Expert Systems With Applications 183 (2021) 115408

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A new biologically inspired global optimization algorithm based on firebug reproductive swarming behaviour

Mathew Mithra Noel [a], Venkataraman Muthiah-Nakarajan [a,*], Geraldine Bessie Amali [b], Advait Sanjay Trivedi [c]

[a] School of Electrical Engineering, Vellore Institute of Technology, Vellore 632014, Tamil Nadu, India
[b] School of Computer Science and Engineering, Vellore Institute of Technology, Vellore 632014, Tamil Nadu, India
[c] Red Hat Inc, Raleigh, NC 27601, USA

ARTICLE INFO

Keywords:
Swarm intelligence
Biologically inspired algorithm
Global optimization
Non-convex optimization
Metaheuristics

ABSTRACT

A new biologically inspired derivative-free global optimization algorithm called Firebug Swarm Optimization (FSO) inspired by reproductive swarming behaviour of Firebugs (Pyrrhocoris apterus) is proposed. The search for fit reproductive partners by individual bugs in a swarm of Firebugs can be viewed naturally as a search for optimal solutions in a search space. This work proposes a mathematical model for five different Firebug behaviours most relevant to optimization and uses these behaviours as the basis of a new global optimization algorithm. Performance of the FSO algorithm is compared with 17 popular heuristic algorithms on the Congress of Evolutionary Computation 2013 (CEC 2013) benchmark suite that contains high dimensional multimodal as well as shifted and rotated functions. Statistical analysis based on Wilcoxon Rank-Sum Test indicates that the proposed FSO algorithm outperforms 17 popular state-of-the-art heuristic global optimization algorithms like Guided Sparks Fireworks Algorithm (GFWA), Dynamic Learning PSO (DNLPSO), and Artificial Bee Colony Bollinger Bands (ABCBB) on the CEC 2013 benchmark.

1. Introduction

The problem of finding the minimum of a real valued function of a large number of real variables is of great importance in many areas. Consider a real valued function $f : \mathfrak{R}^D \rightarrow \mathfrak{R}$. The problem of finding a $\mathbf{x}^* \in \mathfrak{R}^D$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x}$, where f is non-convex is referred to as unconstrained nonlinear global optimization. When the search space is not the whole of $\mathfrak{R}^D$, the problem is referred to as constrained optimization. A majority of heuristic optimization algorithms can only solve unconstrained problems. To solve constrained optimization problems, the "Penalty Function" approach (Chong & Zak, 2005) or the "Barrier Function" approach (Vanderbei, 2001) can be used to transform the constrained problem into an unconstrained problem (which can then be solved by algorithms that perform an unconstrained search). If f is a convex function, efficient and reliable algorithms for computing the global minimum are known and high quality large scale convex optimization software suites are readily available (many as freeware). A strictly convex function can have no more than one minimum on an open set, so the problem of computing the global minimum is greatly simplified for convex functions. However when f is non-convex the problem of global optimization is NP-complete and hence efficient algorithms for finding the minimum do not exist. Algorithms that are guaranteed to find the global minimum of non-convex problems usually work by solving convex subproblems but have exponential worst case complexity. Thus a fundamental theoretical trade-off between finding sub-optimal solutions and not finding the exact global minimum every time is unavoidable when solving non-convex nonlinear optimization problems. Despite this theoretical limitation, the problem of global optimization of non-convex functions is ubiquitous in electrical and computer engineering such as training neural networks (Cang, Lin, & Shieh, 2012), VLSI design (Chen, Liu, Su, & Chen, 2020; Shen, Liu, & Chandler, 2015), non-standard and multidimensional filter design (Wang, Lu, Fu, & Xiong, 2005; Yun et al., 2016), signal processing (Ghamisi, Couceiro, Martins, & Benediktsson, 2014; Uymaza, Tezel, & Yel, 2015) and control (Mandal et al., 2015; Mohamed, Berzoy, & Mohammed, 2017). The problem of global optimization is also important in many diverse fields of science and

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: mathew.m@vit.ac.in (M.M. Noel), mnvenkataraman@vit.ac.in (V. Muthiah-Nakarajan), geraldine.amali@vit.ac.in (G.B. Amali), atrivedi@redhat.com (A.S. Trivedi).

https://doi.org/10.1016/j.eswa.2021.115408
Received 19 April 2020; Received in revised form 30 May 2021; Accepted 9 June 2021
Available online 17 June 2021

engineering such as computational biology (protein folding problem), civil engineering (structural optimization) and economics (portfolio optimization). More generally the problem of the design of optimal engineering structures and systems can in most cases be formulated as an optimization problem. The most challenging economically important optimization problems like the design of efficient antennas and optimal controllers are invariably nonconvex global optimization problems.

The problem of global optimization is NP-complete. Thus algorithms that locate a deep local minimum instead of the global minimum or locate the global minimum for a limited class of functions are of interest. One approach that has proven successful is to use heuristics to explore promising solutions and reduce the search space instead of performing an exhaustive search for the global minimum. These heuristics have, in the past taken inspiration from Biology and Physics. Biologically inspired heuristics can further be classified as evolutionary or social behaviour based heuristics. Evolutionary optimization heuristics copy natural evolutionary processes that create organisms that are optimally designed for their environment over time. Social behaviour based heuristics take inspiration from the collective behaviour of animals and insects. In the past heuristic based global optimization algorithms inspired by biological and physical processes have proved to be useful in solving many practical problems. Popular biologically inspired optimization algorithms include Genetic Algorithm (GA) (Goldberg, 1989), Particle Swarm Optimization (PSO) (Eberhart & Kennedy, 1995), Ant Colony Optimization (ACO) (Dorigo & Gambardella, 1997), Grey Wolf Optimization (GWO) (Mirjalili, Mirjalili, & Lewis, 2014) and Artificial Bee Colony (ABC) (Karaboga, 2005) Optimization. Simulated Annealing (SA) (Kirkpatrick, Gelatt, & Vecchi, 1983) is a powerful and widely used global optimization algorithm inspired by the physics of annealing of crystals. Optimization problems can be classified based on the nature of the search space as either continuous or discrete (combinatorial) problems. Algorithms like SA and GA can be applied to both continuous and combinatorial problems. Discrete (combinatorial) problems like the famous Travelling Salesman Problem (TSP) are common in computer science. In this work, only the problem of continuous global optimization is considered. Although a wide variety of global optimization algorithms exist, the celebrated No Free Lunch (NFL) (Wolpert & Macready, 1997) theorem guarantees that all optimization algorithms' performance over all possible optimization problems is the same on average. Thus statements regarding the superiority of a particular optimization algorithm can only be made with respect to a restricted set of benchmark optimization problems. Also, despite popular belief to the contrary, it is not possible to create the 'best suite of benchmark problems' since a more complex algorithm tuned to explore multiple local minima may perform poorly compared to a simpler algorithm on a weak multimodal problem. In this paper, the Congress of Evolutionary Computation 2013 (CEC 2013) (Liang, Qu, Suganthan, & Hernandez, 2013) test suite of benchmark optimization problems is used.

In the past, heuristics for global optimization have taken inspiration from natural evolutionary improvement processes, crystal annealing, swarming behaviour for predator avoidance, herding, food foraging, and hunting behaviours. This paper proposes a new heuristic for global optimization based on reproductive swarming behaviour in Pyrrhocoris apterus where individual insects attempt to find the best potential mate. To the best of the authors' knowledge, the phenomenon of reproductive swarming (social aggregation and gregarious behaviour in certain insects and animal populations attempting to maximize reproductive success) has not been considered an inspiration for optimization. Since populations that gather together for mating are composed of individual organisms that aim to maximize reproductive success, there is a natural correspondence between reproductive swarming and global optimization. In this paper, an interesting global optimization algorithm inspired by reproductive swarming social behaviour of Firebugs is described.

The Wilcoxon's Rank-Sum Test (Conover, 1980) indicates that the FSO outperforms 17 popular and state-of-the-art algorithms on all 28 CEC 2013 suite problems. The FSO either outperforms each of the 17 algorithms taken for comparison directly, or outperforms on the average another algorithm known to outperform the particular algorithm taken for comparison.

In summary, the main contributions of this paper are as follows:

1. Proposal of a new biologically inspired heuristic inspired by insect social behaviour aimed at maximizing reproductive success.
2. Proposal of a new mathematical model of reproductive swarming behaviour.
3. Proposal of a new heuristic global optimization algorithm based on reproductive swarming.
4. Presentation of extensive experimental results conclusively demonstrating the superior performance of the proposed FSO algorithm over 17 popular algorithms (Table 5) on up to 100 dimensional problems in the CEC 2013 test suite.

The rest of the paper is organized as follows: Section 2 discusses the biological inspiration behind the Firebug Swarm Optimization algorithm and presents the FSO algorithm, Section 3 presents experimental results comparing FSO with other state-of-the-art algorithms and finally Section 4 discusses the implications of the experimental results.

2. Firebug Swarm Optimization (FSO) algorithm

The biological inspiration behind the FSO algorithm is presented in this section. Firebugs (Pyrrhocoris apterus) have fundamentally two types of behaviour: they can roam and explore as solitary individuals or display gregarious behaviour and form aggregations. These aggregations help the firebugs to reduce predation and find fit mates for reproduction. During summer, firebugs form these aggregations. The movement of firebugs attempting to find the best mate can naturally be viewed as an optimization process. Firebugs display complex and varied behaviours so the aspects of Firebug behaviours most relevant to search must be delineated. The various behaviours exhibited by Firebugs are discussed in detail in (Gyuris, Fero, Tartally, & Barta, 2011; Schmuck, 1995; Schofl & Taborsky, 2002; Svádová, Exnerova, & Pavel, 2014; Zdarek, 1970; Zemek & Socha, 2010). Fig. 1 (taken at the author's workplace, where they are numerous) shows solitary and gregarious behaviour in Firebugs.

