table 1
Brief literature review on nature-inspired optimization algorithms

Algorithm	Inspiration	Year
Genetic Algorithm (GA) [4]	Evolution	1975
Simulated Annealing (SA) [5]	Annealing process in matallurgy	1983
Particle Swarm Optimization (PSO) [6]	Intelligent social behaviour of bird flock	1995
Artificial Fish-Swarm Algorithm (AFSA) [23]	Collective intelligence of fish swarm	2003
Termite Algorithm [24]	Termite colony	2006
Ant colony optimization (ACO) [7]	Ant colony	2006
Artificial bee colony (ABC) [8]	Honey Bee	2006
Imperialist competitive algorithm (ICA) [25]	Imperialistic competition	2007
Monkey search (MS) [26]	Monkey climbing process on trees while looking for food	2007
Group Search Optimizer (GSO) [27]	Animal searching (foraging) behaviour	2009
Firefly algorithm (FF) [28]	Social behaviour of fireflies	2009
Gravitational Search Algorithm (GSA) [20]	Law of gravity and mass interactions	2009
Bat algorithm (BA) [29]	Echolocation behaviour of bats	2010
Flower pollination algorithm (FPA) [30]	Pollination process of flowering species	2012
Fruit fly Optimization Algorithm (FFOA) [31]	Fruit foraging behaviour of fruit fly	2012
Krill Herd (KH) [32]	Herding behaviour of krill individuals in nature	2012
Mine blast algorithm (MBA) [33]	Mine bomb explosion	2013
Dolphin Echolocation (DE) [34]	Echolocation ability of dolphins	2013
Lightning search algorithm (LSA) [35]	Natural phenomenon of lightning	2015
Dragonfly algorithm (DA) [18]	Static and dynamic swarming behaviours of dragonflies	2015
Artificial algae algorithm (AAA) [36]	Living behaviours of microalgae	2015
Ant Lion Optimizer (ALO) [37]	Hunting mechanism of antlions in nature	2015
Shark Smell Optimization (SSO) [38]	Ability of shark in finding its prey by smell sense	2016
Dolphin Swarm Optimization Algorithm (DSOA) [39]	Mechanism of dolphins in detecting, chasing and preying on swarms of sardines	2016
Virus colony search [40]	Virus infection and diffusion strategies	2016
Whale Optimization Algorithm (WOA) [41]	Social behaviour of humpback whales	2016
Multi-Verse Optimizer (MVO) [21]	Multi-verse theory	2016
Crow search algorithm (CSA) [42]	Intelligent food hiding behaviour of crows	2016
Salp swarm algorithm [43]	Swarming behaviour of salps during navigating and foraging in oceans	2017
Grasshopper optimisation algorithm [44]	Swarming behaviour of grasshoppers	2017
Selfish herd optimizer (SHO) [45]	Hamilton's selfish herd theory	2017
Electro-Search algorithm [46]	Orbital movement of the electrons around the atomic nucleus	2017
Thermal exchange optimization [47]	Newton's law of cooling	2017
Mouth Brooding Fish algorithm [48]	Life cycle of mouth brooding fish	2017
Weighted Superposition Attraction (WSA) [49]	Superposition principle	2017
Spotted hyena optimizer [50]	Social behaviour of spotted hyenas	2017
Butterfly-inspired algorithm [51]	Mate searching mechanism of butterfly	2017
Lightning Attachment Procedure Optimization [52]	Lightning attachment process	2017

Apart from this, recently various modifications are also proposed in the basic versions of existing nature-inspired algorithms for solving complex optimization problems. As an illustration, slow convergence rate of ABC algorithm is improved in the variant IABC (improved ABC) [53] and tested on several reliability optimization problems. The same issue is resolved by incorporation of improved global best guiding mechanism in ABC [54] and improved exploitation capability is also achieved through adaptive limit mechanism. Higher accuracy with quick convergence characteristic is claimed by co-variance guided ABC [55] while solving portfolio optimization problem. Likewise, the performance of basic DE is improved for large scale optimization problems by embedding a simple switching mechanism for two control parameters of DE [56]. The issue of low convergence efficiency of basic cuckoo search algorithm is resolved by integrating chaos

mechanism and the resulting improved cuckoo search (ICS) is successfully applied to optimization problem of visible light communications (VLC) in smart homes [57]. Successful application of fuzzy logic in diversified fields of science has gained significant attraction of metaheuristic developers and hence various optimization algorithms are improved by utilizing the advantages of fuzzy logic principles. In this context, some of the relevant developments are: fuzzy harmony search algorithm [58], fuzzy imperialist competitive algorithm with dynamic parameter adaptation [59], fuzzy based water cycle algorithm [60], hierarchical GWO algorithm [61], interval type-2 fuzzy logic based bat algorithm [62] and type-2 fuzzy logic based ant colony optimization algorithm [63] etc. In a similar fashion, principles of quantum computing are incorporated in metaheuristics design and improved performance is claimed through various studies. Quantum inspired binary grey wolf optimizer [64], hybrid quantum-inspired genetic algorithm (HQIGA) [65] and quantum inspired particle swarm optimization (QPSO) [66] are some examples of research in this area. All nature-inspired algorithms possess some common characteristics like: (i) they imitate some natural phenomenon (ii) they do not demand gradient information (iii) Employ random variables (iv) and contain various parameters which must be defined adequately to solve a problem [40]. Each algorithm offers distinctive benefits, from the perspective of robustness, performance in the presence of uncertainty and unknown search spaces [40].

The advancement in technology also leads to several complex optimization problems. As an illustration, increased usage of social networking websites, huge data volumes are generated every second, which presents a new optimization problem of effective handling of user generated big data [67]. Another crucial optimization problem is time-dependent pollution-routing problem [68], which is generated due to implementation of new environmental legislations for cities having problem of congestion. Pollution level increases with congestion due to increased emission of greenhouse gases by commercial vehicles of freight companies. In spite of the existence of many prominent optimization algorithms in literature, scientific community is still developing new optimization techniques for solving new and more complex optimization problems under the ideology of continuous improvement in order to achieve better design. Moreover, in accordance with "no free lunch" (NFL) theorem, there is no single nature-inspired optimization technique, which can optimally solve all optimization problems [69]. This means that an optimization algorithm is competent for solving a certain set of problems but ineffective on other class of problems [70]. The NFL theorem, certainly, keeps this domain of research open and allows the researchers to improve the existing algorithms or propose new algorithms for better optimization. Hence, the present study proposes a new simple and powerful nature-inspired algorithm called squirrel search algorithm (SSA) for unconstrained numerical optimization problems. This algorithm simulates the dynamic foraging strategy of southern flying squirrels and their efficient way of locomotion known as gliding. The main contributions of the proposed work are as follows:

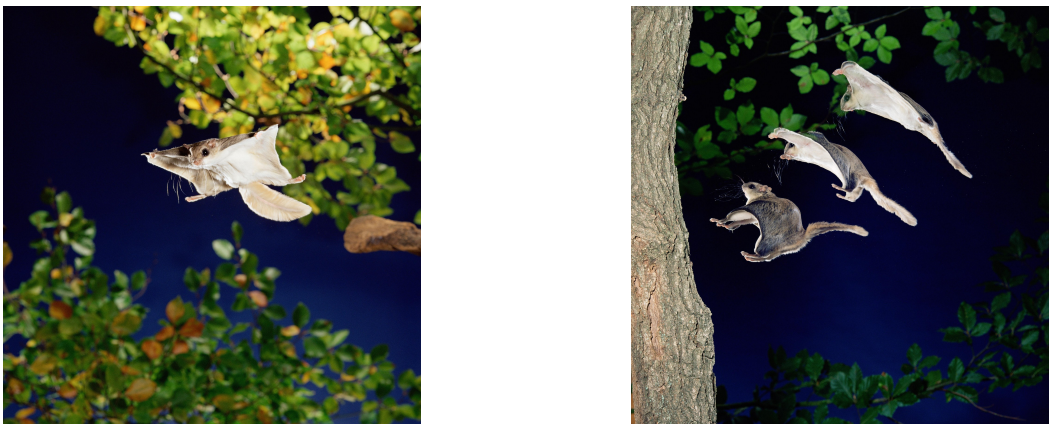
1. A novel nature-inspired squirrel search optimization algorithm is proposed. The foraging behaviour of flying squirrels is studied thoroughly and modeled mathematically including each and every feature of their food search.
2. The proposed algorithm is validated on 33 classical as well as modern CEC 2014 benchmark functions.
3. Rigorous comparative study is performed with existing nature-inspired optimization algorithms using statistical analysis, convergence rate analysis, Wilcoxon's test and ANOVA.
4. Robustness and effectiveness of SSA as well as other optimizers are investigated for optimization of two degree of freedom proportional and integral (2DOFPI) controller for precise temperature control of heat flow experiment (HFE).

The rest of this paper is organized as follows: Section 2 discusses the motivation of proposed algorithm. In Section 3 basic concepts of SSA and its assumptions are discussed. Section 4 presents the details about implementation of SSA. Section 5 presents the comparison of SSA with existing optimizers. Section 6 provides the comparative statistical analysis of results on standard benchmark functions. Section 7 provides the real-time application of the proposed algorithm. Finally the work is concluded in Section 8.

2. Inspiration

Flying squirrels are a diversified group of arboreal and nocturnal type of rodents that are exceptionally adapted for gliding locomotion. Currently, 15 genera and 44 species of flying squirrels are identified and

majority of them are found in deciduous forest area of Europe and Asia, particularly South-eastern Asia. The only species found outside Eurasia and most studied is Glaucomys volans known as southern flying squirrel [71]. The flying squirrels are considered to be the most aerodynamically sophisticated having a parachute-like membrane (patagia), which helps the squirrel in gliding from one tree to the other and makes them capable in modifying lift and drag [72]. The most interesting fact about flying squirrels is that they do not fly, instead they use a special method of locomotion i.e. "Gliding" which is considered to be energetically cheap, allowing small mammals to cover large distances quickly and efficiently [72]. Literature suggests that predator avoidance, optimal foraging and cost of foraging are the primary cause of evolution of gliding [73]. Fig. 1a shows the real image of flying squirrel while gliding and Fig. 1b shows the slow motion sequences of flying squirrel before landing on a tree. The squirrels can optimally use food resources by showing a dynamic



(a) Just after gliding **(b)** Landing on a tree

Fig. 1. Real flying squirrel [74, 75]

foraging behaviour [76, 77]. For instance, to meet nutritional requirements in autumn, they prefer to eat acorns (a mast nut) as they are available in abundance while store other nuts such as hickories in nests, other cavities, and sometimes the ground. During winters when nutritional demands are higher due to low temperature, hickory nuts are eaten promptly at the site of discovery during foraging and are also taken out from reserve food stores. Therefore, selectively eating some nuts and storing others depending upon the nutritional demands, allows optimum utilization of both types of available mast nuts [77]. This intelligent dynamic foraging behaviour of southern flying squirrel is the main source of motivation for proposed SSA. In this work dynamic foraging strategy and gliding mechanism of flying squirrels are modelled mathematically to design SSA for optimization.

3. Squirrel Search Algorithm (SSA)

The search process begins when flying squirrels start foraging. During warm weather (autumn) the squirrels search for food resources by gliding from one tree to the other. While doing so, they change their location and explore different areas of forest. As the climatic conditions are hot enough, they can meet their daily energy needs more quickly on the diet of acorns available in abundance and hence they consume acorns immediately upon finding them. After fulfilling their daily energy requirement, they start searching for optimal food source for winter (hickory nuts). Storage of hickory nuts will help them in maintaining their energy requirements in extremely harsh weather and reduce the costly foraging trips and therefore increase the probability of survival. During winter, a loss of leaf cover in deciduous forests results an increased risk of predation and hence they become less active but do not hibernate in winter. At the end of winter season,

flying squirrels again become active. This is a repetitive process and continues till the lifespan of a flying squirrel and forms the foundation of SSA. The following assumptions are considered for simplification of mathematical model:

1. There is n number of flying squirrels in a deciduous forest and one squirrel is assumed to be on one tree.
2. Every flying squirrel individually searches for food and optimally utilizes the available food resources by exhibiting a dynamic foraging behaviour.
3. In forest, only three types of trees are available such as normal tree, oak tree (acorn nuts food source) and hickory tree (hickory nuts food source).
4. The forest region under consideration is assumed to contain three oak trees and one hickory tree.

In the present study, number of squirrels, n is considered to be 50. 4 nutritious food resources (N_{fs}) are considered with 1 hickory nut tree and 3 acorn nut trees, whereas 46 trees have no food source. That is 92% of the total population of squirrels is on normal trees, while the remaining is on food sources. However, the number of food resources can be varied as per the constraint $1 < N_{fs} < n$ where $N_{fs} \in \mathbb{Z}_{>0}$ with one optimal winter food source.

4. Implementation of SSA

SSA starts with random initial location of flying squirrels similar to other population based algorithms. The location of a flying squirrel is represented by a vector, in d dimensional search space. Hence, the flying squirrels can glide in 1-D, 2-D, 3-D or hyper dimensional search space and change their location vectors.

4.1. Random initialization

There is n number of flying squirrels (FS) in a forest and location of i^{th} flying squirrel can be specified by a vector. The location of all flying squirrels can be represented by the following matrix:

$$FS = \begin{bmatrix} FS_{1,1} & FS_{1,2} & \cdots & \cdots & FS_{1,d} \\ FS_{2,1} & FS_{2,2} & \cdots & \cdots & FS_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ FS_{n,1} & FS_{n,2} & \cdots & \cdots & FS_{n,d} \end{bmatrix} \tag{1}$$

where $FS_{i,j}$ represents the j^{th} dimension of i^{th} flying squirrel. A uniform distribution (Eq. (2)) is used to allocate the initial location of each flying squirrel in the forest.

$$FS_i = FS_L + U(0,1) \times (FS_U - FS_L) \tag{2}$$

where FS_L and FS_U are lower and upper bounds respectively of i^{th} flying squirrel in j^{th} dimension and $U(0,1)$ is a uniformly distributed random number in the range $[0,1]$.

4.2. Fitness evaluation

The fitness of location for each flying squirrel is calculated by putting the values of decision variable (solution vector) into a user defined fitness function and the corresponding values are stored in the following array:

$$f = \begin{bmatrix} f_1\left([FS_{1,1}, FS_{1,2}, \ldots, FS_{1,d}]\right) \\ f_2\left([FS_{2,1}, FS_{2,2}, \ldots, FS_{2,d}]\right) \\ \vdots \\ \vdots \\ f_n\left([FS_{n,1}, FS_{n,2}, \ldots, FS_{n,d}]\right) \end{bmatrix} \tag{3}$$

The fitness value of each flying squirrel's location depicts the quality of food source searched by it i.e. optimal food source (hickory tree), normal food source (acorn tree) and no food source (flying squirrel is on normal tree) and hence their probability of survival also.

4.3. Sorting, declaration and random selection

After storing the fitness values of each flying squirrel's location, the array is sorted in ascending order. The flying squirrel with minimal fitness value is declared on the hickory nut tree. The next three best flying squirrels are considered to be on the acorn nuts trees and they are assumed to move towards hickory nut tree. The remaining flying squirrels are supposed to be on normal trees. Further through random selection, some squirrels are considered to move towards hickory nut tree assuming that they have fulfilled their daily energy requirements. The remaining squirrels will proceed to acorn nut trees (to meet their daily energy need). This foraging behaviour of flying squirrel is always affected by the presence of predators. This natural behaviour is modelled by employing the location updating mechanism with predator presence probability (P_{dp}).

4.4. Generate new locations

As discussed previously, three situations may occur during the dynamic foraging of flying squirrels. In each situation it is assumed that in the absence of predator, flying squirrel glides and searches efficiently throughout the forest for its favourite food, while presence of predator makes it cautious and is forced to use small random walk to search a nearby hiding location. The dynamic foraging behaviour can be mathematically modelled as follows:

Case 1: Flying squirrels which are on acorn nut trees (FS_{at}) may move towards hickory nut tree. In this case, the new location of squirrels can be obtained as follows:

$$FS_{at}^{t+1} = \begin{cases} FS_{at}^{t} + d_g \times G_c \times (FS_{ht}^{t} - FS_{at}^{t}) & R_1 \geq P_{dp} \quad (4a) \\ \text{Random location} & \text{otherwise} \quad (4b) \end{cases}$$

where d_g is random gliding distance, R_1 is a random number in the range of $[0, 1]$, FS_{ht} is the location of flying squirrel that reached hickory nut tree and t denotes the current iteration. The balance between exploration and exploitation is achieved with the help of gliding constant G_c in the mathematical model. Its value significantly affects the performance of proposed algorithm. In the present work value of G_c is considered as 1.9, which is obtained after rigorous analysis.

Case 2: Flying squirrels on normal trees (FS_{nt}) may move towards acorn nut trees to fulfill their daily energy needs. In this case, new location of squirrels can be obtained as follows:

$$FS_{nt}^{t+1} = \begin{cases} FS_{nt}^{t} + d_g \times G_c \times (FS_{at}^{t} - FS_{nt}^{t}) & R_2 \geq P_{dp} \quad (5a) \\ \text{Random location} & \text{otherwise} \quad (5b) \end{cases}$$

where R_2 is a random number in the range $[0, 1]$.

Case 3: Some squirrels which are on normal trees and already consumed acorn nuts may move towards hickory nut tree in order to store hickory nuts which can be consumed at the time of food scarcity. In this case, new location of squirrels can be obtained as follows:

$$FS_{nt}^{t+1} = \begin{cases} FS_{nt}^{t} + d_g \times G_c \times (FS_{ht}^{t} - FS_{nt}^{t}) & R_3 \geq P_{dp} \quad (6a) \\ \text{Random location} & \text{otherwise} \quad (6b) \end{cases}$$

where R_3 is a random number in the range $[0, 1]$. Predator presence probability P_{dp} is considered to be 0.1 in all cases for the present work.

Fig. 2 shows the conceptual model of gliding locomotion used by flying squirrels for their effective movement while foraging in night. A flying squirrel glides by modifying the lift and drag forces [72].

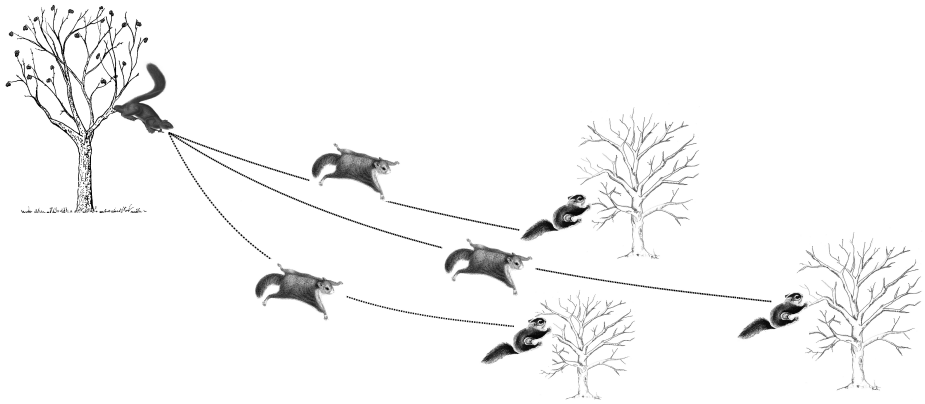


Fig. 2. Conceptual model of flying squirrel moving from one tree to another using gliding locomotion

4.5. Aerodynamics of gliding

Gliding mechanism of flying squirrels is described by equilibrium glide in which sum of lift (L) and drag (D) force produces a resultant force (R) whose magnitude is equal and opposite to the direction of flying squirrel's weight (Mg). Thus, R provides a linear gliding path (Fig. 3a) to flying squirrel at constant velocity (V) [72, 78]. In this work an approximated model of gliding behaviour is (Fig. 3b) utilized in the design of optimization algorithm. A flying squirrel gliding at steady speed always descends at an angle ϕ to horizontal and lift-to-drag ratio or glide ratio, defined as follows [79]:

$$L/D = 1/\tan\phi \tag{7}$$

The flying squirrels can increase their glide-path length by making smaller glide angle (ϕ) and thus lift-to drag ratio is increased. Here, the lift results from downward deflection of air passing over the wings and is defined as:

$$L = 1/2\rho C_L V^2 S \tag{8}$$

where ρ (=1.204 kgm^{-3}) is density of air, C_L is called as lift coefficient, V (=5.25 ms^{-1}) is speed and S (=154 cm^2) is the surface area of body [79]. The frictional drag is defined as:

$$D = 1/2\rho V^2 S C_D \tag{9}$$

where C_D is the frictional drag coefficient. At slow speed, this drag component is very high while at high speed it becomes smaller. Hence from Eq. (7) glide angle at steady state is determined as:

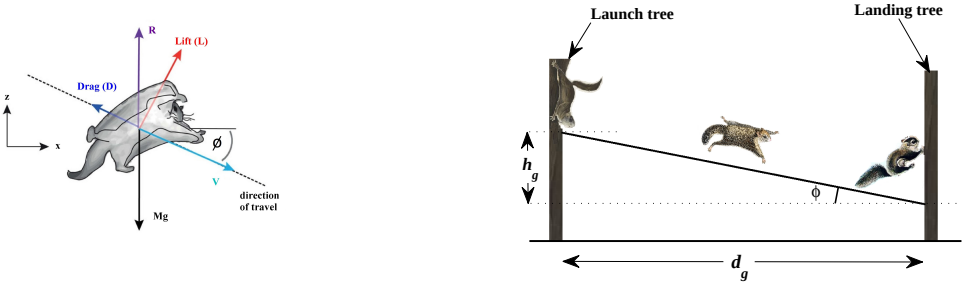
$$\phi = \arctan\left(\frac{D}{L}\right) \tag{10}$$

The approximated gliding distance (d_g) is calculated (Fig. 3b) as follows:

$$d_g = \left(\frac{h_g}{\tan\phi}\right) \tag{11}$$

where h_g (=8m) is the loss in height occurred after gliding. All the parametric values including C_L and C_D, required to compute d_g are considered from the real data [72, 78, 80]. Thus a flying squirrel may vary its glide-path length or d_g by simply changing the lift-to-drag ratio as per the desired landing location. The simulations are performed by incorporating random variations in C_L in the range $0.675 \leq C_L \leq 1.5$ and C_D is considered to be fixed at 0.60.