The FSO algorithm models the following Firebug behaviours:

I. Males Firebugs use chemical signals via pheromones to attract and defend female bugs colonies (Schmuck, 1995; Zdarek, 1970).
II. Each male bug only mates with the fittest female in its colony. Male and female bugs that successfully mate form a tandem formation and stay together for a considerable time (Fig. 1(b)) (Schofl & Taborsky, 2002).
III. Female bugs are attracted strongly towards the male bug defending their colony and to a lesser extent to other male bugs (Gyuris et al., 2011; Schofl & Taborsky, 2002).
IV. Male bugs are also attracted to the chemical signals (pheromones) of fit females, not in its colony (Zdarek, 1970).
V. Swarm cohesion: Individuals in the swarm do not disperse but move together as a single aggregation as shown in Fig. 1(c) (Schmuck, 1995; Svádová et al., 2014).

A mathematical model that captures the essence of the above firebug behaviours is proposed in the following.

2.1. Behaviour I: Formation of female colonies

Since firebugs search for fit mates, and our goal is to minimize a cost function, fit bugs must be associated with low cost values. Since each male bug has a colony of N_F female bugs, the FSO algorithm starts with $N_M N_F$ female bugs randomly distributed in the search space. Each bug has a position (modelled as a column vector) and real scalar cost function value associated with it. In other words, the initial position of each female bug is treated as a uniform vector random variable in the search space.

(a) Solitary Firebug (b) Tandem formation (c) Firebug aggregation

Fig. 1. Firebug (Pyrrhocoris apterus).

2.2. Behaviour II: Mate selection

Since the male bug only mates with the fittest female bug in its colony, each male bug's location is initialized to the location of the best female bug in its colony. The FSO implements Behaviour I, by assigning the best female in a colony to the dominant male controlling that particular colony. Thus, each male bug position is always updated to be identical to the position of the best female in its colony.

A fundamental difference between FSO and other popular heuristic global optimization algorithms is that the FSO is designed to use element-wise Hadamard matrix multiplication operation to implement position updates.

Using Hadamard matrix multiplication allows the FSO to take advantage of the fast Single Instruction Multiple Data (SIMD) feature of modern CPUs and GPUs. SIMD allows modern CPUs to perform multiple arithmetic operations in parallel. This means that a single instruction is used to perform the same operation on multiple data items in parallel, reducing execution time. In other words, all the females in a colony are updated in parallel when the FSO is run on modern multi-core CPUs and GPUs.

Further, cost function evaluations are also highly vectorized since all individuals' position vector is passed as a single matrix to the CEC 2013 test suite. CEC 2013 accepts this matrix of position vectors (the ith column is the position vector of the ith individual) and returns all the corresponding costs as a single array. Expressing operations in terms of matrices instead of scalars is well known to significantly speedup computation in popular very high-level interpreted languages used for scientific computing like Python and MATLAB (Reams, 1999; Ryser, 1963; Styan, 1973).

2.3. Behaviour III: Chemotactic movement of female bugs

After initialization the location of the female bugs is updated in accordance with behaviour III. Each female bug moves towards the dominant bug controlling its colony as well as weakly towards random male bugs. Since one of the main design goals behind the FSO algorithm is to avoid inefficient scalar operations and use efficient matrix operations, the update equation for the female bugs is written in terms of matrices instead of vectors. The positions of all the female bugs in a particular male's colony are stored in a single matrix and updated synchronously. Let $\mathbf{m}(m).\mathbf{F}$ be the D by N_F matrix whose columns correspond to female bug positions. The following matrix update equation is proposed for the simultaneous update of all female bugs in a particular colony using efficient Hadamard multiplication operations:

$$\mathbf{M_x} \leftarrow \text{repmat}(\mathbf{m}(m).\mathbf{x}, 1, N_F) \tag{1}$$

$$\mathbf{M_y} \leftarrow \text{repmat}(\mathbf{m}(a).\mathbf{x}, 1, N_F) \tag{2}$$

where a is an random integer between 1 and N_F. The function repmat($\mathbf{A}$, m, n) returns a matrix which contains m copies of $\mathbf{A}$ along the row dimension and n copies along the column dimension. Thus if $\mathbf{A}$ is a p by q matrix then repmat($\mathbf{A}$, m, n) returns a mp by nq matrix.

$$\begin{aligned}\mathbf{m}(m).\mathbf{F} \leftarrow\ & \mathbf{m}(m).\mathbf{F} + \mathbf{C_1} \odot (\mathbf{M_x} - \mathbf{m}(m).\mathbf{F}) \\ & + \mathbf{C_2} \odot (\mathbf{M_y} - \mathbf{m}(m).\mathbf{F})\end{aligned} \tag{3}$$

In this work, the cost function is also vectorized. The vectorized version of the cost function returns a scalar when applied to a column vector. When the cost function is applied to a matrix, it is applied column-wise to form a row vector. This avoids using inefficient FOR loops in interpreted array programming languages like MATLAB and Numpy Python library.

The above update moves each female bug strongly towards the dominant male bug controlling its colony and weakly towards random male bugs because values in $\mathbf{C_1}$ are chosen to be significantly larger than $\mathbf{C_2}$. The FSO algorithm was observed to perform significantly better when $\mathbf{C_2}$ was kept smaller than $\mathbf{C_1}$. The attraction of female bugs to random males aids exploration. It improves the diversity of solutions since the convergence of all female bugs to the dominant male's location is reduced. Behaviours I, II, III and V perform exploration and Behaviour IV implements exploitation of good solutions already found.

2.4. Behaviour IV: Attraction of male bugs to fittest female bugs

Each male bug is also attracted to fit females, not in its colony. The movement of male bugs towards the same fittest female bug keeps the entire population confined to a particular geographical location and prevents the swarm from dispersing.

There is no competition for the fittest female bug in the colony of a particular male bug. This is done to prevent all male bugs from being attracted towards the same female bug leading to premature convergence. Preventing competition between different male bugs allows the male bugs and female colonies associated with the male bugs to explore a wider region in the search space without being attracted to the same good locations of the fittest female bugs.

The following update rule is proposed for movement of male bugs towards fittest female bug:

$$\mathbf{m}(m).\mathbf{x} \leftarrow \mathbf{m}(m).\mathbf{x} + \mathbf{c_3} \odot (\mathbf{g} - \mathbf{m}(m).\mathbf{x}) \tag{4}$$

2.5. Behaviour V: Swarm cohesion

The entire swarm moves together as a single unit, and individual bugs do not disperse. Since the swarm moves stochastically as a single unit, individual bugs must copy the direction of motion of others. Eq. (5) is proposed as a model of herd cohesion. Eq. (5) makes each male bug copy the direction of movement of a random male bug towards the fittest female bug.

The male bugs are not attracted to other male bugs but only copy the direction of motion of random male bugs. Thus, a male bug does not simply move towards another male bug's location but copies its direction of motion towards good solutions, thus avoiding premature convergence to a local minimum. In this regard, the FSO is fundamentally different from algorithms like the PSO where all particles move towards a single global best solution.

$$\mathbf{m}(m).\mathbf{x} \leftarrow \mathbf{m}(m).\mathbf{x} + \mathbf{c_4} \odot (\mathbf{g} - \mathbf{m}(b).\mathbf{x}) \tag{5}$$

The FSO is a biologically inspired algorithm based on observation of Firebug reproductive swarming behaviour where individual bugs strive to find best mates for reproduction. This search for best partners for reproduction can naturally be viewed as the search for optimal solutions in a search space. The design of the FSO algorithm was accomplished as follows once the fit between optimization and search for optimal mates was recognized: firstly a systematic study of different Firebug social behaviours was done (Gyuris et al., 2011; Schmuck, 1995; Schofl & Taborsky, 2002; Svádová et al., 2014; Zdarek, 1970; Zemek & Socha, 2010), secondly those behaviours most relevant to optimization were recognized (Section 2, points I to V) and behaviours not dependent on the fitness of individual bugs were ignored, thirdly the simplest possible mathematical model that modelled these behaviours was proposed and finally the parameters were tuned using the CEC2013 benchmark.

The simplest possible update equations for Firebug reproductive swarming behaviour can be mathematically derived as follows:

In Fig. 2, when α varies from 0 to 1, the point $\mathbf{x}$ traces out the line-segment between $\mathbf{x^1}$ and $\mathbf{x^2}$. For $\alpha > 1$, the point $\mathbf{x}$, moves past $\mathbf{x^2}$ on the line connecting $\mathbf{x^1}$ and $\mathbf{x^2}$. Similarly for $\alpha < 0$, the point $\mathbf{x}$, moves farther and farther to the left of $\mathbf{x^1}$ as α is made more and more negative. Thus when α takes all possible real values the point $\mathbf{x}$ traces the entire line joining $\mathbf{x^1}$ and $\mathbf{x^2}$. In summary the point $\mathbf{x}$ is close to $\mathbf{x^1}$ for values of α close to 0 and to $\mathbf{x^2}$ for values of α close to 1. By choosing α to be greater than 1 or less than 0, points beyond $\mathbf{x^1}$ and $\mathbf{x^2}$ can be explored.

The vector $\mathbf{x}$ can be expressed in terms of $\mathbf{x^1}$ and $\mathbf{x^2}$ using the triangular law of vector addition Eq. (6).

$$\begin{aligned}\mathbf{x} &= \mathbf{x^1} + \alpha(\mathbf{x^2} - \mathbf{x^1}) \\ &= (1-\alpha)\mathbf{x^1} + \alpha\mathbf{x^2}\end{aligned} \tag{6}$$

Based on the above discussion it is clear that movement of a male bug (with position $\mathbf{m}(m).\mathbf{x}$) towards the fittest female bug (with position $\mathbf{g}$) can be accomplished using Eq. (7):

$$\mathbf{m}(m).\mathbf{x} \leftarrow \mathbf{m}(m).\mathbf{x} + \alpha(\mathbf{g} - \mathbf{m}(m).\mathbf{x}) \tag{7}$$

However the update equation Eq. (7) while simple and computationally cheap constrains the movement of $\mathbf{m}(m).\mathbf{x}$ to the line joining $\mathbf{m}(m).\mathbf{x}$ and $\mathbf{g}$ reducing exploration of new solutions. To improve exploration and independent exploration in different dimensions the authors proposed that the scalar α be replaced by a column matrix and the scalar-vector multiplication be replaced by the element-wise Hadamard multiplication (leading to update equation shown Eq. (8)).

$$\mathbf{m}(m).\mathbf{x} \leftarrow \mathbf{m}(m).\mathbf{x} + \mathbf{c_4} \odot (\mathbf{g} - \mathbf{m}(b).\mathbf{x}) \tag{8}$$