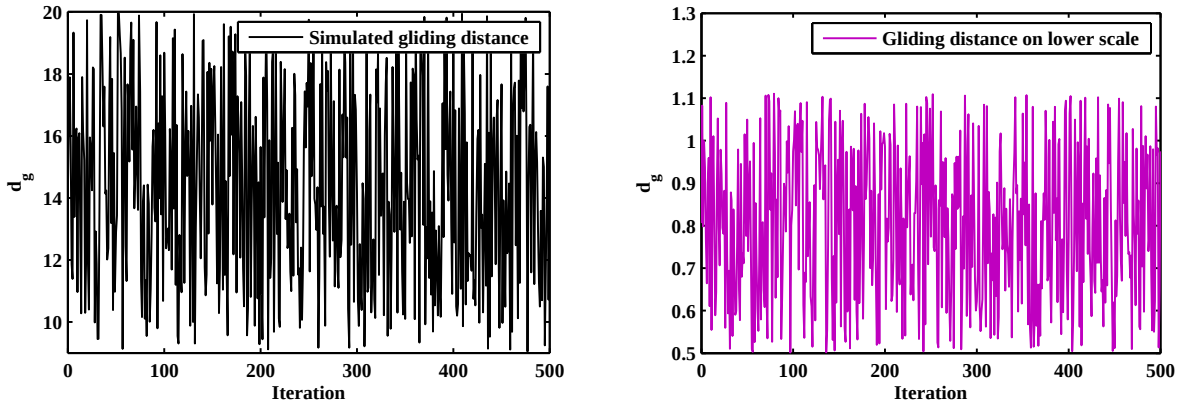
Flying squirrels generally travel a horizontal gliding distance ranging from 5 to 25m in a single glide [72]. Gliding distance in the proposed model is considered to be in the range of 9 to 20m which is validated



(a) Flying squirrel gliding at equilibrium **(b)** Approximated model of gliding behaviour

Fig. 3

through Fig. 4a. The value of d_g is quite large and may introduce large perturbations in Eq. (4a), Eq. (5a) and Eq. (6a), which may cause unsatisfactory performance of the algorithm. The value of d_g is scaled down to achieve acceptable performance of the algorithm. d_g is divided by a suitable non-zero value called as scaling factor (sf) obtained through rigorous experimentation on benchmark functions. Table 2 presents some recorded results of experimentation. It is observed during experimentation that value of scaling factor sf may be varied from 16 to 37 in order to achieve the desired level of accuracy without affecting stability of algorithm. However, in the present work, sf=18 provides sufficient perturbation range of d_g in the interval $[0.5, 1.11]$ (Fig. 4b) and satisfactory performance is achieved for most of the benchmark functions. Thus sf helps to achieve the desired balance between exploration and exploitation phases, which is a mandatory requirement for designing an efficient metaheuristic.



(a) Simulated gliding distance **(b)** Scaling down the gliding distance using sf=18

Fig. 4

4.6. Seasonal monitoring condition

Seasonal changes significantly affect the foraging activity of flying squirrels [81]. They suffer significant heat loss at low temperatures, as they posses high body temperature and small size which makes foraging cost high as well as risky due to the presence of active predators. Climatic conditions force them to be less active in winters as compared to autumn [77]. Thus movement of flying squirrels is affected by weather changes and inclusion of such behaviour may provide a more realistic approach towards optimization. Therefore

Table 2
The effect of scaling factor (sf) on the performance of SSA on four benchmark functions*

Function	Parameter	sf=10	sf=15	sf=18	sf=30	sf=40	sf=50	sf=100
TF1	Mean	2.2000E+01	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	3.3333E-02
	SD	6.6664E+01	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	1.8257E-01
TF5	Mean	1.0522E-02	2.1527E-11	3.9454E-22	1.8899E-25	2.9617E-07	2.3469E-06	1.4112E-05
	SD	8.5168E-03	7.1976E-11	2.0493E-21	8.4211E-25	1.0297E-06	5.5454E-06	2.7179E-05
TF13	Mean	2.0433E+00	2.4120E-10	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	2.9231E-07
	SD	2.0746E+00	5.3536E-10	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	9.3018E-07
TF23	Mean	-182.1364	-186.73	-186.73	-186.73	-186.73	-186.73	-186.73
	SD	4.3904E+00	1.1788E-07	1.9029E-14	2.2392E-14	2.4831E-11	1.2022E-05	2.5781E-04

*See Tables 4, 6, 8 and 10 for benchmark functions

a seasonal monitoring condition is introduced in SSA which prevents the proposed algorithm from being trapped in local optimal solutions. Following steps are involved in modelling the behaviour:

a. First calculate the seasonal constant (S_c) using Eq. (12)

$$S_c^t = \sqrt{\sum_{k=1}^{d} (FS_{at,k}^t - FS_{ht,k})^2} \tag{12}$$

where $t = 1, 2, 3$.

b. Check the seasonal monitoring condition i.e. $S_c^t < S_{min}$ where S_{min} is the minimum value of seasonal constant computed as:

$$S_{min} = \frac{10E^{-6}}{(365)^{t/(t_m/2.5)}} \tag{13}$$

where t and t_m are the current and maximum iteration values respectively. The value S_{min} affects the exploration and exploitation capabilities of the proposed method. Larger value of S_{min} promotes exploration while smaller values of S_{min} enhance the exploitation capability of algorithm. For any effective metaheuristic, there must be a proper balance between these two phases [82]. Although, this balance is maintained by gliding constant G_c (Eq. (4a), Eq. (5a) and Eq. (6a)), but it may be improved by adaptively changing the value of S_{min} during the course of iterations.

c. If seasonal monitoring condition is found true (i.e. winter season is over), randomly relocate those flying squirrels which could not explore the forest for optimal winter food source.

4.7. Random relocation at the end of winter season

As discussed previously, the end of winter season makes flying squirrels active due to low foraging cost. The flying squirrels which could not explore the forest for optimal food source in winter and still survived may forage in new directions. The incorporation of this behaviour in modelling may enhance the exploration capability of proposed algorithm. It is assumed that, only those squirrels which could not search the hickory nuts food source and still survived will move to different directions in order to find better food source. The relocation of such flying squirrels is modelled through the following equation:

$$FS_{nt}^{new} = FS_L + \text{Lévy(n)} \times (FS_U - FS_L) \tag{14}$$

where Lévy distribution encourages better and efficient search space exploration. Lévy flight is a powerful mathematical tool used by the researchers for improving global exploration capability of various metaheuristic algorithms [16, 83–87]. Lévy flights help to find new candidate solutions far away from the current best solution. It is a kind of random walk in which step length is drawn from a Lévy distribution. This distribution is often expressed by a power-law formula $L(s) \sim |s|^{-1-\beta}$ where $0 < \beta \leq 2$ is an index. Lévy distribution is stated mathematically as follows:

$$L(s, \gamma, \mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}}\, exp\left[-\frac{\gamma}{2(s-\mu)}\right] \frac{1}{(s-\mu)^{3/2}} & 0 < \mu < s < \infty \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where $\mu, \gamma > 0$. γ is scale parameter and μ is shift parameter. The Lévy flight is calculated as follows:

$$\text{Lévy}(x) = 0.01 \times \frac{r_a \times \sigma}{|r_b|^{\frac{1}{\beta}}} \tag{16}$$

where r_a and r_b are two normally distributed random numbers in $[0, 1]$, β is a constant considered to be 1.5 in the present work, and σ is calculated as:

$$\sigma = \left(\frac{\Gamma(1+\beta) \times sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{1/\beta} \tag{17}$$

where $\Gamma(x) = (x-1)!$.

4.8. Stopping criterion

Function tolerance is a commonly used convergence criterion in which a permissible but small threshold value is defined between the last two consecutive results. Sometimes maximum execution time is also used as stopping criterion. In the present study maximum number of iterations is considered as stopping criterion. The pseudocode of SSA is provided in Algorithm 1.

4.9. Internal dynamics of SSA

The internal dynamics of SSA is illustrated in Fig. 5. In case 1, when $R_1 > P_{dp}$ the location of a flying squirrel which is on acorn tree (FS^t_{at}) may be represented by a vector with an assumed direction shown in Fig. 5a and randomly scaled difference vector ($d_g \times G_c \times (FS^t_{ht} - FS^t_{at})$) may be directed towards the location of flying squirrel which is on hickory nut tree (FS^t_{ht}). The resultant vector may attain a new search direction towards and near to global optimum point. If $R_1 < P_{dp}$, i.e. predator presence is felt by a flying squirrel, then it may glide in a random direction or may move to a nearby safe location. Few random locations are shown through vectors in Fig. 5b. In the terminology of optimization algorithms, this random orientation of location vectors, may enhance the exploration phase of proposed algorithm. Likewise, the other cases of proposed technique illustrated in Fig. 5c-Fig. 5f may be analyzed. It is thus inferred that proposed technique may explore the complete search space quite efficiently.

It is observed in many studies that "new" algorithms lack rigorous theoretical background and most of these algorithms are basically previous algorithms under new clothes and interpretation [88]. Hence, criterion for releasing a new algorithm becomes much higher. There must be significant differences in the new algorithm from the others in order to release a new algorithm [89]. Therefore in the present study SSA is compared on conceptual grounds with other similar metaheuristics.

5. Conceptual comparative analysis of SSA with other metaheuristic algorithms

Broadly, all nature-inspired metaheuristics imitate two distinct features of nature i.e. adaptability and choice of the fittest, which gives them a similar appearance superficially. Most of the algorithms utilize the concept of pattern matrix, which is constructed through randomly generated solutions of optimization problem under consideration [90]. The pattern matrix is recursively updated at each iteration through a suitable updating mechanism. This mechanism injects new attributes or patterns in the pattern matrix while maintaining diversity in solutions. Metaheuristic algorithms are generally differentiated on the basis of their solution updating strategy. In this work, SSA is compared with particle swarm optimization (PSO), artificial bee colony (ABC), bat algorithm (BA) and firefly algorithm (FF).

Algorithm 1 Pseudocode for SSA

Begin:
Define input parameters
Generate random locations for n number of flying squirrels using Eq. (2)
Evaluate fitness of each flying squirrel's location
Sort the locations of flying squirrels in ascending order depending upon their fitness value
Declare the flying squirrels on hickory nut tree, acorn nuts trees and normal trees
Randomly select some flying squirrels which are on normal trees to move towards hickory nut tree and the remaining will move towards acorn nuts trees

while*(the stopping criterion is not satisfied)*

For*t=1 to n1 (n1=total flying squirrels which are on acorn trees and moving towards hickory nut tree)*

if $R_1 \geq P_{dp}$
$FS_{at}^{t+1} = FS_{at}^{t} + d_g \times G_c \times (FS_{ht}^{t} - FS_{at}^{t})$
else
FS_{at}^{t+1}=a random position of search space
end

end

For*t=1 to n2 (n2=total flying squirrels which are on normal trees and moving towards acorn trees)*

if $R_2 \geq P_{dp}$
$FS_{nt}^{t+1} = FS_{nt}^{t} + d_g \times G_c \times (FS_{at}^{t} - FS_{nt}^{t})$
else
FS_{nt}^{t+1}=a random position of search space
end

end

For*t=1 to n3 (n3=total flying squirrels which are on normal trees and moving towards hickory nut tree)*

if $R_3 \geq P_{dp}$
$FS_{nt}^{t+1} = FS_{nt}^{t} + d_g \times G_c \times (FS_{ht}^{t} - FS_{nt}^{t})$
else
FS_{nt}^{t+1}=a random position of search space
end

end

Calculate seasonal constant (S_c)

if *(Seasonal monitoring condition is satisfied)*
Randomly relocate flying squirrels using Eq. (14)

end

Update the minimum value of seasonal constant (S_{min}) using Eq. (13)

end

The location of squirrel on hickory nut tree is the final optimal solution

End

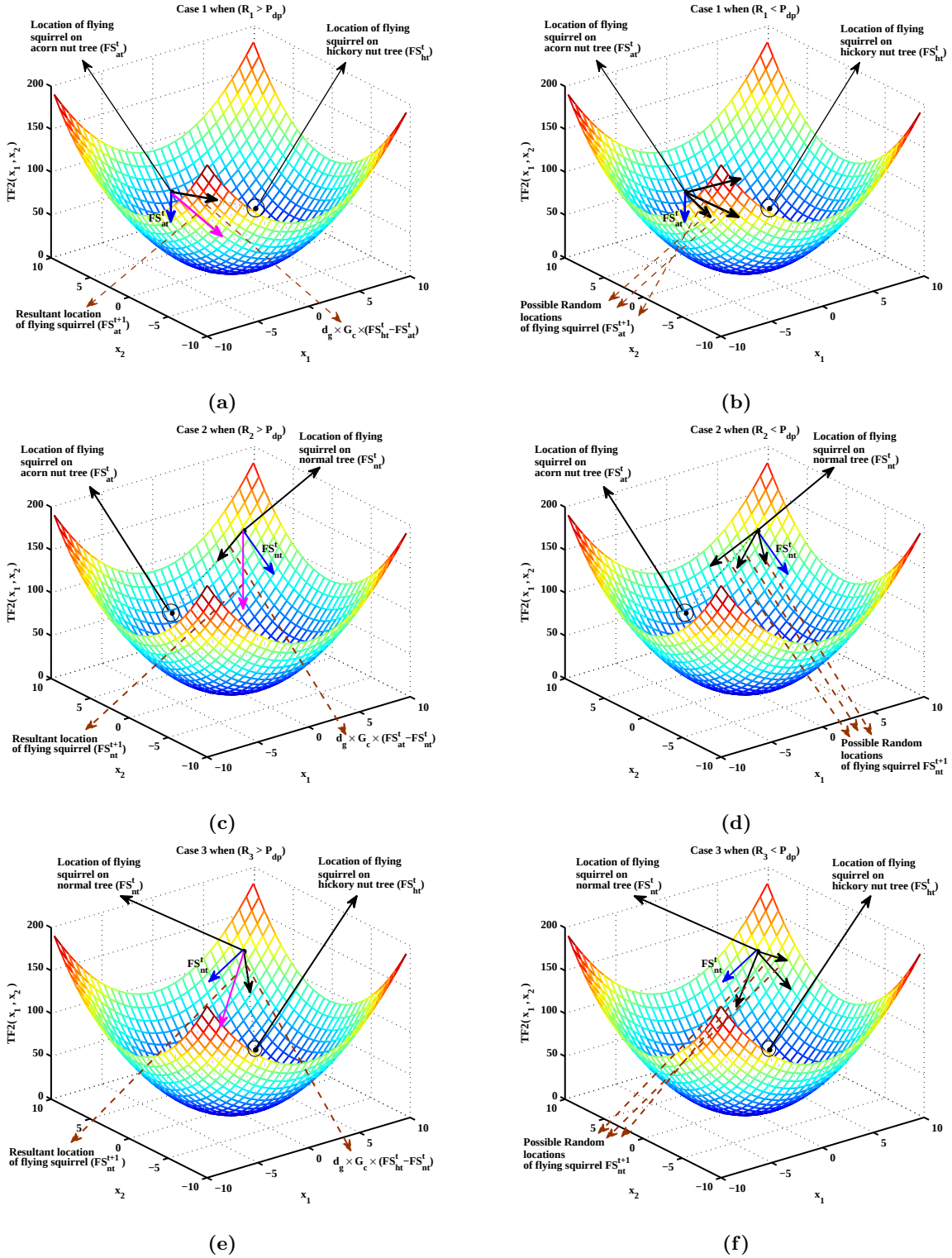


Fig. 5. Internal dynamics of SSA

5.1. Particle swarm optimization algorithm

PSO imitates the social behaviour of flock of birds. It starts optimization using randomly generated solutions commonly known as artificial particles. Each particle in swarm has an associated randomly generated velocity. If X_i is the initial position of i^{th} particle in swarm with velocity V_i then position updating mechanism of PSO can be defined as follows [91]:

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{18}$$

$$V_i(t+1) = wV_i(t) + C_1 r_{i1}(Pbest_i - X_i(t)) + C_2 r_{i2}(Gbest - X_i(t)) \tag{19}$$

where w is the inertial weight, C_1 and C_2 are cognitive and social constants respectively [91, 92]. The r_{i1} and r_{i2} are uniformly distributed random numbers in the interval $[0, 1]$ [93], $Pbest_i$ is the best previous position (local best solution) of i^{th} particle and $Gbest$ is the best position among all the particles (global best solution).

5.1.1. SSA versus PSO

SSA also initiates optimization process similar to PSO by movement of search agents in the search space, however the movement mechanism is entirely different. Some of the major differences are described as follows:

1. In PSO, new direction for movement of i^{th} particle is obtained by $Pbest_i$ and $Gbest$ i.e. cumulative effect of both is considered. However, SSA uses sorted information and divides the pattern matrix initially into three regions like global optimum solution (FS^t_{ht}), near optimal solutions (FS^t_{at}) and random solutions (FS^t_{nt}). The random solutions (FS^t_{nt}) are further randomly bifurcated to redirect the search towards globally optimum solution (FS^t_{ht}) and near optimal solutions (FS^t_{at}). Thus new patterns are injected in three phases. In the first phase, new directions for movement of search agents (FS^t_{at}) close to optimal solutions are obtained using the globally best solution (FS^t_{ht}) (Eq. (4a)). In the second phase, one part of randomly selected search agents (FS^t_{nt}) is promoted to move towards near optimal solutions (FS^t_{at}) (Eq. (5a)). In the last phase remaining search agents (FS^t_{nt}) are moved towards global optimum solution (FS^t_{ht}) (Eq. (6a)). In other words, PSO updates all solutions in the pattern matrix by single strategy, however SSA employs three strategies in different regions of pattern matrix. Thus position updating mechanism of SSA differs from PSO and its variants i.e. Cognitive only PSO and Social only PSO [91].
2. In PSO, the random numbers (r_{i1} and r_{i2}) are obtained from uniform distribution in the interval $[0, 1]$, while SSA uses behaviourally inspired random variations in gliding distance (d_g) in the interval $[0.5, 1.11]$.
3. Flying squirrel movement is affected by predator presence, which is modelled using a probabilistic behaviour. The inclusion of predator presence probability, suddenly redirects the location of any flying squirrel and hence improves the exploration capability of algorithm (Eq. (4b), Eq. (5b) and Eq. (6b)), whereas PSO does not utilize probabilistic phenomenon.
4. The simulation of flying squirrel behaviour provides an opportunity to introduce a seasonal condition in SSA. This invokes the algorithm several times to initiate search in different directions and thus the algorithm does not stuck in local optimal solutions. This feature is not present in PSO due to the natural behaviour of swarm.

5.2. Artificial Bee Colony algorithm

ABC algorithm mimics the foraging process of honey bees. A honey bee colony is composed of three kinds of bees:

- Employed bee: Each employed bee is connected to a nectar rich food source and strives to search new enriched food sources of nectar in the neighbourhood of its present food source. It memorizes the location of new food source, only if it finds that amount of nectar is more than its associated present food source. An employed bee, also shares this information with onlooker bees near the dancing region of hive, through a special dance.

- Onlooker bee: Onlooker bees judge the information received by employed bees and move towards new food sources, due to which the probability of finding a better food source also increases.
- Scout bee: An employed bee is converted into scout bee, if its associated food source is fully exploited and it searches new enrich food source randomly.