Similarly the strong movement of female bugs towards its dominant male bug and weak movement towards a random male bug can be expressed as in Eq. (9):

$$\mathbf{m}(m).\mathbf{F} \leftarrow \mathbf{m}(m).\mathbf{F} + \mathbf{C_1} \odot (\mathbf{M_x} - \mathbf{m}(m).\mathbf{F}) + \mathbf{C_2} \odot (\mathbf{M_y} - \mathbf{m}(m).\mathbf{F}) \tag{9}$$

The two terms, $\mathbf{C_1} \odot (\mathbf{M_x} - \mathbf{m}(m).\mathbf{F})$ and $\mathbf{C_2} \odot (\mathbf{M_y} - \mathbf{m}(m).\mathbf{F})$ represent movement towards the dominant male bug and random male

Algorithm 1: Firebug Swarm Optimization (FSO)

for $m \leftarrow 1, N_M$ **do** ▷ Begin initialization
  $\mathbf{m}(m).\mathbf{F} \leftarrow x_{min} + (x_{max} - x_{min}) \times \text{rand}(D, N_F)$;
  $\mathbf{m}(m).\mathbf{F}_f \leftarrow f(\mathbf{m}(m).\mathbf{F})$;
  $[val, ind] \leftarrow \min(\mathbf{m}(m).\mathbf{F}_f)$;
  $\mathbf{m}(m).\mathbf{x}_f \leftarrow val$;
  $\mathbf{m}(m).\mathbf{x} \leftarrow \mathbf{m}(m).\mathbf{F}(:, ind)$;
end for
$[val, ind] \leftarrow \min(\mathbf{m}.\mathbf{x}_f)$;
$\mathbf{g} \leftarrow \mathbf{m}(ind).\mathbf{x}$;
$\mathbf{g}_f \leftarrow val$; ▷ End initialization
for $s \leftarrow 1, S_{1max}$ **do**
  for $l \leftarrow 1, L_1$ **do**
    $a \leftarrow \text{randperm}(N_M)$;
    for $m \leftarrow 1, N_M$ **do**
      $\mathbf{C_1} \leftarrow -0.75 + 2.255 \times \text{rand}(D, N_F)$;
      $\mathbf{C_2} \leftarrow -0.25 + 1.302 \times \text{rand}(D, N_F)$;
      $b \leftarrow a(m)$;
      $\mathbf{M_x} \leftarrow \text{repmat}(\mathbf{m}(m).\mathbf{x}, 1, N_F)$;
      $\mathbf{M_y} \leftarrow \text{repmat}(\mathbf{m}(b).\mathbf{x}, 1, N_F)$;
      $\mathbf{m}(m).\mathbf{F} \leftarrow \mathbf{m}(m).\mathbf{F} + \mathbf{C_1} \odot (\mathbf{M_x} - \mathbf{m}(m).\mathbf{F}) + \mathbf{C_2} \odot (\mathbf{M_y} - \mathbf{m}(m).\mathbf{F})$;
      $\mathbf{m}(m).\mathbf{F}_f \leftarrow f(\mathbf{m}(m).\mathbf{F})$;
      $[val, ind] \leftarrow \min(\mathbf{m}(m).\mathbf{F}_f)$;
      $\mathbf{m}(m).\mathbf{x}_f \leftarrow val$;
      $\mathbf{m}(m).\mathbf{x} \leftarrow \mathbf{m}(m).\mathbf{F}(:, ind)$;
      $[val, ind] \leftarrow \min(\mathbf{m}.\mathbf{x}_f)$;
      if $val < \mathbf{g}_f$ **then**
        $\mathbf{g} \leftarrow \mathbf{m}(m).\mathbf{F}(:, ind)$;
        $\mathbf{g}_f \leftarrow val$;
      end if
    end for
  end for
  for $l \leftarrow 1, L_2$ **do**
    for $m \leftarrow 1, N_M$ **do**
      $\mathbf{c_3} \leftarrow -0.5 + 2.7 \times \text{rand}(D, 1)$;
      $\mathbf{m}(m).\mathbf{x} \leftarrow \mathbf{m}(m).\mathbf{x} + \mathbf{c_3} \odot (\mathbf{g} - \mathbf{m}(m).\mathbf{x})$;
      $\mathbf{m}(m).\mathbf{x}_f \leftarrow f(\mathbf{m}(m).\mathbf{x})$;
    end for
    for $m \leftarrow 1, N_M$ **do**
      if $\mathbf{m}(m).\mathbf{x}_f < \mathbf{g}_f$ **then**
        $\mathbf{g} \leftarrow \mathbf{m}(m).\mathbf{x}$;
        $\mathbf{g}_f \leftarrow \mathbf{m}(m).\mathbf{x}_f$;
      end if
    end for
  end for
end for
for $s \leftarrow 1, S_{2max}$ **do**
  $a \leftarrow \text{randperm}(N_M)$;
  for $m \leftarrow 1, N_M$ **do**
    $\mathbf{c_4} \leftarrow 1.4 \times \text{rand}(D, 1)$;
    $b \leftarrow a(m)$;
    $\mathbf{m}(m).\mathbf{x} \leftarrow \mathbf{m}(m).\mathbf{x} + \mathbf{c_4} \odot (\mathbf{g} - \mathbf{m}(b).\mathbf{x})$;
    $\mathbf{m}(m).\mathbf{x}_f \leftarrow f(\mathbf{m}(m).\mathbf{x})$;
    if $\mathbf{m}(m).\mathbf{x}_f < \mathbf{g}_f$ **then**
      $\mathbf{g} \leftarrow \mathbf{m}(m).\mathbf{x}$;
      $\mathbf{g}_f \leftarrow \mathbf{m}(m).\mathbf{x}_f$;
    end if
  end for
end for

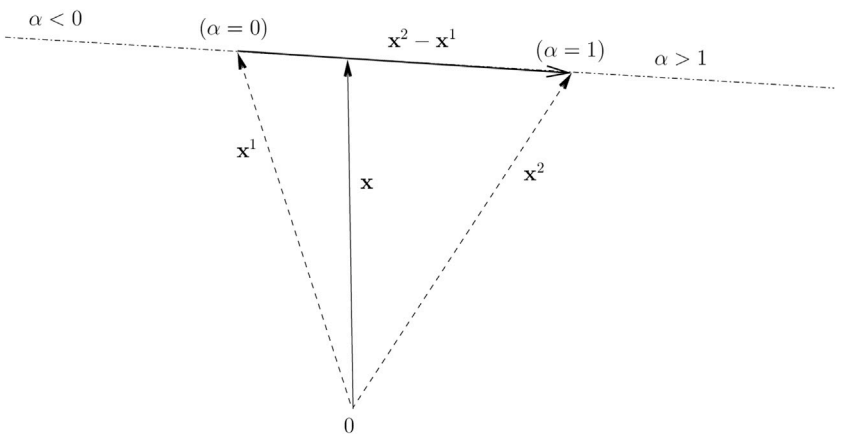


Fig. 2. Modelling the movement of Firebugs. $\mathbf{x}^1$ is the position vector of the bug 1 and $\mathbf{x}^2$ is the position vector of bug 2. The position vector $\mathbf{x}$ can be made to move towards location $\mathbf{x}^2$ by choosing α closer and closer to 1.

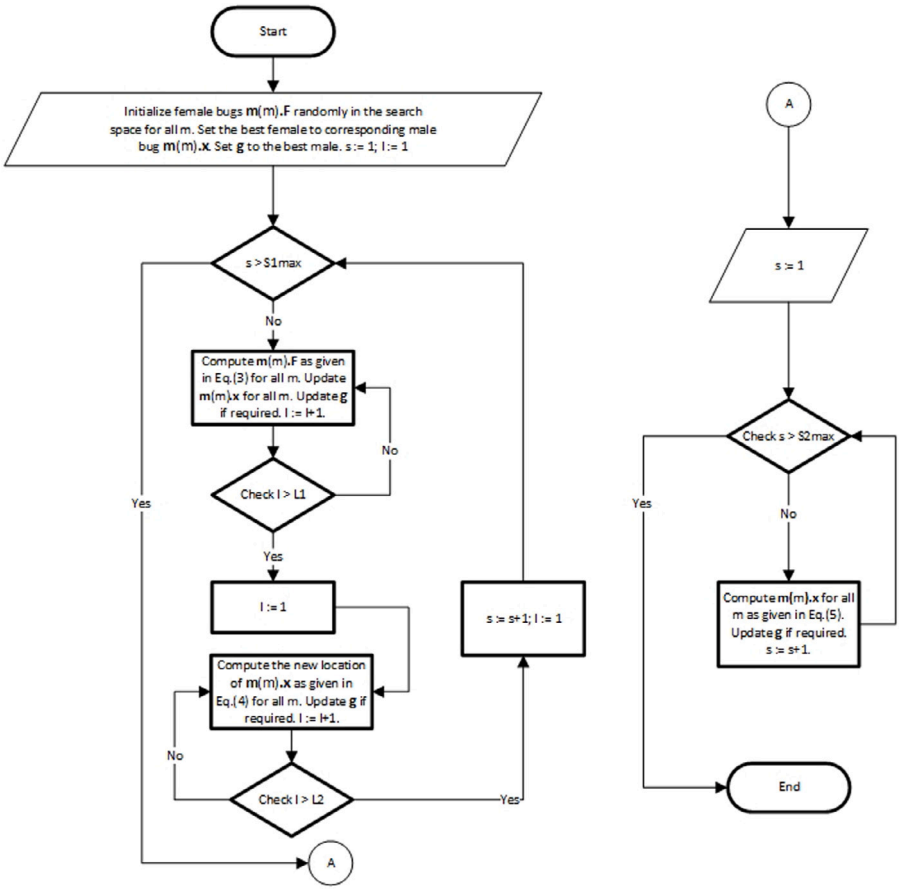


Fig. 3. FSO flowchart.

bug respectively. Matrices $\mathbf{C_1}$ and $\mathbf{C_2}$ determine the strength of the attraction towards the dominant male bug and random male bug. However, update Eq. (9) differs from Eq. (8) in that $\mathbf{C_1}$ and $\mathbf{C_2}$ now matrices instead of column vectors since $\mathbf{m}(m).\mathbf{F}$ is a matrix containing all the female bugs associated with male bug m as columns of $\mathbf{m}(m).\mathbf{F}$. Thus the locations of all female bugs associated with a male bug are updated simultaneously using the efficient Hadamard matrix multiplication operation.