In the basic ABC algorithm, food sources are considered as solutions for an optimization problem. The amount of nectar present in a food source directly indicates the quality of food source and hence the quality of solution. It is also considered that 50% of the total population consists of employed bees (or food sources) and remaining 50% are onlooker bees. The location of i^{th} food source (or employed bee) is represented by a vector $X_i = (X_{i1}, X_{i2}, \ldots, X_{id})$ and all food sources (SN) can be randomly located initially by Eq. (20) [94]:

$$X_{ij} = X_j^{min} + rand_j(0,1)(X_j^{max} - X_j^{min}) \tag{20}$$

where $i = 1, ..., SN$, $j = 1, ..., d$ where d is the problem dimension, X_j^{min} and X_j^{max} are the minimum and maximum values of j^{th} dimension of problem respectively. The movement of employed bees towards new food sources is modelled through Eq. (21):

$$V_{ij} = X_{ij} + \phi_{ij}(X_{ij} - X_{kj}) \tag{21}$$

where $k \in (1, 2, \ldots, SN)$ and j are randomly selected indices provided $k \neq i$. The ϕ_{ij} is a uniformly distributed random number within $[-1, 1]$. The movement of onlooker bees is also decided as per Eq. (21) but utilizes probabilistic information calculated from roulette wheel method [94]. In the employed and onlooker bee phases, every bee tries to discover a more qualified food source until a predefined number of runs named as "*Limit*" is exceeded [95]. If the quality of solution (fitness value) is not elevated, till the predefined *Limit*, the corresponding bee is declared as scout bee and its location is randomly reinitialized using Eq. (20). This process continues until the stopping criterion is satisfied.

5.2.1. SSA versus ABC

ABC and SSA apparently looks quite similar, however technically both present several differences in their formulation and updating mechanism.

1. Both ABC and SSA work on the effective division of labour i.e. pattern matrix is divided into various regions. In ABC, half of the population belongs to employed bees and the remaining half is treated as onlooker bees. In contrary, SSA initially sorts the pattern matrix in ascending order of fitness and then divides it, which is controlled by the user. In the present work, first 8% of population belongs to flying squirrels on food sources and remaining population of squirrels is considered on normal trees. However, this percentage may vary and depends upon the available food sources and hence it is modelled as a user defined variable.
2. In ABC, a new solution is generated by probabilistic neighbourhood selection. The updating mechanism of ABC (Eq. (21)) basically replaces one randomly selected component of i^{th} solution vector by an arithmetic recombination of the component and corresponding component from another solution vector. This mechanism shows a conceptual resemblance with binomial crossover of DE algorithm [95]. On the other hand, the updating strategy of SSA is a kind of directed search approach i.e. the new solutions are forced to move towards best solutions and therefore predefined neighbourhood selection is considered. The best solution(FS_{ht}^t) and sorted series of locally best solutions (FS_{at}^t) are considered (Eq. (4a), Eq. (5a) and Eq. (6a)) while altering the solution vector. Apart from this, SSA also uses a probabilistic random location updating strategy which randomly alters the current solution vector (Eq. (4b), Eq. (5b) and Eq. (6b)) to improve exploration phase of algorithm.
3. Scout bee phase of ABC algorithm randomly reinitializes the completely exploited food source (Eq. (20)) on the basis of uniform distribution. Similarly, SSA also incorporates the concept of random relocation of the flying squirrels, which could not explore the forest for optimal winter food source, but it is based on Lévy distribution.

4. In ABC, the scout bee phase is initiated if a predefined number of runs is exceeded. Apparently, this seems similar to seasonal monitoring condition in SSA, but this concept is behaviourally inspired and hence adaptively updated during run-time.

5.3. Bat Algorithm

BA simulates the echolocation behaviour of bats [29]. In BA, n virtual bats (solutions) are represented by position vector X_i, velocity vector V_i and frequency vector F_i in a d-dimensional search space. Initially the bats are randomly distributed. The position of i^{th} bat in population is updated as follows:

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{22}$$

$$V_i(t+1) = V_i(t) + (X_i(t) - Gbest)F_i \tag{23}$$

$$F_i = F_{min} + (F_{max} - F_{min})\beta \tag{24}$$

where β is a uniformly distributed random number in the range $[0, 1]$. The exploration capability of the algorithm is enhanced by employing a local search strategy. If the solution satisfies a certain condition ($rand > pulse\ rate(r)$) then a new solution is generated through random walk [29].

5.3.1. SSA versus BA

1. BA generates new direction of movement of i^{th} bat by considering the best position ($Gbest$) obtained by any bat so far. BA is basically a balanced combination of PSO and local search [96]. In SSA, movements of flying squirrels are directed by globally best flying squirrel (FS^t_{ht}) as well as few locally found best flying squirrels (FS^t_{at}).
2. In BA the exploration is enhanced by randomly updating bat location depending upon the condition i.e. $rand > pulse\ rate(r)$. However the exploration phase of SSA is enhanced by relocating those flying squirrels which could not explore the forest for optimal winter food source. In BA, random walk is generally implemented on the basis of normal distribution, however better exploration is achieved in SSA using Lévy distribution.
3. In position updating mechanism of BA (Eq. (23)), the difference $(X_i(t) - Gbest)$ is multiplied by a random number F_i, however in SSA proper balance between exploration and exploitation is achieved in Eq. (4a), Eq. (5a) and Eq. (6a) by incorporating gliding constant (G_c) additionally.

5.4. Firefly algorithm

FF algorithm proposed by Yang [82] is based on the concept of flashing light production by fireflies. Fireflies produce light using the phenomenon of bioluminescence for attracting the partners for mating. These flashes are also used to attract the prey or to warn the predator. FF algorithm considers that interaction of fireflies is governed by following assumptions [97]:

1. One firefly will be attracted by all other fireflies regardless of their sex.
2. Attraction is directly proportional to their light intensity (or brightness). Thus less bright firefly will tend to move towards the brighter one. The brightness as well as its attractiveness, both are inversely proportional to the distance between them and hence both decrease as their distance increases. If there is no brighter one, then the respective firefly will move in random direction.
3. The brightness of any firefly is determined from the value of fitness function.

The movement of i^{th} firefly towards more attractive j^{th} firefly is determined by Eq. (25) [82]:

$$X_i = X_i + \beta_0 e^{-\gamma r_{ij}^2}(X_j - X_i) + \alpha\epsilon_i \tag{25}$$

where α is randomization parameter, ϵ_i is a random number considered from Gaussian distribution, γ is fixed light absorption coefficient, β_0 is attractiveness at r=0, r_{ij} is the Euclidean distance between two fireflies i and j and calculated as follows:

$$r_{ij} = \|X_i - X_j\| = \sqrt{\sum_{l=1}^{l=d}(X_{il} - X_{jl})^2} \tag{26}$$

The equation Eq. (25) is basically composed of three terms: first term represents the present location of i^{th} firefly, second term denotes the attraction of i^{th} firefly towards more attractive j^{th} firefly and last one is random walk.

5.4.1. SSA versus FF

Both FF and SSA are population based techniques, however differences in the techniques are as follows:

1. In FF algorithm, fireflies are ranked on the basis of their brightness at the end of each iteration and i^{th} firefly moves towards the more attractive j^{th} firefly using Eq. (25). While updating the location of i^{th} firefly, it uses difference information of current solution and global best solution as well as all locally best solutions. On the other hand, SSA uses difference information of current solution either with globally best solution or from few user defined locally best solutions.
2. As discussed previously, SSA uses the concept of effective division of labour due to which the proposed technique injects patterns in three regions using different strategies, however FF algorithm updates the pattern matrix using single strategy.
3. In order to improve the exploration capability of SSA, the concept of random relocation of some flying squirrels is used which is controlled by seasonal monitoring condition. Whereas FF algorithm continuously updates the location of fireflies.

As discussed previously, an efficient metaheuristic must possess a proper balance between exploration and exploitation. However, there is no thumb rule [98] to achieve this. The minor differences in solution updating strategy and random distributions may create huge impact on performance of the designed algorithm [90]. Consequently, SSA becomes a good competitor for existing metaheuristics.

6. Experimental study

The performance of proposed SSA is analyzed by carrying out rigorous experimentation on classic and modern numerical optimization problems. Initially experimentation is conducted on 26 well-known classic benchmark test functions [99, 100]. These functions are described as continuous, discontinuous, linear, non-linear, unimodal, multimodal, convex, non-convex, separable and non-separable. However for testing and validation of a new algorithm, functional features like dimensionality, modality and separability are relatively more significant. It is considered that difficulty of problem increases with the increase in function dimensions as the search space increases exponentially [101]. The modality of a function is considered as the number of ambiguous peaks in the function surface. A function having two or more ambiguous peaks is called as multimodal function. An algorithm that encounters these peaks during their search process may get trapped in these local optimum solutions. This can affect the search process adversely and may get diverted in a different direction far away from the optimal region. On the other hand, separability refers to the difficulty level offered by different benchmark test functions. In general, separable functions are easier to solve in comparison to their non-separable counterpart as each variable of the function is independent from other variable. In the present study, 26 classic benchmark functions are classified on the basis of their modality and separability (Table 4, 6, 8, 10) and four experimental tests are performed to evaluate the performance of SSA. The first and second experimental study confirm the exploitation capability of proposed algorithm, while the third and fourth experimental study check the exploration capabilities. The fifth experimental study is designed on the basis of 7 modern numerical optimization problems considered from IEEE CEC 2014 special session and competition on single objective real-parameter numerical optimization. These benchmark functions have several novel features such as novel basic problems and functions are shifted, rotated, expanded, and combined variants of the most complicated mathematical optimization problems presented in literature [102].

In the last decade a large number of metaheuristics is proposed by various authors. Some of them are modified or improved versions of the basic algorithms while others are based on an entirely new concept. It is observed from the literature that newly developed or modified metaheuristics are not explained in detail [103]. Further, the techniques are documented with partial parametric settings due to which exact replication of

experiments and results, become almost impossible. Moreover, there is no uniformity in the selection of benchmark test suit and experimental conditions are not identical as original ones [103, 104]. Therefore in the present work, basic versions of commonly used optimization algorithms are implemented and used for the purpose of comparative analysis. In each experimental study, the performance of proposed SSA is compared with six nature-inspired optimization algorithms, namely, GA, PSO, BA, FF, MVO and KH. The population size and maximum iterations are set to be 50 and 500 respectively in first four experimental tests for fair comparison. While for fifth experimental study, maximum iterations are 6000 to get 300000 number of function evaluation (NFEs) as per the CEC 2014 recommendation [102]. The commonly used parametric settings of all algorithms are listed in Table 3. 30 independent runs of each algorithm are considered for every benchmark function in each experimentation and best results are boldfaced throughout the paper.

Table 3
Parametric settings of algorithms

Name of Parameter	GA	PSO	BA	FF	MVO	KH	SSA
Crossover fraction	0.8	-	-	-	-	-	-
Selection	Tournament	-	-	-	-	-	-
Crossover	Arithmetic	-	-	-	-	-	-
Mutation	Adaptive feasible	-	-	-	-	-	-
C_1 and C_2	-	2	-	-	-	-	-
Inertia weight (w)	-	0.9	-	-	-	-	-
Loudness	-	-	0.5	-	-	-	-
Pulse rate	-	-	0.5	-	-	-	-
f_{min}, f_{max}	-	-	0, 2	-	-	-	-
α	-	-	-	0.25	-	-	-
β	-	-	-	0.20	-	-	-
γ	-	-	-	1	-	-	-
WEP_{max}, WEP_{min}	-	-	-	-	1, 0.2	-	-
V_f	-	-	-	-	-	0.02	-
D_{max}	-	-	-	-	-	0.005	-
N_{max}	-	-	-	-	-	0.01	-
N_{fs}	-	-	-	-	-	-	4
G_c	-	-	-	-	-	-	1.9
P_{dp}	-	-	-	-	-	-	0.1

6.1. Experimental Test 1

This experimentation evaluates the effectiveness and accuracy of SSA while solving benchmark functions with unimodal and separable inherent characteristics (Table 4). The best, worst, mean and standard deviation (SD) of the results obtained from each algorithm after 30 independent trials are recorded in Table 5. It is clear from results that SSA achieves success in finding global optimum on TF1, TF2 and TF3. None of the algorithm could find global optimum solution for TF4 but results of FF are better than all other methods. For TF1, the performance of SSA is found identical to FF and KH but far better than other methods. Only SSA could reach the global optimum region while solving TF2 and TF3 and other methods fail. To analyze the performance of proposed algorithm for unimodal functions, ANOVA test (Fig. 6) and convergence rate analysis (Fig. 7) is also performed. The data obtained for 30 independent runs is plotted (Fig. 6) and it is observed that performance of SSA is satisfactory for all functions. This is due to the reason that 25^{th} and 75^{th} percentiles of samples collected for SSA decline towards the minimum solution within a narrow interquartile range. The comparison of convergence rate shown in Fig. 7, reveals that SSA converges faster than other algorithms and hence possesses superior convergence capability for such optimization problems.

6.2. Experimental test 2

This experimental test is conducted to observe the performance and consistency of SSA in solving the unimodal but non-separable functions (Table 6). The difficulty level of this test is bit higher in comparison to Test 1, as functions under consideration have non-separable characteristics. The statistical results of 30

Table 4
The description of classical unimodal and separable benchmark functions

Function	Name	Expression	d	Range	F_{min}
TF1	Step	$TF1(x) = \sum_{j=1}^{d} (x_j + 0.5)^2$	30	[-5.12, 5.12]	0
TF2	Sphere	$TF2(x) = \sum_{j=1}^{d} x_j^2$	30	[-100, 100]	0
TF3	Sum Squares	$TF3(x) = \sum_{j=1}^{d} j x_j^2$	30	[-10, 10]	0
TF4	Quartic	$TF4(x) = \sum_{j=1}^{d} j x_j^4 + rand$	30	[-1.28, 1.28]	0

Table 5
Statistical results obtained by GA, PSO, BA, FF, MVO, KH and SSA through 30 independent runs on classical unimodal and separable benchmark functions

Function		GA	PSO	BA	FF	MVO	KH	SSA
TF1	Best	**0**	**0**	2.3690E+03	**0**	**0**	**0**	**0**
	Worst	1	8.000E+01	1.7712E+04	**0**	2	**0**	**0**
	Mean	1.3333E-01	8.0333E+00	8.9976E+03	**0**	4.6667E-01	**0**	**0**
	SD	3.4575E-01	1.4571E+01	3.5579E+03	**0**	5.7135E-01	**0**	**0**
TF2	Best	2.6545E+00	2.4402E+02	1.9341E+04	4.7333E-03	4.0172E-01	1.0942E-02	**7.9225E-20**
	Worst	5.4141E+00	2.7319E+03	6.6267E+04	2.4806E-02	1.5333E+00	2.1477E-01	**5.7411E-07**
	Mean	4.4609E+00	1.3576E+03	3.9384E+04	1.1597E-02	7.8582E-01	5.7558E-02	**4.1689E-08**
	SD	6.6760E-01	6.4254E+02	1.0736E+04	4.3201E-03	2.4795E-01	5.0330E-02	**1.4356E-07**
TF3	Best	4.1877E+01	6.7587E+01	1.2296E+03	7.2047E-02	9.5186E-02	4.7764E-03	**2.0052E-28**
	Worst	8.6577E+01	3.9871E+02	5.5158E+03	2.3183E+00	3.6143E+00	1.9685E-01	**2.3194E-06**
	Mean	6.4872E+01	1.9168E+02	2.9497E+03	6.5045E-01	6.1321E-01	5.7491E-02	**1.5201E-07**
	SD	1.1399E+01	8.8044E+01	1.1535E+03	5.7685E-01	6.6063E-01	4.7055E-02	**4.6741E-07**
TF4	Best	9.2183E-01	1.5797E-01	8.8356E+00	**9.4275E-03**	1.0669E-01	1.1266E-02	3.0998E-02
	Worst	7.8383E+00	3.0081E+00	3.3492E+01	**7.4867E-02**	5.7664E-01	1.4265E-01	9.9258E-01
	Mean	2.9543E+00	1.0084E+00	1.8324E+01	**3.2152E-02**	2.6035E-01	5.2810E-02	5.0192E-01
	SD	1.5476E+00	7.9586E-01	6.1881E+00	**2.0089E-02**	1.0031E-01	3.0737E-02	2.9565E-01

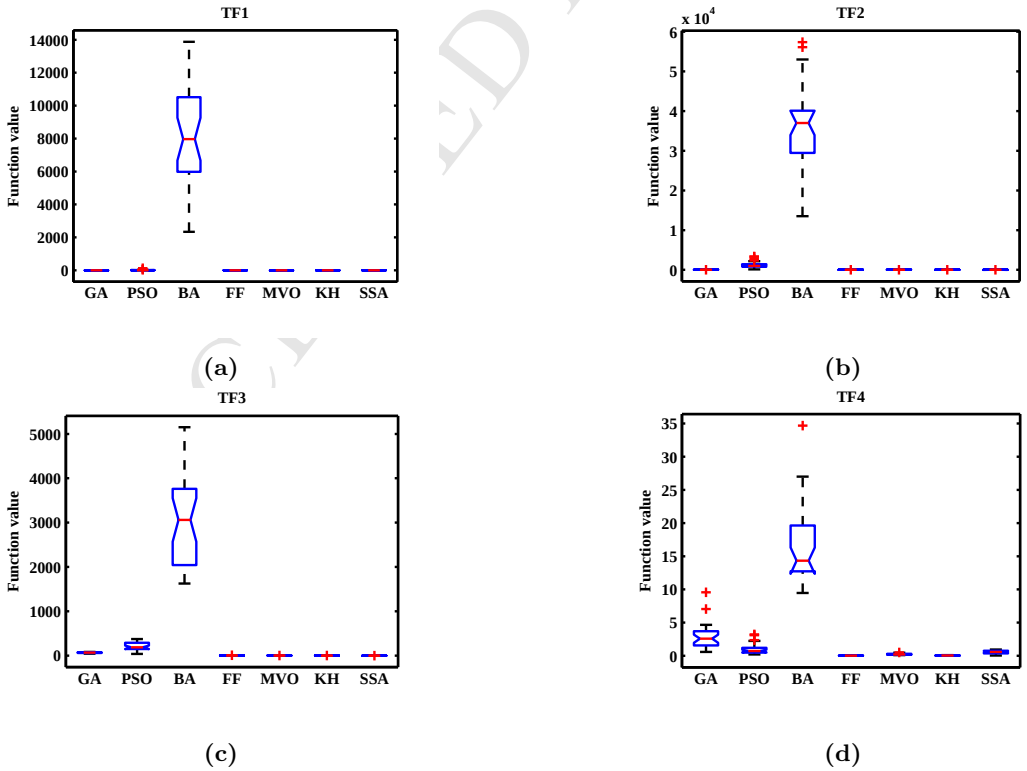


Fig. 6. Variation in global optimization results for benchmark functions (a) TF1, (b) TF2, (c) TF3 and (d) TF4

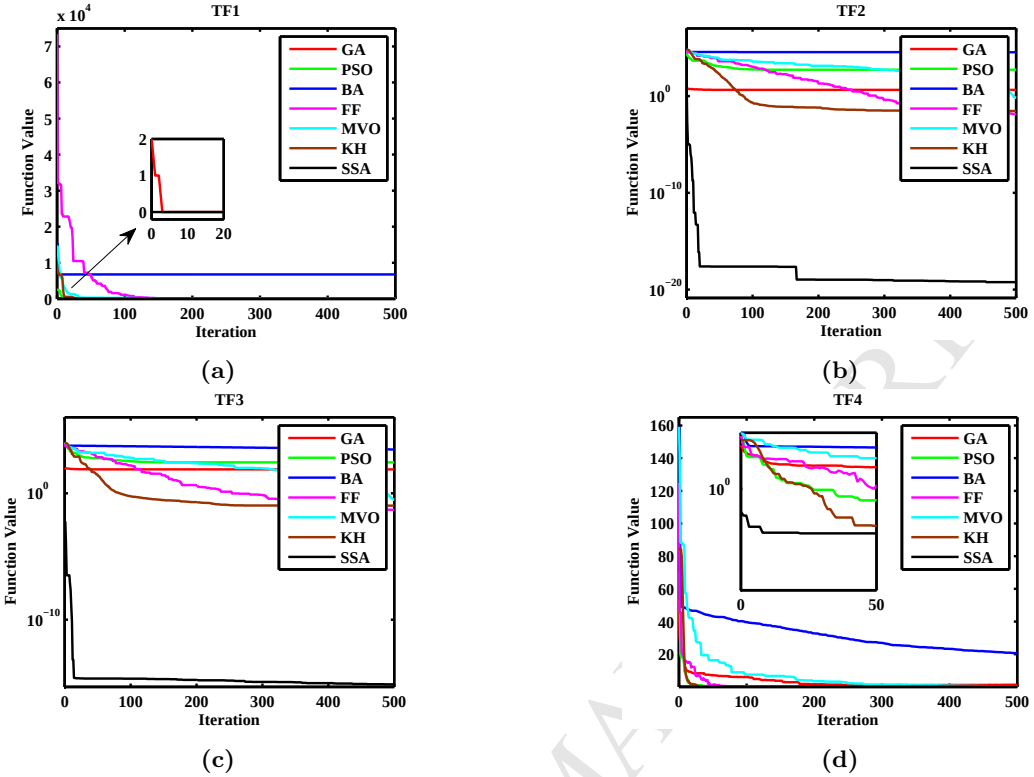


Fig. 7. Convergence rate comparison for benchmark functions (a) TF1, (b) TF2, (c) TF3 and (d) TF4

independent runs, obtained by SSA and other optimization methods are presented in Table 7. It is clear from the results obtained that SSA outperforms the other techniques as it could find global minimum or near global minimum for all eight benchmark functions. To analyze consistency and overall performance of SSA, the data obtained from 30 runs is used for ANOVA test and its results are plotted in Fig. 8. The performance of SSA is found satisfactory and consistent for all functions because 25^{th} and 75^{th} percentiles of samples collected for SSA decline towards the minimum solution within a narrow interquartile range. Further the convergence rate (Fig. 9) of SSA is found better than other optimization algorithms for each benchmark functions. It is observed from the two experimental studies that performance of SSA is quite accurate as well as consistent for unimodal functions. This is due to the realistic modeling of selection ability of flying squirrels for optimal food sources. The flying squirrels search around the neighbourhood of previously visited solutions which provides adequate exploitation capability to SSA.