The complete FSO algorithm is presented in Algorithm 1 and flowchart is given in Fig. 3.

3. Experiments and summary of results

In the following, the FSO algorithm is compared to 17 popular heuristic optimization algorithms. A direct comparison of 17 different algorithms on all possible benchmark problems over different dimensions and a large number of independent runs is not feasible; so the FSO is compared to the three best performing algorithms. The three best performing algorithms chosen for comparison with the FSO are already known to outperform 14 state-of-the-art optimization algorithms (Kocer, 2016; Li, Zheng, & Tan, 2016; Nasir, Das, Maity, Halder, & Suganthan, 2012).

Table 1
Parameters settings for FSO.

D	N_M	N_F	S_{1max}	L_1	L_2	S_{2max}	MaxFE
30	20	5	82	34	11	153	300000
50	20	5	137	34	11	198	500000
100	20	5	275	34	11	220	1000000

Performance on the CEC 2013 suite of benchmark minimization problems is used as the metric for comparing different optimization algorithms because of its universal adoption in optimization literature (Li et al., 2016; Liang et al., 2013). The CEC 2013 test suite has 5 unimodal, 15 basic multimodal, and 8 composite test functions. Algorithms that perform exploitation and local search will perform well on unimodal functions. On the other hand, algorithms that explore the search space effectively perform well on multimodal and high dimensional problems.

Since very efficient convex optimization and local search based techniques such as Interior Point Methods (Dikin, 1967) and Nelder–Mead Simplex (Nelder & Mead, 1965) exist for the solution of unimodal functions, performance on multimodal and high dimensional problems is of more relevance when comparing different global optimization algorithms.

3.1. Parameter settings for FSO

The FSO parameter settings that were empirically observed to provide good performance on CEC 2013 is presented in Table 1.

Guidelines for tuning user-defined parameters in the FSO are provided below. Competing algorithms like DNLPSO and ABCBB maintain a population size of 100. A population size of 100 is also used by the 17 algorithms taken for comparison from optimization literature. The GFWA employs a dynamically varying population size between 1 and 200 and hence also uses an average population size of 100. The FSO also uses a population size of 100 in order to ensure a fair comparison of different algorithms.

The product $N_M N_F$ is the population size of FSO. N_M and N_F, the number of male and female bugs are chosen to be 20 and 5, respectively, to achieve a population size of 100, as discussed above. Since female bugs are attracted to their respective dominant male bugs the number of male bugs is chosen to be larger to favour exploring new solutions. Increasing the relative number of male bugs increases exploration and hence the relative number of male bugs can be increased when applying the FSO to highly multimodal optimization problems.

L_1 increases exploration and L_2 increases exploitation. The values of L_1 and L_2 given in Table 1 were empirically found to work well for the entire CEC13 benchmark suite. Increasing L_1 and L_2 beyond the values given in Table 1 did not result in appreciable improvements in accuracy on the CEC 2013 benchmark.

The values of S_{1max} and S_{2max} are fixed by the competition criteria that the total function evaluations should not exceed $10000D$. S_{1max} affects both exploration and exploitation, while S_{2max} effects only exploitation. Table 1 recommends values of S_{1max} and S_{2max} which were empirically observed to strike a good balance between exploration and exploitation. To perform more exploration in high dimensional multimodal landscape, S_{1max} is increased relative to S_{2max} with increasing dimensions and problem complexity. If higher accuracy is desired, then both S_{1max} and S_{2max} can be simultaneously increased at the cost of higher computational time.

The values of $\mathbf{C_1}$, $\mathbf{C_2}$, $\mathbf{c_3}$ and $\mathbf{c_4}$ suggested in this paper were observed to work uniformly well for all 84 benchmark problems (28 test functions with 30, 50 and 100 dimensions). The constants $\mathbf{C_1}$, $\mathbf{C_2}$, $\mathbf{c_3}$ and $\mathbf{c_4}$ play a role similar to the constants c_1 and c_2 (usually set to 2.5) in the PSO and do not have to be adjusted for different problems. $\mathbf{C_1}$ determines population cohesion and was chosen higher than $\mathbf{C_2}$ to avoid a pure random search.

3.2. Performance on CEC 2013

The test problems in CEC 2013 include shifted, rotated, noisy, highly multimodal, discontinuous, non-differentiable, and high dimensional problems that simulate the challenges posed by real world problems. These similarities between real world problems and CEC 2013 benchmark suite have led to its universal adoption in optimization literature. Hence this paper follows this convention for testing the FSO algorithm.

The No Free Lunch (NFL) theorem states that all optimization algorithms' performance is the same when averaged over all possible test functions (Wolpert & Macready, 1997). For example, deterministic derivative based techniques like Gradient Descent and Newton's method outperform stochastic algorithms like PSO on differentiable and convex problems (like the sphere function) while failing to converge on multimodal problems (like Rastrigin's function). Hence an appropriate subset of problems like CEC 2013 must be chosen instead of particular real world problems. The CEC 2013 was chosen in this paper since it enables a comparison with a majority of algorithms proposed in recent literature.

In this work benchmark problems of dimensions 30, 50 and 100 are considered. Algorithms that demonstrate graceful degradation of performance with increasing dimension are of particular interest for large scale global optimization. Since the final solution computed by stochastic optimization algorithms is a random variable dependent on all random numbers generated during the search process, average performance over large number of independent runs must be considered.

In the following, the FSO algorithm is compared with 3 state-of-the-art algorithms namely Bollinger Bands boosted Artificial Bee Colony Algorithm (ABCBB), Guided Sparks Fireworks Algorithm (GFWA) and Dynamic Neighbourhood Learning based Particle Swarm Optimization (DNLPSO) Algorithm.

3.2.1. Artificial bee colony bollinger bands algorithm

ABCBB was proposed by B. Kocer and improved on the performance of the Artificial Bee Colony algorithm by using a second parameter update using Bollinger Bands (a technique used to predict stock prices) (Kocer, 2016).

ABCBB (Kocer, 2016) was shown to perform better than the recently proposed metaheuristic algorithms such as:

1. Artificial Algae Algorithm (AAA) (Uymaza et al., 2015), and
2. Galactic Swarm Optimization (GSO) (Muthiah-Nakarajan & Noel, 2016).

3.2.2. Fire works optimization algorithm

The working of Fireworks algorithm was greatly improved by guiding sparks which served as the motivation for developing GFWA (Li et al., 2016). The GFWA algorithm uses the concept of "guided sparks" to improve exploitation of information contained in the current global best solution estimate. Results reported in (Li et al., 2016) show that GFWA outperforms a number of leading algorithms like:

1. Artificial Bee Colony Algorithm (ABC) (Karaboga, 2005),
2. Differential Evolution (DE) (Storn & Price, 1997),
3. Particle Swarm Optimization (PSO) (Eberhart & Kennedy, 1995),
4. Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen, 2006),
5. Enhanced Fireworks Algorithm (EFWA) (Zhang, Zheng, Zhang, & Chen, 2015),
6. Adaptive Fireworks Algorithm (AFWA) (Li, Zheng, & Tan, 2014), and
7. Dynamic Search Fireworks Algorithm (dynFWA) (Zheng, Janecek, Li, & Tan, 2014).

Table 2
Performance of FSO on 30 dimensional CEC 2013 problems.