6.3. Experimental Test 3

The purpose of this experimental test is to check the exploration capability of proposed SSA as functions under consideration have multimodal and separable characteristics. The details of these functions are provided in Table 8. The recorded results of statistical analysis over 30 independent runs of each algorithm while solving these multimodal functions are presented in Table 9. It is evident from the results that SSA is superior on TF13, TF14, TF15 and TF18 whereas its performance is comparable to other methods on TF16. GA defeats SSA and other methods for TF17 as it succeeds in finding the best solution. Variance analysis (Fig. 10) of all algorithms also depict that SSA has less median value (marked by red "-") in comparison to other optimization methods for TF13, TF14, TF15 and TF18. However the results of SSA are found slightly deviated in comparison to other methods for TF16 and TF17 (Fig. 10d-10e).

Fig. 11 shows the recorded convergence characteristics of all algorithms while solving multimodal and separable functions. It is revealed from the results that SSA offers better convergence rate in comparison to other six optimization algorithms for TF13, TF14, TF15 and TF18. KH algorithm defeats SSA in case

Table 6
The description of classical unimodal and non-separable benchmark functions

Function	Name	Expression	d	Range	F_{min}
TF5	Beale	$TF5(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	2	[-4.5, 4.5]	0
TF6	Easom	$TF6(x) = -cos(x_1)cos(x_2)exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	2	[-100, 100]	-1
TF7	Matyas	$TF7(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	2	[-10, 10]	0
TF8	Colville	$TF8(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1(x_2 - 1)^2 + (x_4 - 1)^2 + 19.8(x_2 - 1)(x_4 - 1)$	4	[-10, 10]	0
TF9	Zakharov	$TF9(x) = \sum_{j=1}^{d} x_j^2 + (\sum_{j=1}^{d} 0.5jx_j)^2 + (\sum_{j=1}^{d} 0.5jx_j)^4$	10	[-5, 10]	0
TF10	Schwefel 2.22	$TF10(x) = \sum_{j=1}^{d} \lvert x_j \rvert + \prod_{j=1}^{d} \lvert x_j \rvert$	30	[-10, 10]	0
TF11	Schwefel 1.2	$TF11(x) = \sum_{j=1}^{d} (\sum_{k=1}^{j} x_k)^2$	30	[-100, 100]	0
TF12	Dixon-Price	$TF12(x) = (x_1 - 1)^2 + \sum_{j=2}^{d} j(2x_j^2 - x_j - 1)^2$	30	[-10, 10]	0

Table 7
Statistical results obtained by GA, PSO, BA, FF, MVO, KH and SSA through 30 independent runs on classical unimodal and non-separable benchmark functions

Function		GA	PSO	BA	FF	MVO	KH	SSA
TF5	Best	2.1186E-15	1.3624E-14	2.8649E-11	1.2360E-11	9.2179E-09	4.4607E-13	**2.1832E-29**
	Worst	1.0484E-06	4.9376 E-01	7.6207E-01	3.9985E-09	7.6207E-01	1.5103E-09	**2.7633E-20**
	Mean	1.5004E-07	6.2888E-02	1.5241E-01	8.4259E-10	5.0805E-02	1.7322E-10	**9.5584E-22**
	SD	2.9899E-07	1.6315E-01	3.1004E-01	7.7095E-10	1.9334E-01	3.0481E-10	**5.0400E-21**
TF6	Best	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**
	Worst	**-1**	**-1**	0	0	0	0	**-1**
	Mean	**-1**	**-1**	-3.3347E-02	-7.3333E-01	-9.6664E-01	-9.6666E-01	**-1**
	SD	7.9114E-12	2.8316E-11	1.8257E-01	4.4978E-01	1.8257E-01	1.8257E-01	**0**
TF7	Best	1.6769E-16	8.6209E-17	1.4036E-12	2.9676E-12	2.0125E-10	2.1689E-14	**1.5111E-29**
	Worst	4.1772E-06	7.1849E-12	2.2319E-10	1.2805E-09	4.0157E-08	6.8392E-11	**2.0707E-24**
	Mean	1.2012E-06	8.4738E-13	2.9659E-11	3.1396E-10	1.3148E-08	8.8694E-12	**1.542E-25**
	SD	1.3009E-06	1.5676E-12	4.8089E-11	2.9301E-10	1.1549E-08	1.2854E-11	**4.7571E-25**
TF8	Best	6.9053E-05	3.1763E-11	1.1772E-04	1.0409E-05	1.6967E-04	3.1829E-04	**8.5561E-21**
	Worst	2.2752E-01	7.8739E+00	2.5914E+03	6.7122E-01	5.9549E-02	6.6447E+00	**2.4871E-08**
	Mean	2.9713E-02	1.3576E+00	1.1779E+02	6.9498E-02	1.3897E-02	1.4707E+00	**1.4309E-09**
	SD	4.5589E-02	2.2625E+00	4.7216E+02	1.5653E-01	1.5613E-02	2.0549E+00	**4.6907E-09**
TF9	Best	4.2966E-04	2.4899E-01	4.7992E+00	1.0992E-06	5.4262E-05	1.0937E-02	**1.9954E-23**
	Worst	3.3902E-02	1.2411E+01	2.8843E+02	4.2243E-04	8.5280E-04	5.9581E+00	**1.5225E-07**
	Mean	9.3216E-03	2.8145E+00	5.7976E+01	1.9991E-05	3.3653E-04	5.7794E-01	**5.2215E-09**
	SD	8.2697E-03	3.2838E+00	5.6840E+01	7.6121E-05	1.6264E-04	1.0822E+00	**2.7772E-08**
TF10	Best	8.3300E+00	1.0967E+01	6.9978E+01	1.9719E-01	3.1692E-01	2.2580E+01	**2.2266E-08**
	Worst	1.2512E+01	2.4748E+01	1.6143E+07	7.1570E-01	6.0719E+01	9.0126E+01	**7.1423E-03**
	Mean	1.0964E+01	1.8341E+01	6.3387E+05	3.7229E-01	4.2299E+00	4.5339E+01	**5.1849E-04**
	SD	9.7131E-01	3.2867E+00	2.9400E+06	1.1885E-01	1.3708E+01	1.5991E+01	**1.4144E-03**
TF11	Best	2.2068E+01	3.3599E+03	2.5313E+05	7.3076E-01	1.2525E+01	7.7505E-01	**6.6803E-18**
	Worst	8.5932E+01	3.5414E+04	7.9973E+05	8.9206E+01	2.2745E+02	3.3330E+01	**3.4907E-04**
	Mean	6.2471E+01	1.3279E+04	5.4751E+05	2.3487E+01	5.6701E+01	7.6901E+00	**1.6925E-05**
	SD	1.4303E+01	6.6157E+03	1.3978E+05	2.6648E+01	4.6516E+01	7.5797E+00	**6.6811E-05**
TF12	Best	4.4798E-01	9.2401E+01	3.2393E+04	7.2558E-01	7.5851E-01	6.7985E-01	**1.8308E-01**
	Worst	2.8388E+00	7.3619E+03	4.5924E+05	2.2535E+01	3.3602E+01	7.3470E+00	**6.6951E-01**
	Mean	1.3711E+00	1.1892E+03	1.8201E+05	3.0817E+00	5.1183E+00	1.7567E+00	**2.2412E-01**
	SD	5.7415E-01	1.5741E+03	9.4021E+04	4.4236E+00	8.8946E+00	1.6304E+00	**1.2107E-01**

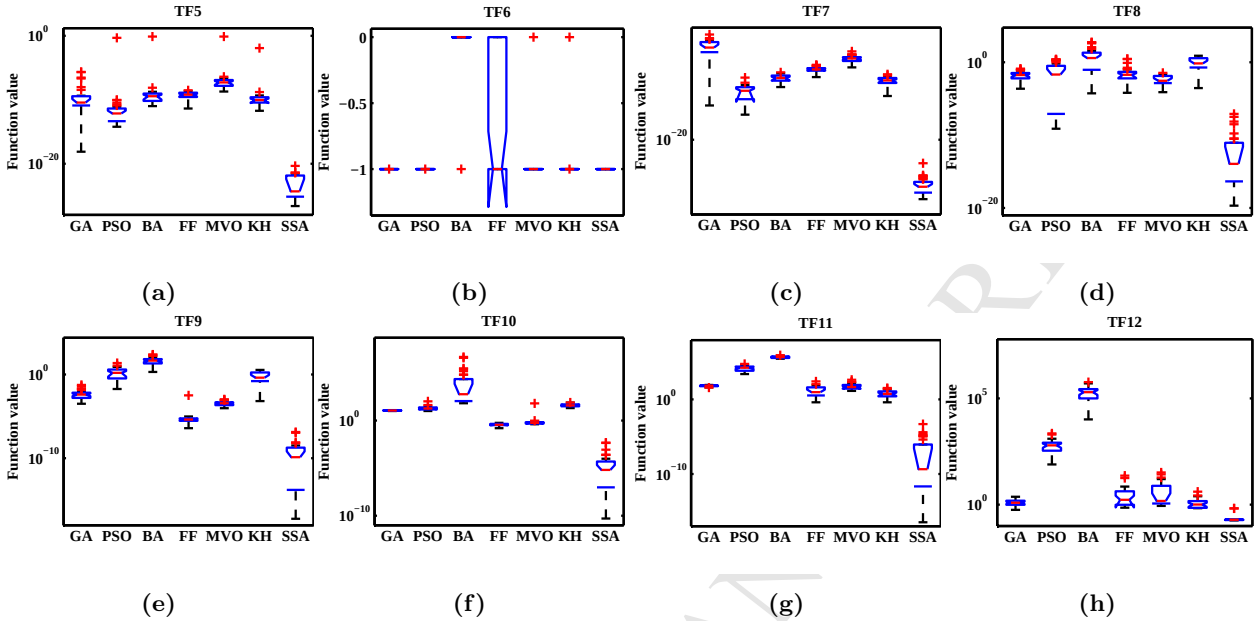


Fig. 8. Variation in global optimization results for benchmark functions (a) TF5, (b) TF6, (c) TF7 (d) TF8 (e) TF9, (f) TF10, (g) TF11 and (h) TF12

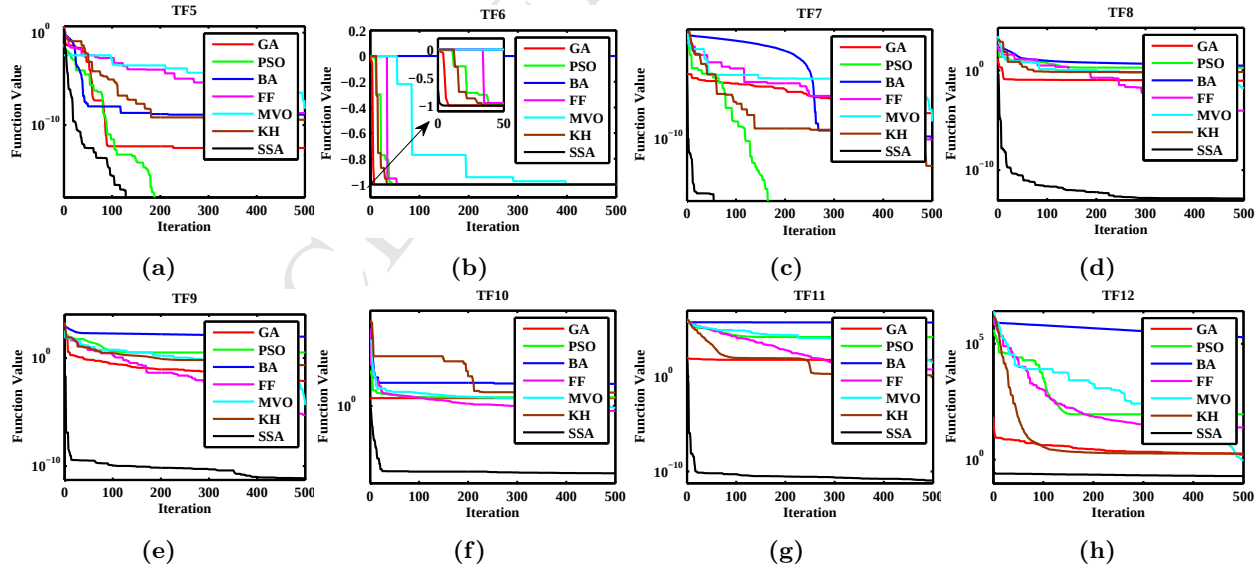


Fig. 9. Convergence rate comparison for benchmark functions (a) TF5, (b) TF6, (c) TF7 (d) TF8 (e) TF9, (f) TF10, (g) TF11 and (h) TF12

of TF16 by providing very fast convergence response. KH and SSA have good convergence response for TF17 but could search only near optimal solution whereas GA found best optimal solution with average convergence response. Further search history of SSA is also recorded (Fig. 12b-12e) while solving very complicated Rastrigin function (Fig. 12a). It is observed from Fig. 12 that flying squirrels explored the search space in the beginning but within 15 iterations most of the flying squirrels moved towards the optimal winter food source. Hence SSA offers a promising performance even for complex multimodal function.

Table 8
The description of classical multimodal and separable benchmark functions

Function	Name	Expression	d	Range	F_{min}
TF13	Bohachevsky1	$TF13(x) = x_1^2 + 2x_2^2 - 0.3cos(3\pi x_1) - 0.4cos(4\pi x_2) + 0.7$	2	[-100, 100]	0
TF14	Booth	$TF14(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	2	[-10, 10]	0
TF15	Michalewicz2	$TF15(x) = -\sum_{j=1}^{d} sin(x_j)(sin(jx_j^2/\pi))^{20}$	2	[0, π]	-1.8013
TF16	Michalewicz5	$TF16(x) = -\sum_{j=1}^{d} sin(x_j)(sin(jx_j^2/\pi))^{20}$	5	[0, π]	-4.6877
TF17	Michalewicz10	$TF17(x) = -\sum_{j=1}^{d} sin(x_j)(sin(jx_j^2/\pi))^{20}$	10	[0, π]	-9.6602
TF18	Rastrigin	$TF18(x) = \sum_{j=1}^{d} (x_j^2 - 10cos(2\pi x_j) + 10)$	30	[-5.12, 5.12]	0

Table 9
Statistical results obtained by GA, PSO, BA, FF, MVO, KH and SSA through 30 independent runs on classical multimodal and separable benchmark functions

Function		GA	PSO	BA	FF	MVO	KH	SSA
TF13	Best	**0**	4.4298E-14	1.6438E+00	4.9443E-08	1.0021E-05	2.8890E-08	**0**
	Worst	2.0161E-09	1.3643E-09	4.3439E+02	9.1103E-06	1.5693E-03	1.4369E-06	**0**
	Mean	1.8113E-10	1.4898E-10	6.9719E+01	3.4682E-06	3.5952E-04	2.2989E-07	**0**
	SD	4.4508E-10	2.7559E-10	1.1194E+02	2.6636E-06	3.7081E-04	2.9457E-07	**0**
TF14	Best	1.1175E-19	1.3482E-14	3.0619E-11	2.7382E-10	4.2336E-09	5.9289E-12	**1.2622E-29**
	Worst	8.4047E-10	1.2419E-09	2.0412E-09	1.9998E-08	2.9865E-06	1.5568E-09	**8.1255E-24**
	Mean	9.8814E-11	8.7865E-11	6.7996E-10	5.4966E-09	5.8026E-07	1.9913E-10	**9.5859E-25**
	SD	2.0126E-10	2.3629E-10	5.4689E-10	4.8029E-09	6.5125E-07	3.1189E-10	**1.7996E-24**
TF15	Best	**-1.8013**	**-1.8013**	**-1.8013**	**-1.8013**	**-1.8013**	**-1.8013**	**-1.8013**
	Worst	**-1.8013**	**-1.8013**	-1	**-1.8013**	**-1.8013**	**-1.8013**	**-1.8013**
	Mean	**-1.8013**	**-1.8013**	-1.6162	**-1.8013**	**-1.8013**	**-1.8013**	**-1.8013**
	SD	1.1799E-09	6.0558E-10	3.1729E-01	1.5947E-09	5.0228E-07	2.5679E-10	**1.0275E-15**
TF16	Best	**-4.6877**	**-4.6877**	**-4.6877**	**-4.6877**	-4.6875	**-4.6877**	**-4.6877**
	Worst	**-3.6946**	-3.2113	-2.5368	**-3.6946**	-2.7363	-2.7851	-3.5563
	Mean	-4.5094	-4.0216	-3.4975	**-4.5706**	-3.9881	-4.1496	-4.3479
	SD	**1.8465E-01**	4.3490E-01	5.1680E-01	1.8507E-01	4.8414E-01	5.6949E-01	3.2785E-01
TF17	Best	**-9.5360**	-8.8819	-7.3075	-9.4087	-8.7506	-9.4288	-9.4806
	Worst	**-7.7744**	-5.7145	-3.6535	-6.6758	-5.6706	-6.3615	-5.9146
	Mean	**-8.8905**	-7.0994	-5.7348	-8.4797	-7.3858	-7.8346	-7.5900
	SD	**3.8495E-01**	8.0055E-01	9.5874E-01	7.5993E-01	9.7022E-01	7.7387E-01	9.9570E-01
TF18	Best	2.0692E+01	6.0234E+01	4.6766E+01	1.7612E+01	6.7986E+01	3.0381E+00	**0**
	Worst	1.3329E+02	1.5111E+02	2.3581E+02	4.9853E+01	2.0129E+02	2.4926E+01	**7.6657E-06**
	Mean	6.3527E+01	1.0309E+02	1.2192E+02	2.5069E+01	1.1867E+02	1.2391E+01	**4.9059E-07**
	SD	2.5564E+01	2.4727E+01	3.9334E+01	6.9514E+00	3.3984E+01	5.4147E+00	**1.5057E-06**

6.4. Experimental Test 4

This test is designed with highest difficulty level in comparison to previous experimental studies. The functions under consideration have multimodal as well as non-separable characteristics. Table 10 presents the details of 8 such benchmark functions with both low as well as high dimensions. The recorded statistical