f	ABCBB		GFWA		DNLPSO		FSO	
f_1	$\mu = 0$	$\sigma = 0$	0	0	8.727732E+02	1.556358E+03	0	0
	$b = 0$	$w = 0$	0	0	0	6.404018E+03	0	0
	m_d =0(1) ↕		0(1) ↕		1.534596E+01(4) ↑		0(1)	
f_2	8.082057E+06	2.392845E+06	7.081762E+05	2.846832E+05	3.108382E+07	2.863738E+07	6.488010E+05	2.361703E+05
	3.372181E+06	1.218166E+07	2.702363E+05	1.739554E+06	1.761976E+06	1.443589E+08	2.679031E+05	1.486321E+06
	7.676477E+06(3) ↑		6.651806E+05(1) ↕		2.365956E+07(4) ↑		5.896009E+05(1)	
f_3	4.284129E+08	3.452240E+08	4.876555E+07	6.534819E+07	1.980561E+10	5.869417E+10	8.708955E+07	1.503723E+08
	3.103086E+07	1.640546E+09	3.235732E+05	2.959982E+08	4.999926E+06	3.795091E+11	1.436169E+06	8.122822E+08
	3.357046E+08(3) ↑		2.266653E+07(1) ↓		1.829604E+09(4) ↑		3.042134E+07(2)	
f_4	9.233495E+04	1.768779E+04	3.642753E−05	4.857635E−05	1.312262E+04	1.224761E+04	8.216250E+03	2.630010E+03
	3.749918E+04	1.236390E+05	1.653667E−06	1.915295E−04	1.112740E+03	5.770326E+04	3.749463E+03	1.771142E+04
	9.486690E+04(4) ↑		1.578282E−05(1) ↓		7.811557E+03(2) ↕		7.757861E+03(2)	
f_5	0	0	1.565999E−03	2.719572E−04	1.566868E+02	3.666370E+02	0	0
	0	0	9.636658E−04	2.083723E−03	0	1.807169E+03	0	0
	0(1) ↕		1.571684E−03(3) ↑		1.872922E−03(3) ↑		0(1)	
f_6	1.637203E+01	4.891733E+00	3.623759E+01	2.443962E+01	2.627231E+02	2.290661E+02	2.629874E+01	2.505534E+01
	5.993240E+00	4.420166E+01	1.349021E+00	9.223178E+01	7.247854E+01	9.084320E+02	2.444443E−01	7.552134E+01
	1.658267E+01(2) ↑		2.845700E+01(3) ↑		1.583214E+02(4) ↑		1.510674E+01(1)	
f_7	1.304037E+02	1.750721E+01	7.511758E+01	2.866110E+01	7.801592E+01	5.678331E+01	2.712994E+01	1.120838E+01
	1.005367E+02	1.696549E+02	3.464998E+01	1.625622E+02	1.702344E+01	2.471126E+02	1.026193E+01	6.727459E+01
	1.278782E+02(4) ↑		7.023116E+01(2) ↑		5.813000E+01(2) ↑		2.600742E+01(1)	
f_8	2.094309E+01	4.888921E−02	2.093298E+01	8.630091E−02	2.093918E+01	5.029552E−02	2.094535E+01	5.527199E−02
	2.081235E+01	2.101195E+01	2.071053E+01	2.109359E+01	2.081617E+01	2.103225E+01	2.077905E+01	2.103771E+01
	2.095504E+01(1) ↕		2.094787E+01(1) ↕		2.094148E+01(1) ↕		2.095076E+01(1)	
f_9	2.901183E+01	1.999077E+00	1.870724E+01	4.013160E+00	2.484625E+01	6.427955E+00	1.823477E+01	1.868398E+00
	2.173609E+01	3.193452E+01	7.566115E+00	2.899348E+01	1.271400E+01	3.836414E+01	1.162535E+01	2.155604E+01
	2.933777E+01(4) ↑		1.870844E+01(1) ↕		2.563511E+01(3) ↑		1.831769E+01(1)	
f_{10}	3.172318E−01	1.384089E−01	6.378213E−02	4.640824E−02	3.740955E+02	5.726566E+02	3.008835E−01	1.644347E−01
	1.265051E−01	9.154235E−01	1.232099E−02	2.441285E−01	4.930612E−02	2.644784E+03	7.634539E−02	1.065062E+00
	2.731289E−01(2) ↕		5.177968E−02(1) ↓		1.394843E+02(4) ↑		2.662403E−01(2)	
f_{11}	0	0	8.092304E+01	2.646397E+01	7.522795E+01	6.648764E+01	2.393184E+01	8.014223E+00
	0	0	3.780838E+01	1.382985E+02	1.711317E+01	3.154327E+02	1.193950E+01	4.278316E+01
	0(1) ↓		7.362681E+01(4) ↑		4.875294E+01(3) ↑		2.288403E+01(2)	
f_{12}	2.209563E+02	4.692358E+01	8.362205E+01	2.553247E+01	8.102667E+01	8.019312E+01	6.105330E+01	1.714135E+01
	1.209497E+02	3.371460E+02	4.676303E+01	1.970006E+02	1.886836E+01	3.524956E+02	2.416743E+01	1.092252E+02
	2.294367E+02(4) ↑		7.661169E+01(3) ↑		5.379324E+01(1) ↕		6.011468E+01(1)	
f_{13}	2.795160E+02	3.998547E+01	1.653644E+02	4.348298E+01	1.580381E+02	7.008590E+01	1.109414E+02	2.801393E+01
	1.782371E+02	3.713978E+02	6.766704E+01	2.719134E+02	5.903763E+01	3.880642E+02	5.797514E+01	1.603846E+02
	2.837499E+02(4) ↑		1.683136E+02(2) ↑		1.399033E+02(2) ↑		1.108585E+02(1)	
f_{14}	5.272490E−01	6.162382E−01	3.188486E+03	6.773052E+02	2.720482E+03	1.566318E+03	8.536505E+02	3.122899E+02
	1.669912E−01	3.616963E+00	1.850038E+03	5.086147E+03	7.459074E+02	6.349623E+03	2.831772E+02	1.586616E+03
	2.547466E−01(1) ↓		3.129357E+03(4) ↑		2.373900E+03(3) ↑		8.177896E+02(2)	
f_{15}	4.213588E+03	3.907525E+02	3.552100E+03	5.811334E+02	5.685082E+03	1.371576E+03	4.309465E+03	7.244616E+02
	3.155274E+03	5.026955E+03	2.319642E+03	4.673958E+03	2.661948E+03	7.317769E+03	3.058741E+03	6.196459E+03
	4.210781E+03(2) ↕		3.556569E+03(1) ↓		6.015865E+03(4) ↑		4.231767E+03(2)	
f_{16}	1.590730E+00	3.343974E−01	1.328275E−01	1.337919E−01	2.173863E+00	6.219405E−01	1.620179E+00	3.201101E−01
	9.343832E−01	2.404819E+00	1.146782E−02	6.716007E−01	2.859186E−01	2.922474E+00	9.923295E−01	2.252025E+00
	1.553258E+00(2) ↕		8.585825E−02(1) ↓		2.338687E+00(4) ↑		1.645106E+00(2)	
f_{17}	3.043856E+01	7.167709E−03	9.123715E+01	2.718261E+01	1.017676E+02	4.093020E+01	6.788991E+01	9.460783E+00
	3.043375E+01	3.046848E+01	2.507416E+01	1.497331E+02	5.071640E+01	1.794877E+02	5.160588E+01	8.768419E+01
	3.043489E+01(1) ↓		9.148177E+01(3) ↑		9.692300E+01(3) ↑		6.669625E+01(2)	
f_{18}	2.461170E+02	3.309405E+01	8.528170E+01	2.474663E+01	1.440475E+02	4.688168E+01	9.340066E+01	1.610561E+01
	1.832467E+02	3.037023E+02	3.918993E+01	1.253986E+02	6.172787E+01	2.858350E+02	6.242764E+01	1.360938E+02
	2.475047E+02(4) ↑		8.661972E+01(1) ↕		1.452597E+02(3) ↑		9.190487E+01(1)	
f_{19}	3.274207E−01	1.165706E−01	5.130373E+00	1.374668E+00	1.305309E+02	3.738487E+02	2.944084E+00	1.017447E+00
	1.478098E−01	7.292380E−01	2.752105E+00	1.040734E+01	3.526952E+00	2.198358E+03	1.015371E+00	7.462665E+00
	3.134392E−01(1) ↓		4.926803E+00(3) ↑		1.700721E+01(4) ↑		2.841039E+00(2)	
f_{20}	1.494633E+01	1.861844E−01	1.328511E+01	1.234744E+00	1.175645E+01	9.062629E−01	1.294006E+01	1.781199E+00
	1.388910E+01	1.500000E+01	1.108827E+01	1.500000E+01	1.019296E+01	1.457866E+01	9.308821E+00	1.500000E+01
	1.500000E+01(4) ↑		1.327894E+01(2) ↕		1.159210E+01(1) ↓		1.216817E+01(2)	
f_{21}	1.970413E+02	4.791089E+01	2.878232E+02	9.371149E+01	4.724476E+02	2.595300E+02	3.180888E+02	8.131071E+01
	1.014536E+02	3.000000E+02	2.000000E+02	4.435441E+02	2.000000E+02	1.191657E+03	1.000000E+02	4.435441E+02
	2.000080E+02(1) ↓		3.000000E+02(2) ↕		3.197644E+02(4) ↑		3.000000E+02(2)	
f_{22}	3.921648E+01	3.900913E+01	4.330029E+03	1.153264E+03	2.436811E+03	1.636422E+03	6.427305E+02	2.964569E+02
	8.983697E+00	1.140562E+02	1.917449E+03	7.119182E+03	6.034466E+02	6.412252E+03	1.540033E+02	1.477079E+03
	1.927839E+01(1) ↓		4.455909E+03(4) ↑		1.785421E+03(3) ↑		6.047571E+02(2)	

(*continued on next page*)

Table 2 (*continued*).

f	ABCBB		GFWA		DNLPSO		FSO	
f_{23}	5.297154E+03	5.688543E+02	4.477303E+03	6.591973E+02	5.158510E+03	1.644893E+03	4.531040E+03	8.785707E+02
	4.054217E+03	6.511831E+03	3.034407E+03	5.839563E+03	2.186146E+03	8.301857E+03	2.634661E+03	6.233682E+03
	5.384834E+03(3) ↑		4.558323E+03(1) ↕		5.114918E+03(3) ↑		4.311993E+03(1)	
f_{24}	2.911580E+02	7.008438E+00	2.567015E+02	2.050658E+01	2.568035E+02	2.545383E+01	2.368372E+02	7.904390E+00
	2.677569E+02	3.067365E+02	2.000000E+02	3.006312E+02	2.182456E+02	3.121112E+02	2.208342E+02	2.572783E+02
	2.922449E+02(4) ↑		2.584566E+02(2) ↑		2.518974E+02(2) ↑		2.358627E+02(1)	
f_{25}	3.093526E+02	8.119258E+00	2.884108E+02	1.297957E+01	3.155100E+02	2.216175E+01	2.707430E+02	6.164625E+00
	2.842179E+02	3.268242E+02	2.561804E+02	3.199335E+02	2.363432E+02	3.542057E+02	2.583930E+02	2.823265E+02
	3.093554E+02(3) ↑		2.880299E+02(2) ↑		3.117359E+02(3) ↑		2.713894E+02(1)	
f_{26}	2.009013E+02	4.707815E−01	2.055428E+02	2.745551E+01	2.132200E+02	4.114127E+01	2.791367E+02	6.429477E+01
	2.003056E+02	2.032877E+02	2.000131E+02	3.421253E+02	2.000106E+02	3.902230E+02	2.000243E+02	3.418713E+02
	2.008081E+02(2) ↓		2.000485E+02(1) ↓		2.012268E+02(2) ↓		3.242400E+02(4)	
f_{27}	7.489477E+02	3.778884E+02	8.318872E+02	1.335788E+02	7.866165E+02	2.577599E+02	6.970324E+02	9.297221E+01
	4.000009E+02	1.254481E+03	4.298739E+02	1.106751E+03	3.911165E+02	1.366919E+03	4.761512E+02	9.066450E+02
	4.024006E+02(1) ↕		8.193173E+02(1) ↕		7.309538E+02(1) ↕		6.972903E+02(1)	
f_{28}	2.718976E+02	6.503624E+01	3.790942E+02	2.899969E+02	9.582459E+02	7.768575E+02	3.222869E+02	1.898025E+02
	1.034576E+02	3.148873E+02	1.000000E+02	1.402861E+03	3.000000E+02	3.654245E+03	1.000000E+02	1.636631E+03
	3.000000E+02(3) ↑		3.000000E+02(2) ↑		6.024041E+02(4) ↑		3.000000E+02(1)	
A.R.	2.3571		1.9286		2.8929		1.5357	

3.2.3. State-of-the-art PSOs

DNLPSO is a highly competitive PSO algorithm, based on learning from particles in dynamically varying neighbourhood (Nasir et al., 2012). It is a modification of the Comprehensive Learning Particle Swarm Optimizer (CLPSO) strategy (Liang, Qin, Suganthan, & Baskar, 2006). In the CLPSO algorithm, the particles move towards a randomly chosen personal best solution instead of the global best solution: this significantly improves exploration and reduces premature convergence to a local minimum. DNLPSO performs better than:

1. Local PSO (PSOLocal) (Li & Engelbrecht, 2007),
2. Unified PSO (UPSO) (Parsopoulos & Vrahatis, 2004),
3. Fully Informed PSO (FIPS) (Mendes, Kennedy, & Neves, 2004),
4. Dynamic-Multi-Swarm PSO (DMSPSO) (Liang & Suganthan, 2005), and
5. Comprehensive Learning PSO (CLPSO) (Liang et al., 2006)

Since DNLPSO has also been shown to outperform many popular algorithms like PSO and CLPSO, it is taken for comparison with the FSO algorithm.