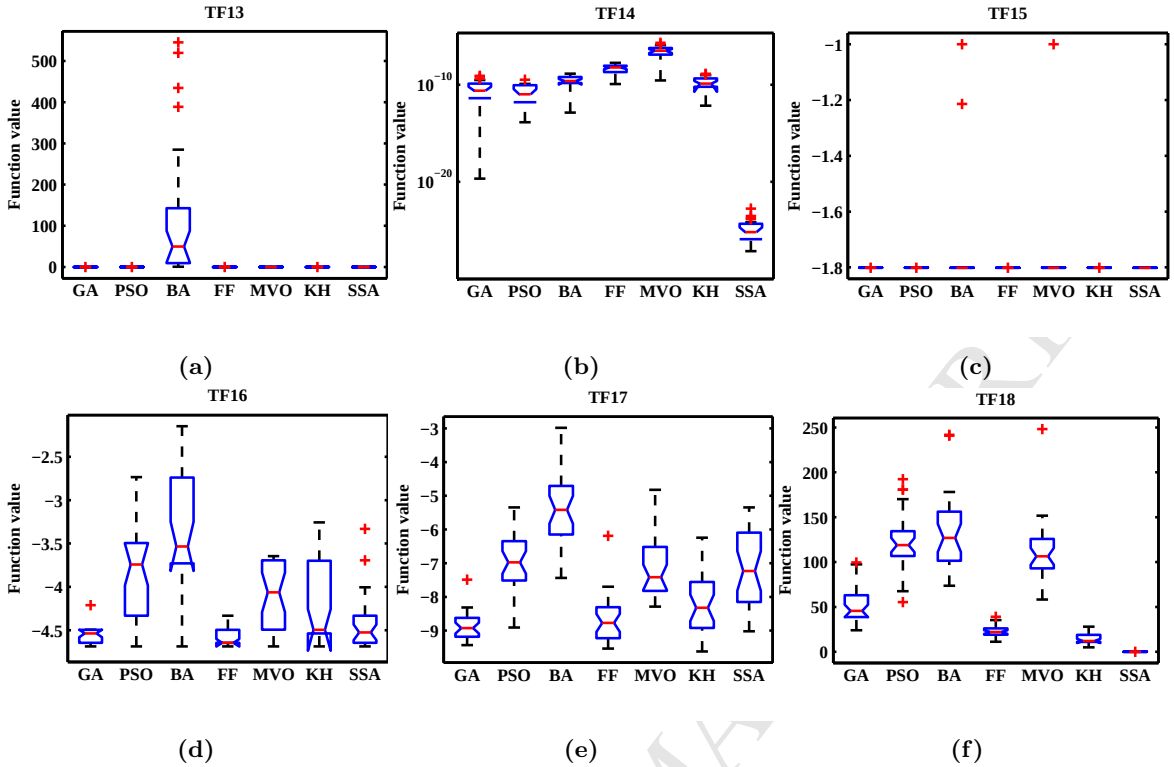


Fig. 10. Variation in global optimization results for benchmark functions (a) TF13, (b) TF14, (c) TF15, (d) TF16, (e) TF17 and (f) TF18

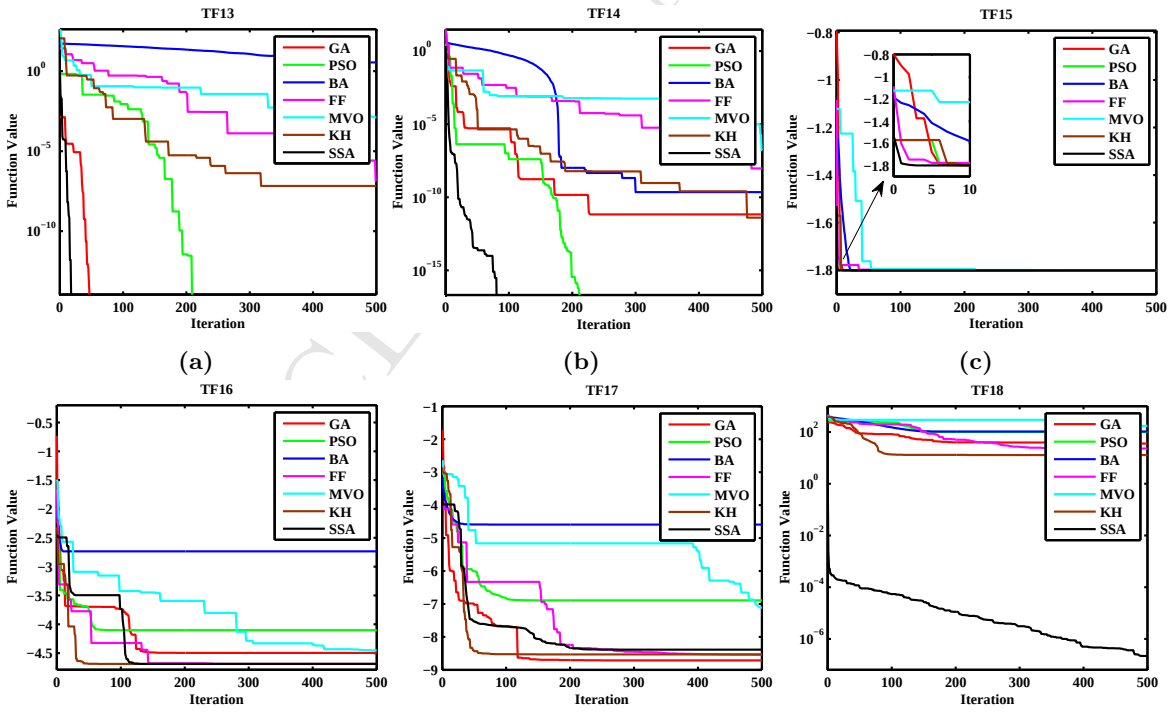


Fig. 11. Convergence rate comparison for benchmark functions (a) TF13, (b) TF14, (c) TF15, (d) TF16, (e) TF17 and (f) TF18

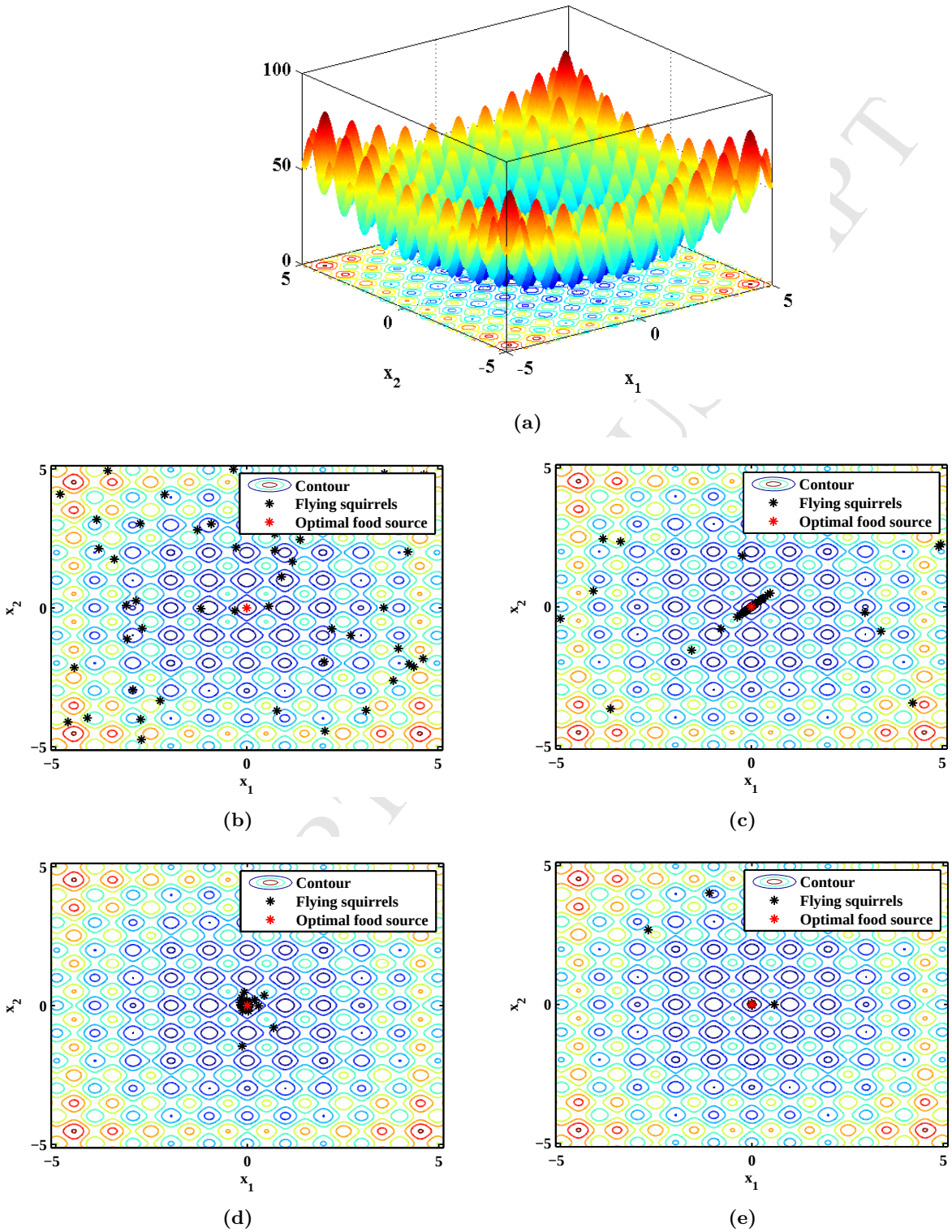


Fig. 12. (a) Perspective view of Rastrigin function (TF18), (b) Position of flying squirrels after 1^{st} iteration, (c) 5^{th} iteration, (d) 10^{th} iteration and (e) 15^{th} iteration

results for 8 benchmark functions are given in Table 11. It is revealed from the results that SSA outperforms the other methods in 7 cases out of 8 benchmark functions. SSA found the best optimal solution for TF19 but could not maintain consistency as its standard deviation (SD) is much higher than GA. All the methods achieve global optimal solution in case of TF20 and TF23 but SSA is found to be most consistent as it offers lowest SD. Results of ANOVA test (Fig. 13) also confirm the satisfactory performance of proposed algorithm. Further convergence rate comparison of optimization algorithms (Fig. 14) reveals that SSA has superior convergence behaviour with sufficient accuracy in comparison to other optimization methods for highly complex multimodal as well as non-separable benchmark functions. The results of experimental test 3 and 4 reveal that SSA provides better exploration capability than the algorithms under consideration. This is due to the incorporation of attributes regarding predator presence and seasonal conditions in foraging behaviour of flying squirrels.

Table 10
The description of classical multimodal and non-separable benchmark functions used in experimental test 4

Function	Name	Expression	d	Range	F_{min}
TF19	Schaffer	$TF19(x) = 0.5 + \frac{sin^2(\sqrt{x_1^2+x_2^2})-0.5}{(1+0.001(x_1^2+x_2^2))^2}$	2	[-100, 100]	0
TF20	Six Hump Camel Back	$TF20(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[-5, 5]	-1.03163
TF21	Boachevsky2	$TF21(x) = x_1^2 + 2x_2^2 - 0.3cos(3\pi x_1)cos(4\pi x_2) + 0.3$	2	[-100, 100]	0
TF22	Boachevsky3	$TF22(x) = x_1^2 + 2x_2^2 - 0.3cos(3\pi x_1 + 4\pi x_2) + 0.3$	2	[-100, 100]	0
TF23	Shubert	$TF23(x) = (\sum_{j=1}^{5} jcos(j+1)x_1 + j)(\sum_{j=1}^{5} jcos((j+1)x_2 + j))$	2	[-10, 10]	-186.73
TF24	Rosenbrock	$TF24(x) = \sum_{j=1}^{d-1} 100(x_{j+1} - x_j^2)^2 + (x_j - 1)^2$	30	[-30, 30]	0
TF25	Griewank	$TF25(x) = \frac{1}{4000}\left(\sum_{j=1}^{d}(x_j - 100)^2\right) - \left(\prod_{j=1}^{d} cos\left(\frac{x_j-100}{\sqrt{j}}\right)\right) + 1$	30	[-600, 600]	0
TF26	Ackley	$TF26(x) = -20exp\left(-0.2\sqrt{\frac{1}{d}\sum_{j=1}^{d} x_j^2}\right) - exp\left(\frac{1}{d}\sum_{j=1}^{d} cos(2\pi x_j)\right) + 20 + e$	30	[-32, 32]	0

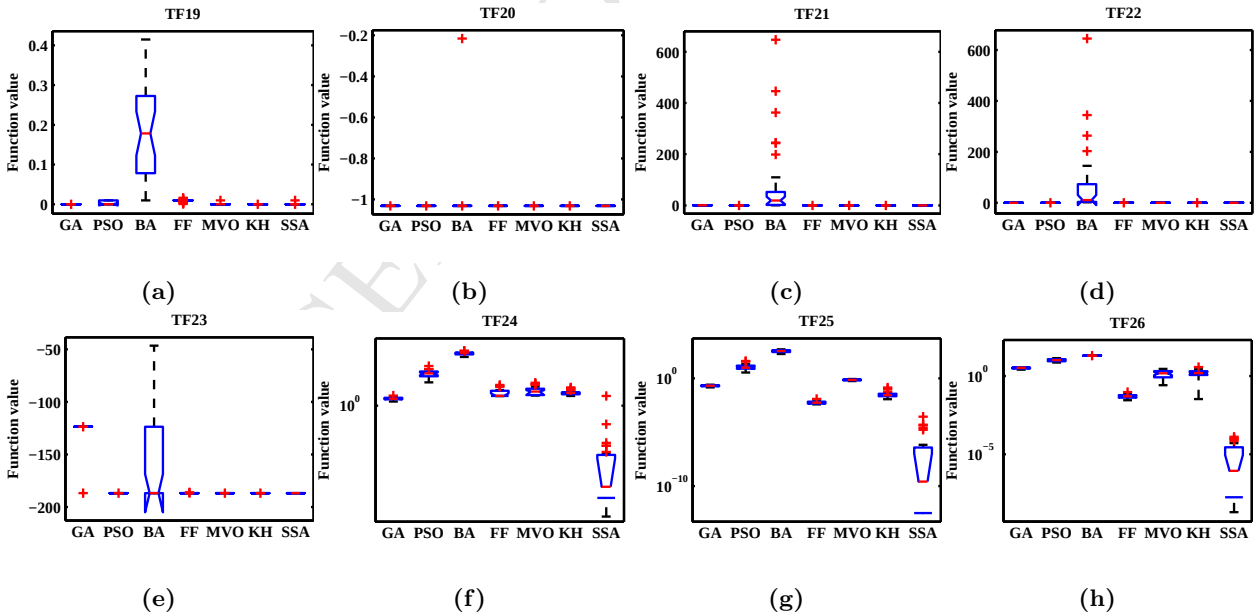


Fig. 13. Variation in global optimization results for benchmark functions (a) TF19, (b) TF20, (c) TF21, (d) TF22, (e) TF23, (f) TF24, (g) TF25 and (h) TF26

Table 11
Statistical results obtained by GA, PSO, BA, FF, MVO, KH and SSA through 30 independent runs on classical multimodal and non-separable benchmark functions

Function		GA	PSO	BA	FF	MVO	KH	SSA
TF19	Best	**0**	**0**	9.7159E-03	4.6615E-03	2.1517E-06	1.2317E-07	**0**
	Worst	**1.6601E-11**	9.7159E-03	3.7329E-01	3.8682E-02	9.7159E-03	2.0569E-05	9.7159E-03
	Mean	**1.6854E-12**	3.5625E-03	1.4118E-01	1.1809E-02	1.3113E-03	3.3604E-06	9.7159E-04
	SD	**4.1693E-12**	4.7621E-03	1.0699E-01	7.3682E-03	3.3529E-03	4.1334E-06	2.9646E-03
TF20	Best	**-1.03163**	**-1.03163**	**-1.03163**	**-1.03163**	**-1.03163**	**-1.03163**	**-1.03163**
	Worst	**-1.03163**	**-1.03163**	-0.21546	**-1.03163**	**-1.03163**	**-1.03163**	**-1.03163**
	Mean	**-1.03163**	**-1.03163**	-0.95001	**-1.03163**	**-1.03163**	**-1.03163**	**-1.03163**
	SD	1.0317E-10	8.2049E-11	2.4904E-01	3.9448E-09	1.6383E-07	6.3582E-10	**4.5168E-16**
TF21	Best	**0**	5.7509E-14	1.0396E+00	2.4610E-08	4.1029E-06	7.5512E-10	**0**
	Worst	2.1831E-01	6.7269E-10	4.9188E+02	1.1613E-05	1.6998E-03	1.4776E-07	**0**
	Mean	1.3099E-01	8.7221E-11	8.1426E+01	2.6897E-06	2.7308E-04	3.0532E-08	**0**
	SD	1.0879E-01	1.6722E-10	1.0682E+02	2.6776E-06	3.6355E-04	3.1152E-08	**0**
TF22	Best	0	1.3156E-14	8.0455E-02	1.1681E-07	1.5801E-05	4.0235E-10	**0**
	Worst	2.2626E-01	8.5075E-10	4.8777E+02	1.7558E-05	6.7957E-04	8.6365E-08	**0**
	Mean	9.8046E-02	4.5608E-11	8.6329E+01	2.2231E-06	2.0969E-04	2.2472E-08	**0**
	SD	1.1404E-01	1.5501E-10	1.2524E+02	3.2160E-06	1.8947E-04	2.5166E-08	**0**
TF23	Best	**-186.73**	**-186.73**	**-186.73**	**-186.73**	**-186.73**	**-186.73**	**-186.73**
	Worst	-123.5767	**-186.73**	-46.5113	-172.9009	-186.7294	**-186.73**	**-186.73**
	Mean	-127.7870	**-186.73**	-144.3476	-186.0049	**-186.73**	**-186.73**	**-186.73**
	SD	1.6023E+01	1.6142E-10	5.4146E+01	2.8688E+00	4.2477E-04	6.5738E-07	**2.6389E-14**
TF24	Best	3.1244E+00	9.4208E+03	1.0786E+07	2.7551E+01	3.1365E+01	3.1196E+01	**3.9637E-19**
	Worst	**2.6849E+01**	2.2741E+07	1.6756E+08	1.2619E+03	3.0082E+03	6.7402E+02	2.8475E+01
	Mean	1.0939E+01	9.1204E+05	6.6428E+07	1.6842E+02	4.5532E+02	1.0894E+02	**9.4919E-01**
	SD	5.4537E+00	4.1286E+06	4.0242E+07	3.1325E+02	7.4799E+02	1.2680E+02	**5.1988E+00**
TF25	Best	1.2089E-01	3.9313E+00	1.5357E+02	3.0228E-03	5.6442E-01	1.2790E-02	**0**
	Worst	2.6858E-01	7.5168E+01	5.8882E+02	9.7682E-03	8.8953E-01	1.0182E-01	**4.1375E-05**
	Mean	1.9528E-01	1.5221E+01	3.4748E+02	5.6221E-03	7.4609E-01	3.9617E-02	**3.435E-06**
	SD	3.9152E-02	1.3167E+01	9.7449E+01	1.5095E-03	8.2684E-02	1.9782E-02	**9.6702E-06**
TF26	Best	2.1422E+00	7.0538E+00	1.7898E+01	2.4479E-02	3.3005E-01	4.2757E-03	**2.2418E-10**
	Worst	3.5251E+00	1.2792E+01	1.9964E+01	8.2453E-02	2.8880E+00	2.5808E+00	**2.5867E-03**
	Mean	3.2409E+00	9.8587E+00	1.9834E+01	5.3031E-02	1.5634E+00	1.4531E+00	**1.3915E-04**
	SD	3.0589E-01	1.5864E+00	4.9211E-01	1.3147E-02	6.1466E-01	8.5527E-01	**4.8513E-04**

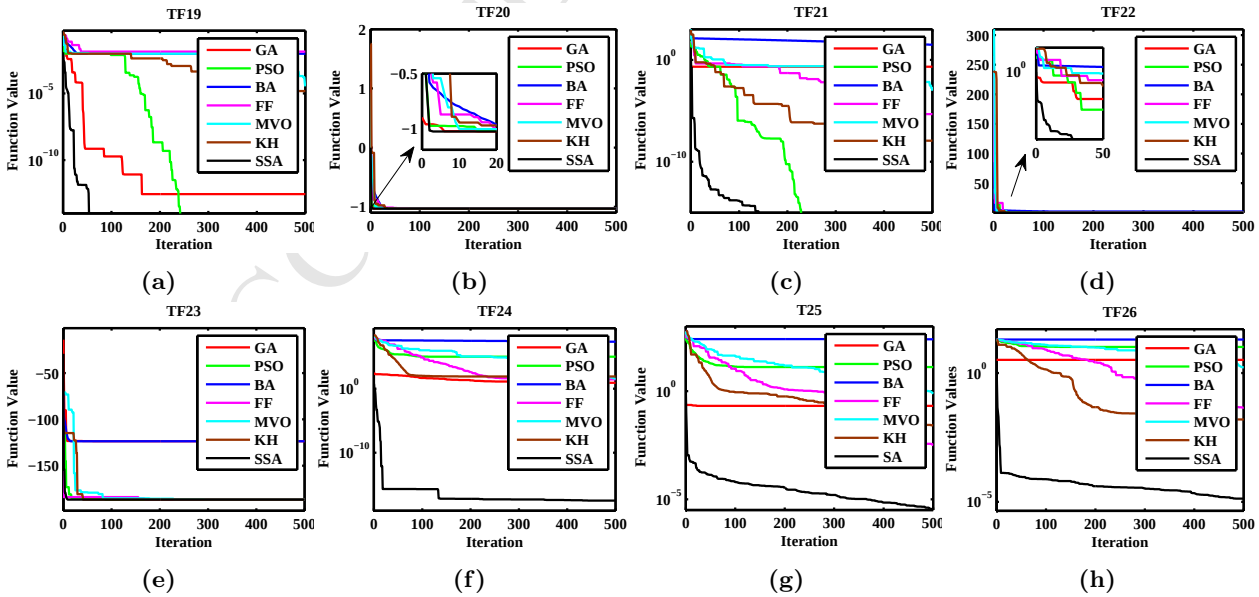


Fig. 14. Convergence rate comparison for benchmark functions (a) TF19, (b) TF20, (c) TF21, (d) TF22, (e) TF23, (f) TF24, (g) TF25 and (h) TF26

6.5. Experimental test 5

The aim of this test is to evaluate the effectiveness and robustness of SSA. Therefore most intensely investigated benchmark functions used in IEEE CEC 2014 are considered for the purpose. CEC 2014 was a special session and competition on single objective real-parameter numerical optimization problems. These modern benchmark functions are especially equipped with various novel characteristics such as basic problems with shifting and rotation. Several hybrid and composite test problems are designed by extracting features dimension-wise for various problems, graded level of linkages, rotated trap problems, and so on [102]. In this experimental study seven CEC 2014 functions are considered with at least one function from each category and the details are provided in Table 12. Results obtained from each algorithm after 30 independent runs are recorded in Table 13. As mentioned previously CEC 2014 functions are specially developed with complex features, consequently all the algorithms can hardly find the global optimum for these seven functions. However, as per the results reported in Table 13, SSA provides more acceptable results on the seven benchmark functions than other algorithms. Further, the analysis of variance shown in Fig. 15 also ensures the stable performance of proposed SSA, as interquartile range is narrow and 25^{th} and 75^{th} percentiles of the samples collected for the SSA during 30 runs, also decline towards the minimum solution. The graphical results of convergence analysis (Fig. 16) show that SSA has promising convergence behaviour in comparison to other six optimization algorithms and hence SSA proves to be the best among other algorithms on seven CEC 2014 functions. It is revealed from the literature that complex optimization problems may be solved efficiently, if the metaheuristic possesses an equilibrium among the exploration and exploitation phases. It is observed from the experimental analysis that SSA maintains the required balance between exploration and exploitation through proper selection of controlling parameters (N_{fs}, G_c and P_{dp}).

It is observed from the results (Table 5, 7, 9, 11) that performance of BA is not very convincing, this may be due to poor parametric tuning of algorithm. Therefore parametric tuning is carried out and recorded results for few cases are presented in Table 14 and Table 15. It observed from the analysis that performance of BA is problem dependent and parameter setting also depends on the type of optimization problem. Similar parametric analysis is also performed for other algorithms used in the comparative analysis. These algorithms have several tuning parameters which result in large number of combinations. The analysis is carried out for all possible combinations and few significant results are presented in Table 16, 17, 18, 19, 20 and 21. The analysis reveals that existing optimizers produce optimal results within certain accuracy range irrespective of the parametric settings for the present 33 benchmark problems. However, the results obtained by the proposed method are found superior in comparison to other methods.

Table 12
The brief description of CEC 2014 benchmark functions

Function	Name	d	Type	Range	F_{min}
TF27	Rotated High Conditioned Elliptic Function (CEC1)	30	U, N	[-100, 100]	100
TF28	Rotated Bent Cigar Function (CEC2)	30	U, N	[-100, 100]	200
TF29	Shifted and Rotated Rosenbrock's Function (CEC4)	30	M, N	[-100, 100]	400
TF30	Hybrid Function 1 (CEC17)	30	-	[-100, 100]	1700
TF31	Composition Function 1 (CEC23)	30	-	[-100, 100]	2300
TF32	Composition Function 2 (CEC24)	30	-	[-100, 100]	2400
TF33	Composition Function 3 (CEC25)	30	-	[-100, 100]	2500

6.6. Comprehensive significance analysis

Several non-parametric statistical tests are discussed in the literature by Derrac [105] to analyze the performance of any two algorithms. However Wilcoxon's test, the most frequently used non-parametric statistical test is considered for the present work and results are summarized in Table 22. The test is performed by considering the best solution obtained by each algorithm for each benchmark function with 30 independent runs and 95% significance level ($\alpha = 0.05$). In Table 22 '+' sign indicates that reference algorithm performed better than the compared algorithm and '-' sign indicates that reference algorithm is inferior to the compared one. The results of last row show that SSA has large number of '+' counts as compared to other optimization algorithms. It confirms that SSA demonstrates statistically significant and superior performance than the six compared algorithms in Wilcoxon test under 95% level of significance.

Table 13
Statistical results obtained by GA, BA, FF, MVO, KH, DA, PSO and SSA through 30 independent runs on CEC 2014 benchmark functions with 30 Dim

Function		GA	PSO	BA	FF	MVO	KH	SSA
TF27	Best	3.2121E+08	2.3069E+07	7.2988E+07	5.1533E+05	9.9798E+05	5.6081E+06	**9.1044E+04**
	Worst	4.6388E+08	4.0159E+08	2.5523E+09	5.6161E+06	5.9024E+06	9.2581E+07	**1.7503E+06**
	Mean	3.8257E+08	1.2539E+08	1.0102E+09	2.1259E+06	2.8959E+06	1.7788E+07	**8.1899E+05**
	SD	3.5194E+07	7.9595E+07	6.1849E+08	1.0427E+06	1.1257E+06	1.6125E+07	**4.0164E+05**
TF28	Best	5.3189E+10	4.4691E+09	3.2169E+10	1.4579E+03	5.8171E+03	7.6562E+03	**2.1599E+02**
	Worst	6.1164E+10	3.5536E+10	9.2830E+10	2.8959E+04	3.8972E+04	2.3140E+05	**2.8185E+04**
	Mean	5.7641E+10	1.4965E+10	6.6295E+10	1.2959E+04	1.7855E+04	5.3419E+04	**1.0049E+04**
	SD	1.9557E+09	8.4581E+09	1.5227E+10	**9.0248E+03**	1.0593E+04	4.6499E+04	9.8268E+03
TF29	Best	1.3568E+04	7.6658E+02	6.0632E+03	4.6835E+02	4.6344E+02	4.0438E+02	**4.0000E+02**
	Worst	1.6908E+04	4.7321E+03	2.5164E+04	**5.0416E+02**	5.6404E+02	5.5844E+02	5.4414E+02
	Mean	1.5512E+04	1.6374E+03	1.2585E+04	4.7708E+02	4.9611E+02	4.9723E+02	**4.5717E+02**
	SD	6.9712E+02	9.1113E+02	4.9397E+03	**8.8691E+00**	3.1158E+01	3.4872E+01	3.9877E+01
TF30	Best	6.1113E+06	1.1359E+05	1.6640E+06	1.0691E+04	2.4724E+04	2.6364E+05	**7.0506E+03**
	Worst	1.5695E+07	4.3836E+06	1.6579E+08	4.3608E+05	4.9843E+05	3.1581E+06	**6.0131E+04**
	Mean	1.0730E+07	9.8676E+05	5.0365E+07	1.2391E+05	1.8421E+05	1.3223E+06	**2.6151E+04**
	SD	2.6344E+06	1.0314E+06	4.5527E+07	9.5801E+04	1.2883E+05	7.1854E+05	**1.5238E+04**
TF31	Best	2.5409E+03	2.6406E+03	2.8174E+03	2.6153E+03	2.6153E+03	2.6153E+03	**2.500E+03**
	Worst	2.5619E+03	2.7447E+03	3.8780E+03	2.6154E+03	2.6158E+03	2.6241E+03	**2.500E+03**
	Mean	2.5540E+03	2.6764E+03	3.0626E+03	2.6153E+03	2.6155E+03	2.6161E+03	**2.500E+03**
	SD	5.4976E+00	2.7560E+01	2.2366E+02	2.9169E-02	1.2047E-01	1.7847E+00	**8.4083E-10**
TF32	Best	2.6033E+03	2.6398E+03	2.7135E+03	2.6002E+03	2.6009E+03	2.6013E+03	**2.600E+03**
	Worst	2.6078E+03	2.6966E+03	2.8588E+03	2.6260E+03	2.6426E+03	2.6291E+03	**2.6004E+03**
	Mean	2.6053E+03	2.6629E+03	2.7801E+03	2.6079E+03	2.6237E+03	2.6229E+03	**2.6000E+03**
	SD	1.3394E+00	1.3087E+01	3.7039E+01	1.0967E+01	1.3737E+01	6.5139E+00	**8.3747E-02**
TF33	Best	2.7008E+03	2.7161E+03	2.7250E+03	2.7040E+03	2.7034E+03	**2.700E+03**	**2.700E+03**
	Worst	2.7011E+03	2.7542E+03	2.7798E+03	2.7069E+03	2.7089E+03	2.7131E+03	**2.700E+03**
	Mean	2.7009E+03	2.7260E+03	2.7499E+03	2.7051E+03	2.7048E+03	2.7055E+03	**2.700E+03**
	SD	6.4016E-02	8.7844E+00	1.5781E+01	8.3501E-01	1.3219E+00	4.0995E+00	**4.8285E-11**

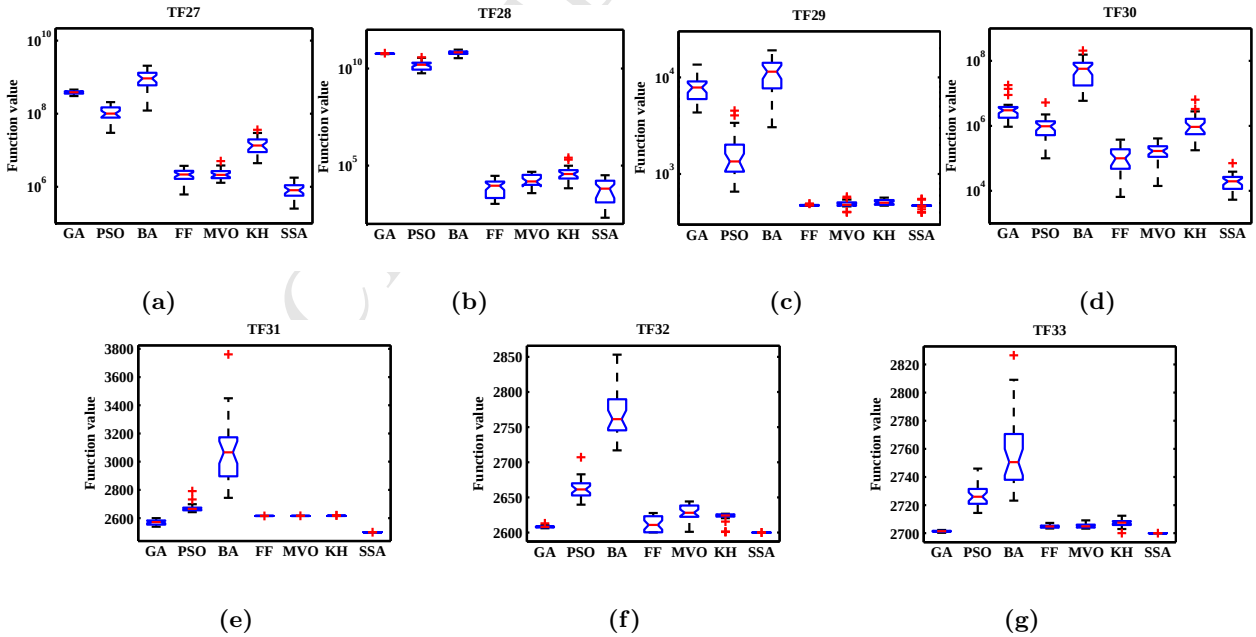


Fig. 15. Variation in global optimization results for benchmark functions (a) TF27, (b) TF28, (c) TF29, (d) TF30, (e) TF31, (f) TF32 and (g) TF33

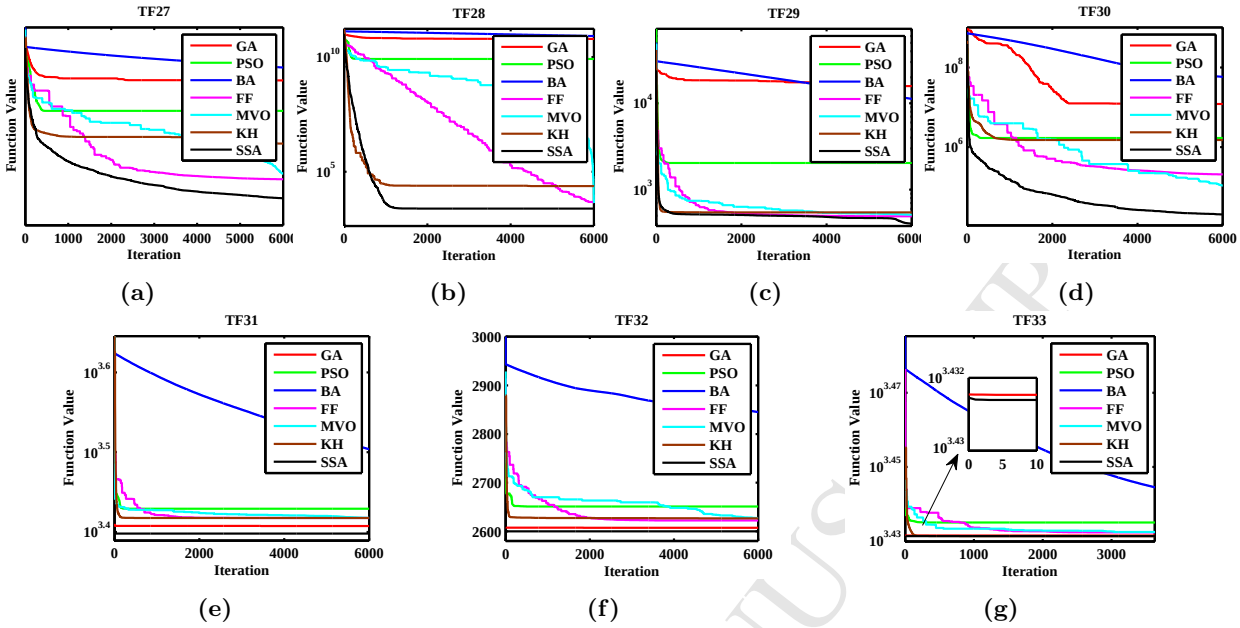


Fig. 16. Convergence rate comparison for benchmark functions (a) TF27, (b) TF28, (c) TF29, (d) TF30, (e) TF31, (f) TF32 and (g) TF33

Table 14
The effect of variation of loudness (A) keeping pulse rate ($r = 0.5$) on the performance of BA

Function	Parameter	$A = 2$	$A = 1.5$	$A = 1$	$A = 0.5$	$A = 0.35$	$A = 0.25$	$A = 0.1$
TF1	Mean	7.0930E+03	5.1973E+03	7.3436E+03	8.0254E+03	9.3928E+03	1.0877E+04	1.0908E+04
	SD	2.7636E+03	1.8015E+03	2.5153E+03	2.9201E+03	3.4583E+03	3.4485E+03	3.7797E+03
TF9	Mean	7.1232E+01	5.6507E+01	7.7221E+01	5.6136E+01	6.4776E+01	7.8141E+01	6.6976E+01
	SD	7.3551E+01	5.5139E+01	8.5643E+01	5.4799E+01	4.8650E+01	7.2201E+01	7.6153E+01
TF13	Mean	5.9455E+01	7.4631E+01	6.7279E+01	1.0399E+02	1.8798E+02	9.0527E+01	1.1965E+02
	SD	1.1565E+02	1.3289E+02	9.6873E+01	1.3961E+02	2.2600E+02	1.2093E+02	1.6878E+02
TF19	Mean	1.7442E-01	1.3640E-01	1.3837E-01	1.7202E-01	1.6560E-01	2.0144E-01	1.7363E-01
	SD	1.1084E-01	1.0372E-01	1.0675E-01	1.0729E-01	1.0390E-01	1.2805E-01	1.1782E-01

Table 15
The effect of variation of pulse rate (r) keeping loudness ($A = 0.5$) on the performance of BA

Function	Parameter	$r = 0.9$	$r = 0.7$	$r = 0.6$	$r = 0.4$	$r = 0.35$	$r = 0.25$	$r = 0.1$
TF1	Mean	1.0321E+04	1.0506E+04	9.0349E+03	7.6793E+03	8.2458E+03	7.7267E+03	7.9055E+03
	SD	4.4886E+03	3.8773E+03	3.7198E+03	3.2272E+03	2.9998E+03	2.7651E+03	2.4840E+03
TF9	Mean	1.6041E+02	7.3076E+01	8.7196E+01	5.6924E+01	4.1761E+01	3.5332E+01	1.6734E+01
	SD	1.1298E+02	5.3355E+01	9.6869E+01	6.3834E+01	5.8968E+01	5.1145E+01	2.7489E+01
TF13	Mean	2.8402E+02	1.9413E+02	1.2746E+02	3.8388E+01	9.6727E+01	3.9832E+01	1.9349E+01
	SD	3.1375E+02	1.8680E+02	1.8827E+02	6.4789E+01	1.7199E+02	1.0640E+02	4.4570E+01
TF19	Mean	2.1917E-01	1.6084E-01	2.2677E-01	1.9497E-01	1.5760E-01	1.9243E-01	1.5051E-01
	SD	1.4373E-01	1.0701E-01	1.3385E-01	1.0420E-01	1.2137E-01	1.1164E-01	1.0732E-01

Table 16
The effect of variation of cognitive constant (C_1) keeping social constant (C_2=2) and inertial weight (w=0.9) on the performance of PSO

Function	Parameter	C_1=0.1	C_1=0.3	C_1=0.5	C_1=1	C_1=1.3	C_1=1.5	C_1=2
TF13	Mean	5.7648E-08	9.6235E-09	2.4484E-09	3.6733E-10	3.9848E-10	2.1737E-10	1.6065E-10
	SD	2.5309E-07	4.5132E-08	9.0398E-09	1.3841E-09	6.9483E-10	4.8836E-10	3.4677E-10

Table 17
The effect of variation of α keeping β=0.20 and γ=1 on the performance of FF

Function	Parameter	α=0.1	α=0.25	α=0.3	α=0.5	α=0.6	α=0.8	α=0.9
TF24	Mean	3.9836E+01	5.3053E+01	1.3151E+02	8.2012E+01	2.2691E+02	7.3650E+01	8.7297E+01
	SD	2.6186E+01	7.4206E+01	2.3776E+02	8.6548E+01	4.5404E+02	8.2723E+01	1.0847E+02

Table 18
The effect of variation of WEP_{max} keeping WEP_{min}=0.2 on the performance of MVO

Function	Parameter	WEP_{max}=0.3	WEP_{max}=0.6	WEP_{max}=0.9	WEP_{max}=1.1	WEP_{max}=1.3	WEP_{max}=1.5
TF14	Mean	1.1116E-06	5.4784E-07	6.1269E-07	6.4180E-07	5.0591E-07	4.6799E-07
	SD	1.2520E-06	6.8009E-07	5.1326E-07	6.9295E-07	4.6183E-07	2.9343E-07

Table 19
The effect of variation of V_f keeping D_{max}=0.005 and N_{max}=0.01 on the performance of KH algorithm

Function	Parameter	V_f=0.05	V_f=0.1	V_f=0.3	V_f=0.6	V_f=0.8	V_f=1	V_f=1.2
TF21	Mean	4.8877E-08	8.8259e-08	1.4895E-06	7.4618E-06	2.1242E-05	3.9623E-05	6.1378E-05
	SD	7.2770E-08	8.2842e-08	1.7252E-06	9.0189E-06	2.8757E-05	5.8044E-05	6.5387E-05

Table 20
The effect of different types of Crossover on the performance of GA while keeping Crossover fraction=0.8, Tournament selection and Adaptive feasible mutation

Function	Parameter	Crossover =Scattered	Crossover =Single point	Crossover =Two points
TF18	Mean	9.2654E+00	1.1424E+01	1.0583E+01
	SD	3.3379E+00	2.9959E+00	2.9161E+00

Table 21
The effect of different types of selection operators on the performance of GA while keeping Crossover fraction=0.8, Arithmetic crossover and Adaptive feasible mutation

Function	Parameter	Selection =Stochastic uniform	Selection =Uniform	Selection =Roulette
TF18	Mean	1.1026E+02	1.9581E+02	1.0917E+02
	SD	3.0096E+01	3.0510E+01	2.8615E+01

Table 22
Results of Wilcoxon's test for SSA against other six algorithms for each benchmark function with 30 independent runs (α = 0.05)

Function	GA vs SSA		SPSO vs SSA		BA vs SSA		FF vs SSA		MVO vs SSA		KH vs SSA	
	p-value	win	p-value	win	p-value	win	p-value	win	p-value	win	p-value	win
TF1(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF2(x)	8.2702E-10	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF3(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF4(x)	5.3430E-08	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF5(x)	4.8415E-13	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF6(x)	3.0495E-05	+	4.9499E-14	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF7(x)	3.1E-03	+	7.8683E-05	+	1.6911E-17	+	3.5480E-14	+	1.8614E-08	+	1.0603E-07	+
TF8(x)	9.2268E-014	+	1.7411E-10	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF9(x)	9.2268E-14	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF10(x)	5.2425E-16	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF11(x)	1.6911E-17	+	4.9940E-01	-	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF12(x)	1.6911E-17	+	7.6236E-14	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF13(x)	6.125E-01	-	2.2395E-06	+	3.5555E-09	+	5.039E-01	-	8.2025E-04	+	5.32E-02	-
TF14(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF15(x)	1.0858E-010	+	6.438E-01	-	5.9588E-08	+	9.1611E-08	+	9.357E-01	-	5.4264E-04	+
TF16(x)	7.9701E-04	+	1.6911E-17	+	1.6911E-17	+	1.000E+00	-	1.2362E-04	+	1.000E+00	-
TF17(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF18(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF19(x)	1.5721E-13	+	3.1900E-02	+	1.6911E-17	+	3.2131E-16	+	1.7733E-05	+	1.5474E-14	+
TF20(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF21(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF22(x)	3.8938E-13	+	1.6911E-17	+	1.6911E-17	+	1.1331E-15	+	1.6911E-17	+	1.6911E-17	+
TF23(x)	6.7645E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF24(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF25(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF26(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF27(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	9.3016E-11	+	2.3507E-15	+	1.6911E-17	+
TF28(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	6.333E-01	-	8.2025E-04	+	7.5007E-09	+
TF29(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	5.91E-02	-	1.42E-02	+	4.6453E-07	+
TF30(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.4295E-09	+	4.8415E-13	+	1.6911E-17	+
TF31(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF32(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
TF33(x)	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+	1.6911E-17	+
+/-		32/1		31/2		33/0		29/4		32/1		31/2

The quantitative analysis of all algorithms is also carried out on the basis of mean absolute error (MAE) for all 33 cases. MAE is an effective and valid performance index used to rank the optimization algorithms [106]. Table 23 provides the average error rates obtained for these benchmark functions. The MAE can be computed as:

$$\text{MAE} = \frac{\sum_{j=1}^{N_s} |m_j - o_j|}{N_s} \tag{27}$$

where m_j is the mean of optimal results produced by an algorithm, o_j is actual value of global optimum for function under optimization and N_s indicates the number of samples. In the present study, N_s number of benchmark functions are considered and computed MAE is given in Table 24. SSA is ranked 1 as it provides minimum MAE. Further SSA is found most consistent in comparison to other optimization algorithms as it reached the global optimum solution 475 times out of 990 runs (Fig. 17).