The comparison of different algorithms was made in accordance with the CEC 2013 (Liang et al., 2013) competition guidelines. In recent years novel optimization algorithms that use Fuzzy rules to intelligently adapt user-defined parameters were proposed (Bernal, Castillo, Soria, & Valdez, 2020; Olivas, Valdez, Castillo, Gonzalez, & Melin, 2017; Olivas, Valdez, Castillo, & Melin, 2016; Olivas, Valdez, Melin, Sombra, & Castillo, 2019).

Since only the number of function evaluations needed on average to achieve a certain accuracy is considered as a performance metric in CEC 2013, the computational cost associated with Fuzzy inference is ignored. Hence a comparison with algorithms that adaptively tune their parameters using a Fuzzy inference system will not be a fair comparison since the extra computational cost associated with the Fuzzy inference system is completely ignored.

However, intelligently adapting the parameters of the FSO using a Fuzzy inference system can provide significant performance benefits and is of great interest. In future works, a fuzzy FSO algorithm where the parameters of the FSO are intelligently tuned using a Fuzzy inference system can be explored. Further, an extensive comparison of Fuzzy GSO, Fuzzy PSO, Fuzzy FSO and other Fuzzy logic based optimization algorithms can be attempted.

All algorithms taken for comparison with the FSO were tuned by their respective authors to be competitive on CEC 2013 or CEC 2015 except DLNPSO which was tuned by its authors to be competitive on CEC 2005. For GFWA parameter settings that were shown to be very competitive on CEC 2013 was used. For ABCBB parameter settings that performed well on the CEC 2015 suite which is an advanced version of CEC 2013 was used. In every case best known parameter settings from literature and source code provided by the authors of the competing algorithms were used to ensure a fair comparison.

Average performance on 51 independent runs was considered. Since the number of function evaluations required to achieve a predefined accuracy is universally considered as a measure of the computational complexity of optimization algorithms, each algorithm was allowed to run till it exceeded a maximum function evaluation limit. Since an accuracy exceeding 10^{-8} is seldom required in engineering applications, the algorithm is said to converge perfectly and a convergence error of 0 is reported if the final solution is within 10^{-8} of the actual minimum value (Liang et al., 2013).

Since a more accurate answer can always be computed with more computational effort (measured in terms of the number of function evaluations), the allowable maximum number of function evaluations must be kept fixed at the same value for all algorithms to ensure a fair comparison. The parameter S_{1max} in the FSO controls the maximum allowable number of function evaluations and is varied with dimension to match the maximum allowable number of function evaluations.

Tables 2–4 show the performance of the competing algorithms on CEC 2013 benchmark functions for 30, 50 and 100 dimensions. Each algorithm was run for $D \times 10000$ function evaluations in accordance with CEC 2013 competition guidelines.

Each entry in Tables 2–4 contains the μ (mean), σ (standard deviation), b (best), w (worst), m_d (median) values of the algorithms for 51 independent trials (Liang et al., 2013). Performance over 51 independent trials is considered to enable the median to be easily computed as the 26 value after sorting.

Since the mean is highly sensitive to outliers, it is more appropriate to consider the median performance to identify algorithms that perform inconsistently. Tables 2–4 provides a comparison of the median best value of different algorithms using the non-parametric Wilcoxon Rank-Sum (Conover, 1980) statistical test.

The null hypothesis ($h = 0$) is the proposition that the medians are not different with 95% confidence and the alternate hypothesis ($h = 1$) is the proposition that the medians are different with 95% confidence. The result of the Rank-Sum Test is highlighted as ↓, ↑ or ↕. ↓ (↑) indicates that the median of the associated algorithm performed superior (inferior) with 95% confidence in comparison to the FSO algorithm. ↕ symbol indicates that the medians are not different with 95% confidence. Each algorithm is assigned a rank, which is indicated next to the median in the tables. The final row of each table provides the Average Rank (A.R.).

Table 3
Performance of FSO on 50 dimensional CEC 2013 problems.

f	ABCBB		GFWA		DNLPSO		FSO	
f_1	$\mu = 0$	$\sigma = 0$	0	0	3.159725E+03	6.461801E+03	0	0
	$b = 0$	$w = 0$	0	0	0	2.448061E+04	0	0
	m_d =0(1) ↕		0(1) ↕		4.845931E+01(4) ↑		0(1)	
f_2	1.206186E+07	3.266898E+06	1.544798E+06	5.634614E+05	7.231399E+07	9.144095E+07	8.361949E+05	2.587998E+05
	5.498225E+06	2.137639E+07	5.898464E+05	2.784480E+06	9.712904E+05	4.911370E+08	4.064620E+05	1.611520E+06
	1.185751E+07(3) ↑		1.439438E+06(2) ↑		4.282928E+07(4) ↑		7.656333E+05(1)	
f_3	1.766739E+09	1.365930E+09	1.202956E+08	1.416665E+08	2.625060E+10	5.090811E+10	2.773580E+08	3.772773E+08
	1.565654E+08	8.321115E+09	2.292556E+06	8.697728E+08	5.728564E+07	2.835914E+11	8.670655E+06	2.039714E+09
	1.545039E+09(3) ↑		7.544388E+07(1) ↓		4.420439E+09(4) ↑		1.329993E+08(2)	
f_4	1.183908E+05	1.786795E+04	8.956374E−07	7.980423E−07	2.142774E+04	1.938123E+04	2.569983E+04	5.579415E+03
	6.696296E+04	1.560749E+05	9.256541E−08	3.127395E−06	3.152101E+03	9.692119E+04	1.553197E+04	3.714328E+04
	1.200650E+05(4) ↑		5.941254E−07(1) ↓		1.354271E+04(2) ↓		2.560632E+04(3)	
f_5	0	0	1.638245E−03	1.740886E−04	1.308009E+02	3.283484E+02	0	0
	0	0	1.283225E−03	2.008870E−03	0	1.555940E+03	0	0
	0(1) ↕		1.659446E−03 (3)↑		2.226026E−03(3) ↑		0(1)	
f_6	4.221466E+01	4.814465E+00	4.355987E+01	8.012730E−01	3.165301E+02	3.052478E+02	5.118921E+01	1.836852E+01
	2.301001E+01	4.638988E+01	4.344757E+01	4.916990E+01	6.668062E+01	1.302846E+03	2.035588E+01	9.909488E+01
	4.353590E+01(1) ↕		4.344765E+01(1) ↕		1.636348E+02(4) ↑		4.345098E+01(1)	
f_7	1.679750E+02	2.317012E+01	1.015758E+02	2.372346E+01	6.344341E+01	4.144564E+01	3.193059E+01	9.768184E+00
	1.150282E+02	2.272800E+02	6.042363E+01	1.736100E+02	2.278895E+01	2.477343E+02	1.586739E+01	7.171614E+01
	1.697686E+02(4) ↑		9.769620E+01(3) ↑		4.947238E+01(2) ↑		3.276066E+01(1)	
f_8	2.113525E+01	3.012707E−02	2.113986E+01	5.616403E−02	2.112744E+01	3.777320E−02	2.113166E+01	3.251069E−02
	2.103143E+01	2.118647E+01	2.102613E+01	2.123015E+01	2.099345E+01	2.118729E+01	2.106930E+01	2.119566E+01
	2.114130E+01(1) ↕		2.114848E+01(1) ↕		2.113165E+01(1) ↕		2.113654E+01(1)	
f_9	5.804654E+01	3.052088E+00	3.762214E+01	8.446621E+00	4.490404E+01	1.269123E+01	3.532066E+01	3.125135E+00
	5.138112E+01	6.408160E+01	2.185926E+01	6.294196E+01	2.793506E+01	7.283914E+01	2.822752E+01	4.098420E+01
	5.886200E+01(4) ↑		3.633725E+01(1) ↕		4.212478E+01(3) ↑		3.601679E+01(1)	
f_{10}	4.983809E−01	3.095836E−01	4.296041E−02	3.138952E−02	8.054457E+02	1.142270E+03	1.866288E−01	1.039401E−01
	1.361448E−01	1.249315E+00	7.396040E−03	1.675592E−01	2.459044E−02	3.780937E+03	3.202082E−02	5.269185E−01
	3.859434E−01(3) ↑		2.958663E−02(1) ↓		1.263541E+02(4) ↑		1.724499E−01(2)	
f_{11}	0	0	1.262032E+02	4.746700E+01	1.867401E+02	1.495108E+02	7.826996E+01	2.577703E+01
	0	0	4.576809E+01	2.547077E+02	3.820848E+01	5.682360E+02	4.576807E+01	1.631724E+02
	0(1) ↓		1.193946E+02(3) ↑		1.184762E+02(3) ↑		7.064198E+01(2)	
f_{12}	5.930511E+02	7.434519E+01	1.739523E+02	6.719887E+01	2.253094E+02	1.983144E+02	1.304812E+02	2.840475E+01
	4.242659E+02	7.578594E+02	7.263193E+01	4.848919E+02	5.074287E+01	7.427827E+02	7.266246E+01	1.835549E+02
	5.943889E+02(4) ↑		1.651626E+02(3) ↑		1.263595E+02(1) ↕		1.307880E+02(1)	
f_{13}	6.549249E+02	8.401655E+01	3.180513E+02	8.539405E+01	3.078704E+02	1.207017E+02	2.219649E+02	4.587859E+01
	4.853715E+02	8.394137E+02	1.660568E+02	6.314026E+02	1.248083E+02	6.610623E+02	1.161006E+02	3.343229E+02
	6.546689E+02(4) ↑		3.161242E+02(2) ↑		2.870757E+02(2) ↑		2.152033E+02(1)	
f_{14}	1.770912E+00	8.207989E−01	5.087340E+03	1.119591E+03	5.873974E+03	3.598884E+03	1.620927E+03	6.176288E+02
	2.874635E−01	4.319734E+00	2.110408E+03	7.752515E+03	1.043258E+03	1.321301E+04	4.666468E+02	3.040425E+03
	1.632867E+00(1) ↓		5.036874E+03(3) ↑		4.319254E+03(3) ↑		1.532500E+03(2)	
f_{15}	8.220510E+03	6.054259E+02	3.552100E+03	5.811334E+02	1.093091E+04	3.256832E+03	8.934480E+03	1.066719E+03
	6.279885E+03	9.383413E+03	2.319642E+03	4.673958eE03	5.319015E+03	1.489645E+04	4.993458E+03	1.136683E+04
	8.178009E+03(1) ↓		3.556569E+03(3) ↑		1.268973E+04(3) ↑		8.866041E+03(2)	
f_{16}	1.969436E+00	3.364035E−01	6.915462E−02	6.261179E−02	3.216287E+00	5.580489E−01	3.068442E+00	4.189680E−01
	1.101531E+00	2.662009E+00	9.876419E−03	3.152696E−01	7.357556E−01	3.747695E+00	2.089822E+00	3.733470E+00
	1.990037E+00(2) ↓		5.363339E−02(1) ↓		3.355730E+00(4) ↑		3.154532E+00(3)	
f_{17}	5.079441E+01	1.075606E−02	1.322921E+02	2.922723E+01	2.243479E+02	1.201503E+02	1.232843E+02	2.040032E+01
	5.078576E+01	5.083213E+01	7.562982E+01	1.983412E+02	8.696132E+01	7.223114E+02	9.155977E+01	1.969723E+02
	5.079034E+01(1) ↓		1.314014E+02(2) ↕		1.904119E+02(4) ↑		1.208358E+02(2)	
f_{18}	5.692331E+02	6.814325E+01	1.407101E+02	3.691386E+01	3.177029E+02	1.152286E+02	2.285822E+02	2.759408E+01
	4.412395E+02	7.401760E+02	2.426750E+00	2.607850E+01	1.511305E+02	6.771145E+02	1.745817E+02	2.979753E+02
	5.612302E+02(4) ↑		1.352700E+02(1) ↓		3.213153E+02(3) ↑		2.313524E+02(2)	
f_{19}	6.263268E−01	1.462612E−01	8.722114E+00	2.547101E+00	1.610283E+03	5.041151E+03	6.353680E+00	2.823314E+00
	3.637270E−01	9.906629E−01	3.255279E+00	1.487337E+01	1.009866E+01	2.789496E+04	3.283236E+00	1.857356E+01
	6.131729E−01(1) ↓		8.442377E+00(3) ↑		7.340482E+01(4) ↑		5.493902E+00(2)	
f_{20}	2.490925E+01	1.485366E−01	2.306127E+01	1.373825E+00	2.162989E+01	1.429030E+00	2.134617E+01	6.348001E−01
	2.451003E+01	2.500000E+01	2.052771E+01	2.450972E+01	1.809722E+01	2.464834E+01	1.975829E+01	2.242734E+01
	2.499822E+01(4) ↑		2.346959E+01(3) ↑		2.139992E+01(1) ↕		2.143715E+01(1)	
f_{21}	2.616982E+02	1.710546E+02	3.334510E+02	3.163672E+02	1.355784E+03	7.305147E+02	8.199147E+02	3.840724E+02
	2.000000E+02	8.364610E+02	2.000000E+02	1.122186E+03	2.000000E+02	3.637026E+03	2.000000E+02	1.122354E+03
	2.003409E+02(2) ↓		2.000000E+02(1) ↓		1.122354E+03(4) ↑		1.122186E+03(3)	
f_{22}	1.958456E+01	5.250550E+00	8.268873E+03	1.303060E+03	6.843870E+03	3.014408E+03	1.595638E+03	5.388442E+02
	1.041265E+01	3.203723E+01	6.015689E+03	1.116989E+04	2.127197E+03	1.190334E+04	6.395211E+02	3.085785E+03
	1.875343E+01(1) ↓		8.234675E+03(4) ↑		6.860680E+03(3) ↑		1.517371E+03(2)	

(*continued on next page*)

Table 3 (*continued*).

f	ABCBB		GFWA		DNLPSO		FSO	
f_{23}	1.052602E+04	9.457690E+02	8.691956E+03	1.260352E+03	1.194002E+04	2.602080E+03	9.340752E+03	1.317750E+03
	8.032461E+03	1.187777E+04	6.327129E+03	1.261568E+04	6.632618E+03	1.580521E+04	6.225354E+03	1.255373E+04
	1.050870E+04(3) ↑		8.704952E+03(1) ↓		1.294030E+04(4) ↑		9.154374E+03(2)	
f_{24}	3.749278E+02	9.432216E+00	3.052683E+02	3.208367E+01	3.402890E+02	5.975840E+01	2.801362E+02	1.258812E+01
	3.537733E+02	3.953902E+02	2.177825E+02	3.520831E+02	2.554571E+02	5.212700E+02	2.464398E+02	3.179157E+02
	3.755608E+02(4) ↑		3.078204E+02(2) ↑		3.308274E+02(3) ↑		2.781137E+02(1)	
f_{25}	4.193522E+02	1.149591E+01	3.846574E+02	2.375633E+01	4.611187E+02	4.445527E+01	3.341556E+02	8.730330E+00
	3.940911E+02	4.403543E+02	3.304837E+02	4.717096E+02	3.976899E+02	5.673368E+02	3.136905E+02	3.583883E+02
	4.190195E+02(3) ↑		3.803023E+02(2) ↑		4.567835E+02(4) ↑		3.342221E+02(1)	
f_{26}	2.020808E+02	4.829488E−01	3.204895E+02	9.857138E+01	2.748602E+02	1.126149E+02	3.722643E+02	3.698227E+01
	2.012590E+02	2.035493E+02	2.000864E+02	4.330861E+02	2.001511E+02	4.885762E+02	2.001510E+02	4.000393E+02
	2.020361E+02(1) ↓		3.815151E+02(3) ↕		2.029052E+02(1) ↓		3.790210E+02(3)	
f_{27}	1.879104E+03	3.823675E+02	1.368899E+03	1.742121E+02	1.533671E+03	3.384247E+02	1.197222E+03	1.042129E+02
	4.000655E+02	2.161746E+03	9.343827E+02	1.856814E+03	9.577042E+02	2.198221E+03	8.304901E+02	1.388737E+03
	1.975134E+03(4) ↑		1.328317E+03(2) ↑		1.552595E+03(3) ↑		1.199887E+03(1)	
f_{28}	4.776233E+02	5.543248E+02	7.551512E+02	9.823446E+02	1.591741E+03	1.315416E+03	1.145162E+03	1.436423E+03
	4.000000E+02	4.358673E+03	4.000000E+02	3.454140E+03	4.000000E+02	4.749183E+03	4.000000E+02	4.058335E+03
	4.000000E+02(3) ↑		4.000000E+02(1) ↓		9.215136E+02(4) ↑		4.000000E+02(2)	
A.R.	2.5000		1.8929		3.0714		1.7143	

Table 4
Performance of FSO on 100 dimensional CEC 2013 problems.