Table 23
Average Error rates offered by various algorithms including SSA, for 33 benchmark functions

Function	GA	SPSO	BA	FF	MVO	KH	SSA
TF1(x)	1.5004E-07	6.2888E-02	1.5241E-01	8.4259E-10	5.0805E-02	1.7322E-10	**9.5584E-22**
TF2(x)	**0**	**0**	9.6665E-01	2.6667E-01	3.3359E-02	3.334E-02	**0**
TF3(x)	1.2012E-06	8.4738E-13	2.9659E-11	3.1396E-10	1.3148E-08	8.8694E-12	**1.542E-25**
TF4(x)	1.8113E-10	1.4898E-10	6.9719E+01	3.4682E-06	3.5952E-04	2.2989E-07	**0**
TF5(x)	9.8814E-11	8.7865E-11	6.7996E-10	5.4966E-09	5.8026E-07	1.9913E-10	**9.5859E-25**
TF6(x)	0	**0**	1.851E-01	**0**	**0**	**0**	**0**
TF7(x)	**1.6854E-12**	3.5625E-03	1.4118E-01	1.1809E-02	1.3113E-03	3.3604E-06	9.7159E-04
TF8(x)	**0**	**0**	8.162E-02	**0**	**0**	**0**	**0**
TF9(x)	1.3099E-01	8.7221E-11	8.1426E+01	2.6897E-06	2.7308E-04	3.0532E-08	**0**
TF10(x)	9.8046E-02	4.5608E-11	8.6329E+01	2.2231E-06	2.0969E-04	2.2472E-08	**0**
TF11(x)	5.8943E+01	**0**	4.2382E+01	7.2509E-01	**0**	**0**	**0**
TF12(x)	2.9713E-02	1.3576E+00	1.1779E+02	6.9498E-02	1.3897E-02	1.4707E+00	**1.4309E-09**
TF13(x)	1.783E-01	6.661E-01	1.1902E+00	**1.1710E-01**	6.996E-01	5.381E-01	3.398E-01
TF14(x)	9.3216E-03	2.8145E+00	5.7976E+01	1.9991E-05	3.3653E-04	5.7794E-01	**5.2215E-09**
TF15(x)	**7.697E-01**	2.5608E+00	3.9254E+00	1.1805E+00	2.2744E+00	1.8256E+00	2.0702E+00
TF16(x)	1.3333E-01	8.0333E+00	8.9976E+03	**0**	4.6667E-01	**0**	**0**
TF17(x)	4.4609E+00	1.3576E+03	3.9384E+04	1.1597E-02	7.8582E-01	5.7558E-02	**4.1689E-08**
TF18(x)	6.4872E+01	1.9168E+02	2.9497E+03	6.5045E-01	6.1321E-01	5.7491E-02	**1.5201E-07**
TF19(x)	2.9543E+00	1.0084E+00	1.8324E+01	**3.2152E-02**	2.6035E-01	5.2810E-02	5.0192E-01
TF20(x)	1.0964E+01	1.8341E+01	6.3387E+05	3.7229E-01	4.2299E+00	4.5339E+01	**5.1849E-04**
TF21(x)	6.2471E+01	1.3279E+04	5.4751E+05	2.3487E+01	5.6701E+01	7.6901E+00	**1.6925E-05**
TF22(x)	1.0939E+01	9.1204E+05	6.6428E+07	1.6842E+02	4.5532E+02	1.0894E+02	**9.4919E-01**
TF23(x)	1.3711E+00	1.1892E+03	1.8201E+05	3.0817E+00	5.1183E+00	1.7567E+00	**2.2412E-01**
TF24(x)	6.3527E+01	1.0309E+02	1.2192E+02	2.5069E+01	1.1867E+02	1.2391E+01	**4.9059E-07**
TF25(x)	1.9528E-01	1.5221E+01	3.4748E+02	5.6221E-03	7.4609E-01	3.9617E-02	**3.435E-06**
TF26(x)	3.2409E+00	9.8587E+00	1.9834E+01	5.3031E-02	1.5634E+00	1.4531E+00	**1.3915E-04**
TF27(x)	3.8257E+08	1.2538E+08	1.0102E+09	2.1258E+06	2.8958E+06	1.7788E+07	**8.1889E+05**
TF28(x)	5.7641E+10	1.4965E+10	6.6295E+10	1.2759E+04	1.7655E+04	5.3219E+04	**9.8490E+03**
TF29(x)	1.5112E+04	1.2374E+03	1.2185E+04	7.7080E+01	9.6110E+01	9.7230E+01	**5.7170E+01**
TF30(x)	1.0728E+07	9.8506E+05	5.0363E+07	1.2221E+05	1.8251E+05	1.3206E+06	**2.4451E+04**
TF31(x)	2.54E+02	3.7640E+02	7.6260E+02	3.1530E+02	3.1550E+02	3.1610E+02	**2.00E+02**
TF32(x)	2.0530E+02	2.6290E+02	3.8010E+02	2.0790E+02	2.2370E+02	2.2290E+02	**2.00E+02**
TF33(x)	2.0090E+02	2.26E+02	2.4990E+02	2.0510E+02	2.0480E+02	2.0550E+02	**2.00E+02**

Table 24
Ranking of algorithms using MAE

Algorithm	MAE	Rank
SSA	**2.5874E+04**	**1**
FF	6.8539E+04	2
MVO	9.3862E+04	3
KH	5.8069E+05	4
SPSO	4.5734E+08	5
GA	1.7586E+09	6
BA	2.0431E+09	7

6.7. Effect of N_{fs} and n on performance of SSA

SSA is formulated by considering a small region of forest around one hickory tree and hence it can be assumed that the availability of food sources is limited in the vicinity of this region. If the boundary of this region is expanded or a different region is considered, then number of food sources as well as number of flying squirrels will be different. In the proposed algorithm, global optimum is assumed to be one, therefore one hickory nut tree is considered and one squirrel is assumed to be on one tree. However, there is no strict

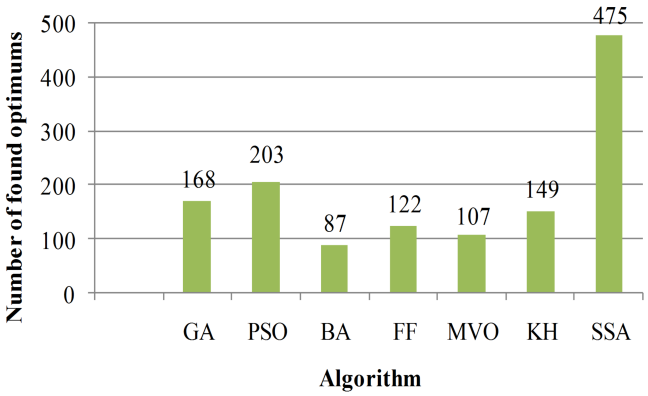


Fig. 17. Comparison of algorithms in finding the global optimal solution out of 990 runs

criterion to decide the number of food sources (N_{fs}) and flying squirrels (n). These parameters depend on the nature of optimization problem and hence user defined. Therefore, the effect of N_{fs} and n on the performance of SSA is also studied on benchmark problems and some significant results are presented in Table 25 and Table 26. It is observed from the analysis that increase in number of food sources results in

Table 25
The effect of food sources (N_{fs}) on the performance of SSA with fixed population size n=50

Function	Parameter	N_{fs}=5%	N_{fs}=8%	N_{fs}=15%	N_{fs}=30%	N_{fs}=40%	N_{fs}=60%	N_{fs}=80%
TF5	Best	4.5368E-27	1.6006E-28	4.1323E-30	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Worst	1.5066E-19	2.3501E-21	6.5759E-22	3.3398E-23	2.4831E-24	2.0901E-26	1.9083E-28
	Mean	5.2319E-21	9.9573E-23	3.7590E-23	1.2447E-24	9.9097E-26	1.0914E-27	1.2975E-29
	SD	2.7485E-20	4.3201E-22	1.3303E-22	6.0955E-24	4.5557E-25	3.8835E-27	4.3389E-29
TF18	Best	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Worst	9.9506E-01	2.4154E-03	6.6716E-05	1.0149E-05	5.9740E-06	1.2212E-08	2.1005E-07
	Mean	3.3318E-02	8.2779E-05	6.0602E-06	3.8840E-07	2.5481E-07	1.4247E-09	7.4923E-09
	SD	1.8165E-01	4.4060E-04	1.5405E-05	1.8467E-06	1.0969E-06	3.2659E-09	3.8335E-08
TF26	Best	1.3483E-10	4.4079E-10	1.1789E-10	3.4897E-12	2.5757E-14	6.1284E-14	8.8818E-16
	Worst	2.5999E-03	5.7313E-04	1.3847E-03	2.8901E-05	5.4130E-04	1.1424E-05	3.6429E-06
	Mean	1.0417E-04	2.6091E-05	7.6173E-05	4.2343E-06	3.4202E-05	1.2524E-06	2.3699E-07
	SD	4.7571E-04	1.0590E-04	2.6303E-04	8.5141E-06	1.0745E-04	2.8913E-06	7.0631E-07

enhanced optimization accuracy as well as stability of the algorithm. Increased percentage of N_{fs} leads to more points in search space around which search is focused. Thus new solutions are generated and better exploration of search space is achieved. Hence, N_{fs} is an attribute of SSA which provides flexibility to vary exploration capability of the algorithm. However, there is no thumb rule for selection of N_{fs} and it depends on the nature of problem. Similarly, increase in population size also optimizes the problem more accurately. However higher values of n provide accuracy at the cost of computational effort while lower values of n leads to unsatisfactory performance of the algorithm. Thus proper selection of N_{fs} and n is necessary for satisfactory performance of SSA. The experimentation results reveal the robustness of proposed algorithm as classical and CEC 2014 functions are considered for optimization. Further efficiency and consistency of developed SSA is proved using standard procedures i.e. convergence rate analysis, Wilcoxon's test and ANOVA. The practical applicability of proposed algorithm is also tested by implementing it on a real-time system.

Table 26
The effect of population size (n) on the performance of SSA with N_{fs}=8%

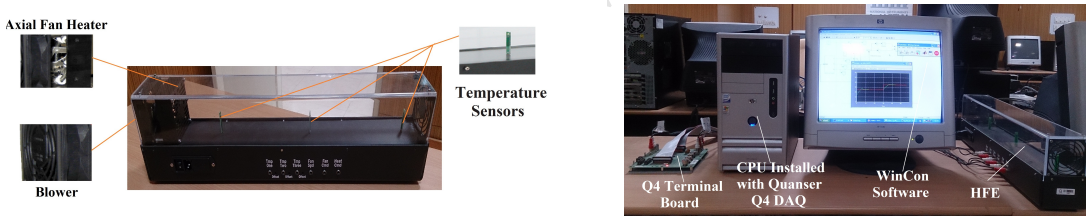
Function	Parameter	n=10	n=20	n=30	n=40	n=60	n=70
TF5	Best	1.1003E-19	1.6719E-25	2.1846E-28	1.1229E-28	7.3426E-29	3.1382E-29
	Worst	1.5457E-07	3.0109E-18	1.8408E-17	1.2910E-20	2.6412E-21	3.4424E-23
	Mean	5.1643E-09	2.6981E-19	6.6253E-19	7.2629E-22	8.9792E-23	1.6093E-24
	SD	2.8219E-08	6.4466E-19	3.3557E-18	2.4352E-21	4.8192E-22	6.2747E-24
TF18	Best	4.4906E-12	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Worst	1.3421E+02	2.9849E+01	2.9849E+01	2.6864E-04	5.9108E-07	5.6132E-07
	Mean	2.4862E+01	5.9748E+00	1.0007E+00	1.3586E-05	3.2866E-08	3.0694E-08
	SD	3.1152E+01	1.2141E+01	5.4486E+00	4.9998E-05	1.2477E-07	1.0906E-07

7. Real-time experimental study

In this study, a common problem of process industry known as "controller tuning" is considered and SSA is employed for the purpose. The results obtained are compared statistically with the existing optimization algorithms. Finally, the performance of SSA optimized controller is validated and compared with the conventional controller on a real-time hardware known as Heat Flow Experiment.

7.1. 2DOFPI control scheme for Heat Flow Experiment

The Quanser Heat Flow Experiment (HFE) is an excellent platform for researchers to design and validate a new control strategy. It consists of a blower followed by a coil-based heater at one end and three equidistant temperature sensors in duct with other end open (Fig. 18a). The apparatus is enclosed by solid Plexiglass chamber and the objective is to ensure a constant temperature profile inside the duct area. Fig. 18b shows the laboratory setup of HFE. The plant can be controlled using MATLAB/Simulink environment installed on a personal computer, however WinCon 5.2 software facilitates the real-time control of HFE. Analog



(a) Apparatus **(b)** Experimental setup

Fig. 18. Heat Flow Experiment

signals ranging from 0 to 5V are used to control the power of heater and speed of fan. These control signals are generated by controller and applied to HFE apparatus through a data acquisition (DAQ) board. The temperature variation inside the chamber depends on the magnitude of applied input voltage signals and measured at three different points along the duct by temperature sensors. The output of sensors is available on three analog input channels of DAQ board. As one side of the duct is open, changes in ambient environment directly affect the temperature inside the plant. Hence, it becomes difficult to maintain a constant temperature profile inside the chamber by conventional one degree of freedom proportional and integral (1DOFPI) controller. In the present work, a two degree of freedom PI (2DOFPI) control scheme (Fig. 19) is employed to control the temperature of HFE. The output (U_c) of controller is generally expressed as follows [107, 108]:

$$\begin{aligned} U_c(s) &= K_p\left[\beta R(s) - Y(s) + \frac{1}{T_i s}\{R(s) - Y(s)\}\right] \\ &= K_p\{\beta R(s) - Y(s)\} + \frac{K_i}{s}\{R(s) - Y(s)\} \end{aligned} \tag{28}$$

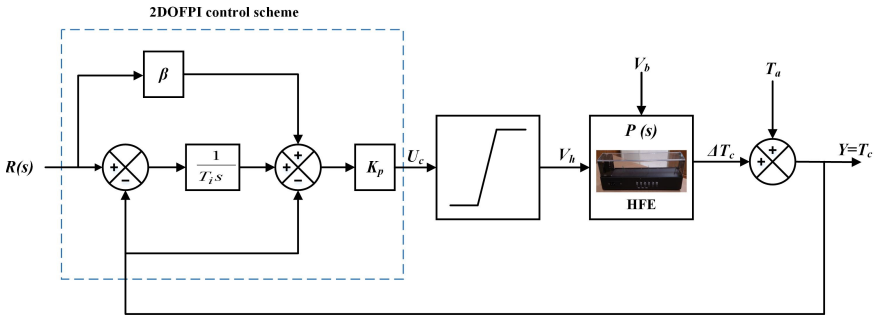


Fig. 19. Block diagram of 2DOFPI control scheme for temperature control of HFE

In Eq. (28), there are three unknown parameters: proportional gain (K_p), integral gain (K_i) and set-point weighing factor (β) known as controller parameters. Appropriate value of these parameters leads to precise control action and stable performance of the plant. The situation presents a combinatorial optimization problem whose optimal solution may be obtained heuristically. In the present study, SSA is applied for tuning of 2DOFPI controller which leads to SSA2DOFPI controller. The transfer function of HFE under consideration is identified as follows:

$$P(s) = \frac{-0.2405s + 1.721}{s^2 + 1.17s + 0.2} \tag{29}$$

The controller parameters are tuned to meet the following design objective:

$$J = 0.30IAE + 0.25t_r + 0.45t_s \tag{30}$$

where IAE is integral absolute error, t_r is rise time and t_s is settling time. The aim is to find a solution set $[K_p, K_i, \beta]$ while optimizing J.

Apart from SSA, other existing algorithms are also employed for tuning of 2DOFPI controller and statistical analysis for 30 independent runs is presented in Table 27. The common parameters of algorithms are considered to be same for fair comparison. For example same parametric search range is considered ($0 \leq K_p \leq 5$, $0 \leq K_i \leq 5$ and $0 \leq \beta \leq 2$) and maximum 800 number of function evaluations are allowed. It is observed from Table 27 that SSA outperforms all other algorithms except KH. The SSA and KH performed equally well on the defined combinatorial optimization problem on statistical grounds. However SSA provides accelerated convergence (Fig. 20) in comparison to KH as well as other techniques. The parameters obtained after tuning by SSA i.e. K_p= 0.7524, K_i= 0.6978 and β= 0.9972 are used in 2DOFPI controller for real-time control of HFE. The experimental results are compared with conventional Tyreus-Luyben tuned 1DOFPI controller. It is revealed from Fig. 21a that SSA2DOFPI controller provides more precise and tight temperature control of HFE. The obvious reason is that SSA2DOFPI controller makes more precise variations in control signal (Fig. 21b) in comparison to conventional controller. Quantitative analysis based on IAE (Fig. 22) also confirms the superiority of proposed technique. The successful implementation of SSA on real-time system proves the robustness and suitability of the algorithm for complex optimization problems.

Table 27
Statistical analysis of algorithms for objective function J

Function		GA	PSO	BA	FF	MVO	KH	SSA
J	Mean	4.7209E+01	5.0446E+01	5.1236E+01	6.5932E+01	4.6276e+001	4.0149E+01	4.0149E+01
	SD	2.7214E+01	5.2425E+00	2.6044E+01	4.0365E+01	2.7573e+001	3.6604E+00	3.6604E+00

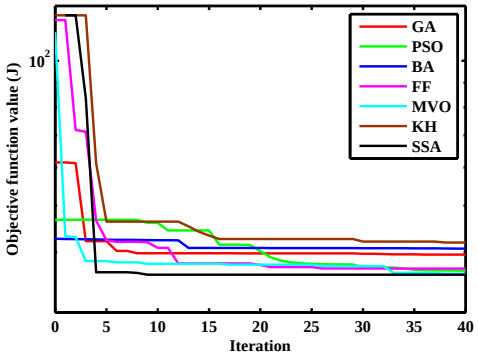
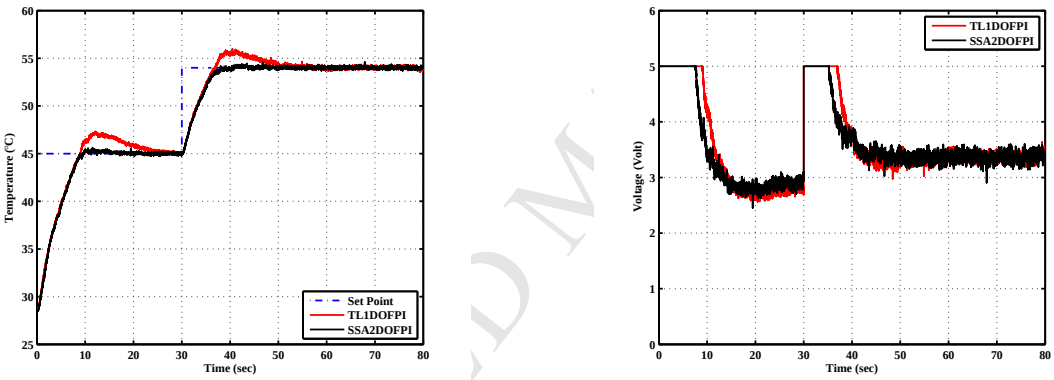


Fig. 20. Convergence rate comparison for objective function J



(a) Temperature variations **(b)** Control effort

Fig. 21. Performance comparison of designed controllers for set point tracking

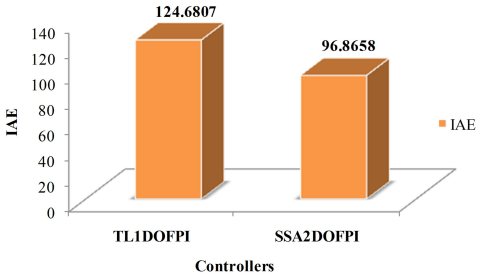


Fig. 22. IAE comparison of designed controllers for set point tracking

8. Conclusion

A novel nature-inspired squirrel search algorithm is designed for unconstrained optimization problems. The foraging behaviour of southern flying squirrels is studied and modeled mathematically including each and every feature of their food search for the desired optimization. The proposed algorithm is tested using several classical and modern unconstrained benchmark functions. It is observed from the comparative statistical analysis that SSA achieves the global optimum solutions with remarkable convergence behaviour in comparison to the other reported optimizers. Further in case of modern highly complex CEC 2014 benchmark functions, all the algorithms can hardly find the global optimum solution but the performance of SSA is found accurate and consistent. Moreover, the SSA is successfully applied to design 2DOFPI controller for temperature control of HFE. Hence it is concluded that SSA offers quite competitive results in comparison to other reported optimizers for numerical optimization as well as real-time problems. The present work provides a basic framework of SSA for low dimension optimization problems, which may be further extended to large scale optimization and constrained optimization problems. In future SSA may also be used for multi-objective optimization problems. The proposed method may also be applied to solve NP-hard combinatorial optimization problems found in real-world.

References

[1] I. BoussaïD, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Information Sciences 237 (2013) 82–117.
[2] A. Gogna, A. Tayal, Metaheuristics: review and application, Journal of Experimental & Theoretical Artificial Intelligence 25 (4) (2013) 503–526.
[3] E. Talbi, Metaheuristics: From Design to Implementation, Wiley Series on Parallel and Distributed Computing, Wiley, 2009.
[4] J. H. Holland, Adaptation in Natural and Artificial systems, University of Michigan Press, Ann Arbor, MI, 1975.
[5] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing, science 220 (4598) (1983) 671–680.
[6] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, IEEE, 1995, pp. 39–43.
[7] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, IEEE computational intelligence magazine 1 (4) (2006) 28–39.
[8] B. Basturk, D. Karaboga, An artificial bee colony (ABC) algorithm for numeric function optimization, in: IEEE swarm intelligence symposium, Vol. 8, 2006, pp. 687–697.
[9] A. Noshadi, J. Shi, W. S. Lee, P. Shi, A. Kalam, Optimal PID-type fuzzy logic controller for a multi-input multi-output active magnetic bearing system, Neural Computing and Applications 27 (7) (2016) 2031–2046.
[10] P. Nguyen, J.-M. Kim, Adaptive ECG denoising using genetic algorithm-based thresholding and ensemble empirical mode decomposition, Information Sciences 373 (2016) 499–511.
[11] R. Arnay, F. Fumero, J. Sigut, Ant colony optimization-based method for optic cup segmentation in retinal images, Applied Soft Computing 52 (2017) 409–417.
[12] K. Z. Gao, P. N. Suganthan, T. J. Chua, C. S. Chong, T. X. Cai, Q. K. Pan, A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion, Expert Systems with Applications 42 (21) (2015) 7652–7663.
[13] A. Gandomi, X. Yang, S. Talatahari, A. Alavi, Metaheuristic Applications in Structures and Infrastructures, Elsevier Science, 2013.
[14] X. Yang, A. Gandomi, S. Talatahari, A. Alavi, Metaheuristics in Water, Geotechnical and Transport Engineering, Elsevier Science, 2012.
[15] S. Das, P. N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 4–31.
[16] X. S. Yang, S. Deb, Cuckoo search via Lévy flights, in: World Congress on Nature Biologically Inspired Computing, NaBIC 2009, 2009, pp. 210–214.
[17] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, Advances in Engineering Software 69 (2014) 46–61.
[18] S. Mirjalili, Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, Neural Computing and Applications 27 (4) (2016) 1053–1073.
[19] X. Yang, Z. Cui, R. Xiao, A. Gandomi, M. Karamanoglu, Swarm Intelligence and Bio-Inspired Computation: Theory and Applications, Elsevier insights, Elsevier Science, 2013.
[20] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Information sciences 179 (13) (2009) 2232–2248.
[21] S. Mirjalili, S. M. Mirjalili, A. Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization, Neural Computing and Applications 27 (2) (2016) 495–513.
[22] A. Kaveh, S. Talatahari, A novel heuristic optimization method: charged system search, Acta Mechanica 213 (3) (2010) 267–289.

[23] X. Li, A new intelligent optimization-artificial fish swarm algorithm, Doctor thesis, Zhejiang University of Zhejiang, China.
[24] R. Martin, W. Stephen, Termite: A swarm intelligent routing algorithm for mobilewireless Ad-Hoc networks, in: Stigmergic optimization, Springer, 2006, pp. 155–184.
[25] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition, in: 2007 IEEE Congress on Evolutionary Computation, 2007, pp. 4661–4667.
[26] A. Mucherino, O. Seref, O. Seref, O. E. Kundakcioglu, P. Pardalos, Monkey search: a novel metaheuristic search for global optimization, in: AIP conference proceedings, Vol. 953, AIP, 2007, pp. 162–173.
[27] S. He, Q. H. Wu, J. R. Saunders, Group search optimizer: An optimization algorithm inspired by animal searching behavior, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 973–990.
[28] X.-S. Yang, Firefly algorithms for multimodal optimization, in: International symposium on stochastic algorithms, Springer, 2009, pp. 169–178.
[29] X.-S. Yang, A new metaheuristic bat-inspired algorithm, in: Nature inspired cooperative strategies for optimization (NICSO 2010), Springer, 2010, pp. 65–74.
[30] X.-S. Yang, Flower pollination algorithm for global optimization, in: International Conference on Unconventional Computing and Natural Computation, Springer, 2012, pp. 240–249.
[31] W.-T. Pan, A new fruit fly optimization algorithm: Taking the financial distress model as an example, Knowledge-Based Systems 26 (2012) 69–74.
[32] A. H. Gandomi, A. H. Alavi, Krill herd: A new bio-inspired optimization algorithm, Communications in Nonlinear Science and Numerical Simulation 17 (12) (2012) 4831–4845.
[33] A. Sadollah, A. Bahreininejad, H. Eskandar, M. Hamdi, Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems, Applied Soft Computing 13 (5) (2013) 2592–2612.
[34] A. Kaveh, N. Farhoudi, A new optimization method: Dolphin echolocation, Advances in Engineering Software 59 (2013) 53–70.
[35] H. Shareef, A. A. Ibrahim, A. H. Mutlag, Lightning search algorithm, Applied Soft Computing 36 (2015) 315–333.
[36] S. A. Uymaz, G. Tezel, E. Yel, Artificial algae algorithm (AAA) for nonlinear global optimization, Applied Soft Computing 31 (2015) 153–171.
[37] S. Mirjalili, The ant lion optimizer, Advances in Engineering Software 83 (2015) 80–98.
[38] O. Abedinia, N. Amjady, A. Ghasemi, A new metaheuristic algorithm based on shark smell optimization, Complexity 21 (5) (2016) 97–116.
[39] W. Yong, W. Tao, Z. Cheng-Zhi, H. Hua-Juan, A new stochastic optimization approach dolphin swarm optimization algorithm, International Journal of Computational Intelligence and Applications 15 (02) (2016) 1650011.
[40] M. D. Li, H. Zhao, X. W. Weng, T. Han, A novel nature-inspired algorithm for optimization: Virus colony search, Advances in Engineering Software 92 (2016) 65–88.
[41] S. Mirjalili, A. Lewis, The whale optimization algorithm, Advances in Engineering Software 95 (2016) 51–67.
[42] A. Askarzadeh, A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm, Computers & Structures 169 (2016) 1–12.
[43] 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, Advances in Engineering Software.
[44] S. Saremi, S. Mirjalili, A. Lewis, Grasshopper optimisation algorithm: Theory and application, Advances in Engineering Software 105 (2017) 30–47.
[45] F. Fausto, E. Cuevas, A. Valdivia, A. González, A global optimization algorithm inspired in the behavior of selfish herds, Biosystems 160 (2017) 39–55.
[46] A. Tabari, A. Ahmad, A new optimization method: Electro-search algorithm, Computers & Chemical Engineering 103 (2017) 1–11.
[47] A. Kaveh, A. Dadras, A novel meta-heuristic optimization algorithm: Thermal exchange optimization, Advances in Engineering Software 110 (2017) 69–84.
[48] E. Jahani, M. Chizari, Tackling global optimization problems with a novel algorithm-mouth brooding fish algorithm, Applied Soft Computing.
[49] A. Baykasoğlu, Ş. Akpinar, Weighted superposition attraction (WSA): A swarm intelligence algorithm for optimization problems–part 1: Unconstrained optimization, Applied Soft Computing 56 (2017) 520–540.
[50] G. Dhiman, V. Kumar, Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications, Advances in Engineering Software.
[51] X. Qi, Y. Zhu, H. Zhang, A new meta-heuristic butterfly-inspired algorithm, Journal of Computational Science.
[52] A. F. Nematollahi, A. Rahiminejad, B. Vahidi, A novel physical based meta-heuristic optimization method known as Lightning Attachment Procedure Optimization, Applied Soft Computing 59 (2017) 596–621.
[53] S. Ghambari, A. Rahati, An improved artificial bee colony algorithm and its application to reliability optimization problems, Applied Soft Computingdoi:https://doi.org/10.1016/j.asoc.2017.10.040.
[54] F. Zhong, H. Li, S. Zhong, A modified ABC algorithm based on improved-global-best-guided approach and adaptive-limit strategy for global optimization, Applied Soft Computing 46 (2016) 469–486.
[55] D. Kumar, K. Mishra, Portfolio optimization using novel co-variance guided artificial bee colony algorithm, Swarm and Evolutionary Computation 33 (2017) 119–130.
[56] A. Ghosh, S. Das, S. S. Mullick, R. Mallipeddi, A. K. Das, A switched parameter differential evolution with optional blending crossover for scalable numerical optimization, Applied Soft Computing 57 (2017) 329–352.
[57] G. Sun, Y. Liu, M. Yang, A. Wang, S. Liang, Y. Zhang, Coverage optimization of VLC in smart homes based on improved

cuckoo search algorithm, Computer Networks 116 (2017) 63–78.
[58] C. Peraza, F. Valdez, M. Garcia, P. Melin, O. Castillo, A new fuzzy harmony search algorithm using fuzzy logic for dynamic parameter adaptation, Algorithms 9 (4) (2016) 69.
[59] E. Bernal, O. Castillo, J. Soria, F. Valdez, Imperialist competitive algorithm with dynamic parameter adaptation using fuzzy logic applied to the optimization of mathematical functions, Algorithms 10 (1) (2017) 18.
[60] E. Méndez, O. Castillo, J. Soria, A. Sadollah, Fuzzy dynamic adaptation of parameters in the water cycle algorithm, in: Nature-Inspired Design of Hybrid Intelligent Systems, Springer, 2017, pp. 297–311.
[61] L. Rodríguez, O. Castillo, J. Soria, P. Melin, F. Valdez, C. I. Gonzalez, G. E. Martinez, J. Soto, A fuzzy hierarchical operator in the grey wolf optimizer algorithm, Applied Soft Computing 57 (2017) 315–328.
[62] J. Perez, F. Valdez, O. Castillo, P. Melin, C. Gonzalez, G. Martinez, Interval type-2 fuzzy logic for dynamic parameter adaptation in the bat algorithm, Soft Computing 21 (3) (2017) 667–685.
[63] F. Olivas, F. Valdez, O. Castillo, C. I. Gonzalez, G. Martinez, P. Melin, Ant colony optimization with dynamic parameter adaptation based on interval type-2 fuzzy logic systems, Applied Soft Computing 53 (2017) 74–87.
[64] K. Srikanth, L. K. Panwar, B. Panigrahi, E. Herrera-Viedma, A. K. Sangaiah, G.-G. Wang, Meta-heuristic framework: Quantum inspired binary grey wolf optimizer for unit commitment problem, Computers & Electrical Engineering-doi:https://doi.org/10.1016/j.compeleceng.2017.07.023.
[65] D. Konar, S. Bhattacharyya, K. Sharma, S. Sharma, S. R. Pradhan, An improved hybrid quantum-inspired genetic algorithm (HQIGA) for scheduling of real-time task in multiprocessor system, Applied Soft Computing 53 (2017) 296–307.
[66] R. Logesh, V. Subramaniyaswamy, V. Vijayakumar, X.-Z. Gao, V. Indragandhi, A hybrid quantum-induced swarm intelligence clustering for the urban trip recommendation in smart city, Future Generation Computer Systems-doi:https://doi.org/10.1016/j.future.2017.08.060.
[67] F. Pulgar-Rubio, A. Rivera-Rivas, M. D. Pérez-Godoy, P. González, C. J. Carmona, M. del Jesus, MEFASD-BD: Multi-objective evolutionary fuzzy algorithm for subgroup discovery in big data environments - A MapReduce solution, Knowledge-Based Systems 117 (2017) 70–78.
[68] A. Franceschetti, E. Demir, D. Honhon, T. Van Woensel, G. Laporte, M. Stobbe, A metaheuristic for the time-dependent pollution-routing problem, European Journal of Operational Research 259 (3) (2017) 972–991.
[69] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67–82.
[70] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, Knowledge-Based Systems 89 (2015) 228–249.
[71] B. S. Arbogast, A brief history of the new world flying squirrels: phylogeny, biogeography, and conservation genetics, Journal of Mammalogy 88 (4) (2007) 840–849.
[72] K. Vernes, Gliding performance of the northern flying squirrel (glaucomys sabrinus) in mature mixed forest of eastern canada, Journal of Mammalogy 82 (4) (2001) 1026–1033.
[73] S. Jackson, P. Schouten, Gliding Mammals of the World, CSIRO Publishing, 2012.
[74] `https://www.warrenphotographic.co.uk/11469-southern-flying-squirrel`.
[75] `https://www.warrenphotographic.co.uk/11474-southern-flying-squirrel`.
[76] Animal facts: Flying squirrel, `https://www.canadiangeographic.ca/article/animal-facts-flying-squirrel`.
[77] R. B. Thomas, P. D. Weigl, Dynamic Foraging Behavior in the Southern Flying Squirrel (Glaucomys volans): Test of a Model, The American Midland Naturalist 140 (2) (1998) 264–270.
[78] J. W. Bahlman, S. M. Swartz, D. K. Riskin, K. S. Breuer, Glide performance and aerodynamics of non-equilibrium glides in northern flying squirrels (glaucomys sabrinus), Journal of The Royal Society Interface 10 (80).
[79] U. M. Norberg, Evolution of vertebrate flight: An aerodynamic model for the transition from gliding to active flight, The American Naturalist 126 (3) (1985) 303–327.
[80] K. L. Bishop, The relationship between 3-d kinematics and gliding performance in the southern flying squirrel, glaucomys volans, Journal of Experimental Biology 209 (4) (2006) 689–701.
[81] P. Stapp, P. J. Pekins, W. W. Mautz, Winter energy expenditure and the distribution of southern flying squirrels, Canadian Journal of Zoology 69 (10) (1991) 2548–2555.
[82] I. Fister, I. F. Jr., X.-S. Yang, J. Brest, A comprehensive review of firefly algorithms, Swarm and Evolutionary Computation 13 (2013) 34–46.
[83] L. Guo, G.-G. Wang, A. H. Gandomi, A. H. Alavi, H. Duan, A new improved krill herd algorithm for global numerical optimization, Neurocomputing 138 (2014) 392–402. doi:http://dx.doi.org/10.1016/j.neucom.2014.01.023.
[84] X.-S. Yang, Firefly Algorithm, Lévy Flights and Global Optimization, Springer, London, 2010, pp. 209–218.
[85] H. Hakli, H. Uğuz, A novel particle swarm optimization algorithm with Lévy flight, Applied Soft Computing 23 (2014) 333–345.
[86] R. Jensi, G. W. Jiji, An enhanced particle swarm optimization with Lévy flight for global optimization, Applied Soft Computing 43 (2016) 248–261.
[87] J. Xie, Y. Zhou, H. Chen, A novel bat algorithm based on differential operator and Lévy flights trajectory, Computational intelligence and neuroscience 2013.
[88] I. Fister Jr, U. Mlakar, J. Brest, I. Fister, A new population-based nature-inspired algorithm every month: is the current era coming to the end?, in: Proceedings of the 3^{rd} Student Computer Science Research Conference, University of Primorska Press, 2016, pp. 33–37.
[89] K. Sörensen, Metaheuristics-the metaphor exposed, International Transactions in Operational Research 22 (1) (2015) 3–18.

[90] P. Civicioglu, E. Besdok, A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms, Artificial intelligence review 39 (2013) 315–346.
[91] J. Yan, W. He, X. Jiang, Z. Zhang, A novel phase performance evaluation method for particle swarm optimization algorithms using velocity-based state estimation, Applied Soft Computing 57 (2017) 517–525.
[92] B. Akay, A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding, Applied Soft Computing 13 (6) (2013) 3066–3091.
[93] S. Das, A. Abraham, A. Konar, Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives, in: Ying Liu et al. (Eds.), Advances of Computational Intelligence in Industrial Systems. Studies in Computational Intelligence, Springer, Berlin, Heidelberg, 2008, pp. 1–38.
[94] C. Ozturk, E. Hancer, D. Karaboga, A novel binary artificial bee colony algorithm based on genetic operators, Information Sciences 297 (2015) 154–170.
[95] A. Rajasekhar, N. Lynn, S. Das, P. Suganthan, Computing with the collective intelligence of honey bees–A survey, Swarm and Evolutionary Computation 32 (2017) 25–48.
[96] S. Mirjalili, S. M. Mirjalili, X.-S. Yang, Binary bat algorithm, Neural Computing and Applications 25 (3-4) (2014) 663–681.
[97] X.-S. Yang, Nature-inspired metaheuristic algorithms, Luniver press, 2010.
[98] G. I. Sayed, A. E. Hassanien, A. T. Azar, Feature selection via a novel chaotic crow search algorithm, Neural Computing and Applications (2017) 1–18.
[99] M.-Y. Cheng, D. Prayogo, Symbiotic organisms search: A new metaheuristic optimization algorithm, Computers & Structures 139 (2014) 98–112.
[100] M.-Y. Cheng, L.-C. Lien, Hybrid Artificial Intelligence-Based PBA for Benchmark Functions and Facility Layout Design Optimization, Journal of Computing in Civil Engineering 26 (5) (2012) 612–624.
[101] M. Jamil, X.-S. Yang, A literature survey of benchmark functions for global optimisation problems, International Journal of Mathematical Modelling and Numerical Optimisation 4 (2) (2013) 150–194.
[102] J. Liang, B. Qu, P. Suganthan, Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore.
[103] N. Veček, M. Mernik, M. Črepinšek, A chess rating system for evolutionary algorithms: A new method for the comparison and ranking of evolutionary algorithms, Information Sciences 277 (2014) 656–679.
[104] A. E. Eiben, M. Jelasity, A critical note on experimental research methodology in EC, in: Proceedings of the Congress on Evolutionary Computation, Vol. 1, 2002, pp. 582–587.
[105] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 3–18.
[106] E. Nabil, A Modified Flower Pollination Algorithm for Global Optimization, Expert Systems with Applications 57 (2016) 192–203.
[107] V. M. Alfaro, R. Vilanova, Model-reference robust tuning of 2DoF PI controllers for first-and second-order plus dead-time controlled processes, Journal of Process Control 22 (2) (2012) 359–374.
[108] N. Pachauri, V. Singh, A. Rani, Two degree of freedom PID based inferential control of continuous bioreactor for ethanol production, ISA Transactions 68 (2017) 235–250.

Highlights

- A novel nature-inspired algorithm named as squirrel search algorithm (SSA) is proposed.
- Testing is performed using 33 optimization benchmark problems.
- The proposed algorithm is compared with six well-known optimization algorithms.
- Experimental results show the superiority of the proposed algorithm.