f	ABCBB		GFWA		DNLPSO		FSO	
f_1	$\mu = 0$	$\sigma = 0$	0	0	1.884926E+04	2.612295E+04	0	0
	$b = 0$	$w = 0$	0	0	0	1.024379E+05	0	0
	m_d =0(1) ↕		0(1) ↕		8.817678E+03(4) ↑		0(1)	
f_2	2.696289E+07	4.484896E+06	2.645465E+06	6.405896E+05	7.946299E+08	8.513642E+08	2.146684E+06	4.073680E+05
	1.855633E+07	3.832372E+07	2.579221E+06	3.179528E+09	5.782645E+07	3.179528E+09	1.343423E+06	2.823117E+06
	2.686206E+07(3) ↑		5.782645E+07(2) ↑		4.659965E+08(4) ↑		2.169410E+06(1)	
f_3	1.205160E+10	5.135746E+09	4.236573E+08	2.925250E+08	5.136820E+16	3.515227E+17	3.186424E+09	2.977051E+09
	2.503006E+09	2.257441E+10	1.046528E+08	1.640046E+09	1.412800E+10	2.511870E+18	3.480590E+08	1.358168E+10
	1.267628E+10(3) ↑		3.636040E+08(1) ↓		3.350783E+12(4) ↑		2.267397E+09(2)	
f_4	1.689818E+05	2.206696E+04	0	0	5.855015E+04	4.677049E+04	7.698378E+04	1.484667E+04
	1.117254E+05	2.190934E+05	0	0	9.295743E+03	2.209533E+05	4.446474E+04	1.034743E+05
	1.687472E+05(4) ↑		0(1) ↓		4.062521E+04(2) ↓		7.740362E+04(3)	
f_5	0	0	1.655889E−03	1.118280E−04	2.149674E+03	3.218425E+03	0	0
	0	0	1.333702E−03	1.881002E−03	0	1.369924E+04	0	0
	0(1) ↕		1.660455E−03(3) ↑		6.265890E+02(4) ↑		0(1)	
f_6	1.584547E+02	2.913753E+01	2.358327E+02	5.704088E+01	4.240621E+03	4.202712E+03	1.562458E+02	6.488020E+01
	9.521203E+01	2.035012E+02	1.018891E+02	3.512735E+02	6.238394E+02	2.034866E+04	1.525018E+01	2.896437E+02
	1.661378E+02(1) ↕		2.401757E+02(3) ↑		2.439116E+03(4) ↑		1.546465E+02(1)	
f_7	1.062546E+03	8.183360E+02	1.397136E+02	3.194829E+01	6.117598E+04	2.453912E+05	9.104411E+01	1.740206E+01
	2.364798E+02	3.987374E+03	9.074641E+01	2.342160E+02	9.157872E+01	1.324173E+06	5.599305E+01	1.287569E+02
	8.158775E+02(4) ↑		1.321193E+02(2) ↑		3.027559E+02(3) ↑		8.810690E+01(1)	
f_8	2.130067E+01	2.381753E−02	2.131145E+01	4.917987E−02	2.131179E+01	2.728793E−02	2.128299E+01	3.491161E−02
	2.125026E+01	2.134346E+01	2.120360E+01	2.139502E+01	2.122036E+01	2.135366E+01	2.118673E+01	2.134315E+01
	2.130244E+01(2) ↑		2.131920E+01(3) ↑		2.131798E+01(3) ↑		2.128503E+01(1)	
f_9	1.387034E+02	3.549023E+00	9.694587E+01	1.873298E+01	1.193146E+02	2.220803E+01	9.142441E+01	5.327308E+00
	1.319234E+02	1.450086E+02	4.855223E+01	1.495314E+02	8.065653E+01	1.595961E+02	7.608983E+01	1.076424E+02
	1.385810E+02(4) ↑		9.596003E+01(2) ↑		1.192976E+02(3) ↑		9.169914E+01(1)	
f_{10}	1.007674E+00	3.175973E−01	4.279886E−02	2.736799E−02	3.293424E+03	4.015444E+03	9.585188E−02	6.790269E−02
	1.984148E−01	1.385810E+00	7.396040E−03	1.207488E−01	9.094511E−02	1.867904E+04	9.857285E−03	3.175776E−01
	1.147815E+00(3) ↑		3.447713E−02(1) ↓		1.653593E+03(4) ↑		8.124342E−02(2)	
f_{11}	0	0	3.712735E+02	1.132284E+02	7.120635E+02	4.274376E+02	3.735299E+02	8.072794E+01
	0	0	1.552133E+02	6.875074E+02	1.791993E+02	2.000055E+03	2.169006E+02	6.357746E+02
	0(1) ↓		3.631583E+02(2) ↕		5.524056E+02(4) ↑		3.601733E+02(2)	
f_{12}	1.819905E+03	2.335952E+02	3.708750E+02	1.160516E+02	6.851580E+02	4.886436E+02	3.704596E+02	5.849581E+01
	1.281708E+03	2.488944E+03	1.144202E+02	7.243225E+02	1.433170E+02	1.967279E+03	2.539567E+02	5.012953E+02
	1.840180E+03(4) ↑		3.591790E+02(1) ↕		5.185673E+02(3) ↑		3.702815E+02(1)	
f_{13}	2.024673E+03	1.954380E+02	7.325591E+02	1.697462E+02	1.044028E+03	4.414304E+02	5.435308E+02	7.886530E+01
	1.502904E+03	2.471181E+03	4.517451E+02	1.284213E+03	4.568537E+02	2.408744E+03	3.587811E+02	7.440087E+02
	2.032999E+03(4) ↑		7.460565E+02(2) ↑		8.902323E+02(3) ↑		5.420385E+02(1)	
f_{14}	5.477252E+00	1.479705E+00	1.043851E+04	1.531874E+03	1.282070E+04	7.458332E+03	5.543642E+03	1.048739E+03
	2.825886E+00	9.030542E+00	7.053042E+03	1.587193E+04	4.924814E+03	2.749976E+04	3.881237E+03	8.814382E+03
	5.419896E+00(1) ↓		1.034968E+04(3) ↑		1.043890E+04(3) ↑		5.319933E+03(2)	

(*continued on next page*)

Table 4 (*continued*).

f	ABCBB		GFWA		DNLPSO		FSO	
f_{15}	1.613962E+04	7.909618E+02	1.366685E+04	1.644892E+03	2.385344E+04	6.523655E+03	2.065651E+04	1.386008E+03
	1.459011E+04	1.779132E+04	1.016145E+04	1.672143E+04	1.134852E+04	3.123725E+04	1.656657E+04	2.326446E+04
	1.619100E+04(2) ↓		1.381308E+04(1) ↓		2.694832E+04(4) ↑		2.082628E+04(3)	
f_{16}	2.521188E+00	2.722387E−01	3.840948E−02	3.161007E−02	3.647256E+00	8.197243E−01	3.942619E+00	2.525466E−01
	2.044540E+00	3.105710E+00	9.017893E−03	1.511944E−01	1.548340E+00	4.460020E+00	3.240814E+00	4.345709E+00
	2.559024E+00(2) ↓		2.682927E−02(1) ↓		3.965931E+00(3) ↕		3.947625E+00(3)	
f_{17}	1.015783E+02	1.366878E−02	2.781178E+02	8.314719E+01	6.163356E+02	5.477824E+02	2.384915E+02	3.434119E+01
	1.015625E+02	1.016300E+02	1.674701E+02	5.759491E+02	2.340417E+02	3.311361E+03	1.742766E+02	3.227595E+02
	1.015748E+02(1) ↓		2.623005E+02(3) ↑		3.939285E+02(4) ↑		2.358145E+02(2)	
f_{18}	1.779936E+03	1.948583E+02	2.706375E+02	6.393614E+01	1.040562E+03	5.848978E+02	6.636707E+02	5.209247E+01
	1.452996E+03	2.323657E+03	1.653441E+02	4.650663E+02	3.642386E+02	3.032475E+03	5.608998E+02	7.831968E+02
	1.769390E+03(4) ↑		2.788345E+02(1) ↓		9.470821E+02(3) ↑		6.672201E+02(2)	
f_{19}	1.507779E+00	2.866941E−01	2.020472E+01	7.020233E+00	3.746009E+04	1.195081E+05	5.279156E+01	3.050148E+01
	8.686085E−01	2.289392E+00	9.224990E+00	4.347949E+01	7.971954E+01	7.056324E+05	1.925415E+01	1.705975E+02
	1.461679E+00(1) ↓		1.863904E+01(2) ↓		1.966623E+03(4) ↑		4.293583E+01(3)	
f_{20}	5.000000E+01	0	5.000000E+01	0	4.993830E+01	1.576547E−01	4.999475E+01	3.749851E−02
	5.000000E+01	5.000000E+01	5.000000E+01	5.000000E+01	4.950000E+01	5.000000E+01	4.973221E+01	5.000000E+01
	5.000000E+01(1) ↕		5.000000E+01(1) ↕		5.000000E+01(1) ↓		5.000000E+01(1)	
f_{21}	3.207083E+02	4.492983E+01	3.294118E+02	4.601790E+01	2.299040E+03	1.894954E+03	3.882353E+02	3.253957E+01
	2.076783E+02	4.000000E+02	3.000000E+02	4.000000E+02	4.348619E+02	7.275756E+03	3.000000E+02	4.000000E+02
	3.000000E+02(1) ↓		3.000000E+02(1) ↓		1.607397E+03(4) ↑		4.000000E+02(3)	
f_{22}	3.426373E+01	1.583822E+01	1.706148E+04	2.837589E+03	1.761936E+04	6.698643E+03	5.377192E+03	9.988508E+02
	2.174989E+01	1.249694E+02	1.117650E+04	2.410636E+04	6.646667E+03	3.035387E+04	2.946532E+03	7.762149E+03
	2.856846E+01(1) ↓		1.689743E+04(3) ↑		1.635789E+04(3) ↑		5.493361E+03(2)	
f_{23}	2.262400E+04	1.280026E+03	1.942530E+04	2.235490E+03	2.579050E+04	5.181124E+03	2.264511E+04	2.111978E+03
	1.933255E+04	2.499096E+04	1.313139E+04	2.448482E+04	1.598529E+04	3.244360E+04	1.898425E+04	2.840562E+04
	2.267416E+04(2) ↕		1.960696E+04(1) ↓		2.584436E+04(4) ↑		2.232669E+04(2)	
f_{24}	6.161999E+02	1.475029E+01	5.032984E+02	6.020604E+01	6.271520E+02	2.062676E+02	4.082594E+02	1.697913E+01
	5.863276E+02	6.471783E+02	3.975561E+02	7.669412E+02	3.633188E+02	1.352143E+03	3.771507E+02	4.436581E+02
	6.151907E+02(3) ↑		4.958378E+02(2) ↑		5.542458E+02(3) ↑		4.116526E+02(1)	
f_{25}	7.213921E+02	1.992680E+01	6.645119E+02	6.061913E+01	8.513304E+02	1.220622E+02	5.078497E+02	1.430700E+01
	6.833715E+02	7.631227E+02	5.629042E+02	7.920842E+02	6.354759E+02	1.087494E+03	4.756492E+02	5.417507E+02
	7.252695E+02(3) ↑		6.488780E+02(2) ↑		8.588907E+02(4) ↑		5.076834E+02(1)	
f_{26}	2.964896E+02	1.874689E+02	5.414980E+02	4.023402E+01	5.765066E+02	8.680846E+01	5.100151E+02	1.351902E+01
	2.028260E+02	6.921487E+02	4.543729E+02	6.163042E+02	4.051574E+02	7.272618E+02	4.813612E+02	5.325291E+02
	2.055263E+02(1) ↓		5.468247E+02(3) ↑		5.822544E+02(4) ↑		5.121062E+02(2)	
f_{27}	4.147964E+03	7.780751E+02	3.170055E+03	3.245474E+02	3.724674E+03	8.608956E+02	2.579881E+03	1.431003E+02
	4.001825E+02	4.589274E+03	1.887273E+03	3.772787E+03	1.828917E+03	5.807058E+03	2.198366E+03	2.852101E+03
	4.294089E+03(4) ↑		3.233484E+03(2) ↑		3.667432E+03(3) ↑		2.578379E+03(1)	
f_{28}	7.162620E+03	2.332630E+03	3.524154E+03	9.919295E+02	8.251396E+03	3.688100E+03	7.336450E+03	1.964827E+03
	3.810742E+03	1.236367E+04	2.702313E+03	4.922516E+03	2.981866E+03	1.525838E+04	3.580214E+03	1.011746E+04
	6.866818E+03(2) ↕		2.851488E+03(1) ↓		7.898642E+03(2) ↕		7.546185E+03(2)	
A.R.	2.2857		1.8214		3.3571		1.7143	

Figs. 4–7 show the convergence characteristics of different algorithms on all 28 problems in the CEC 2013 test suite. It is observed that the performance of the FSO is highly competitive on all 28 problems.

Results presented in Tables 2–4 clearly indicate that the FSO algorithm performs significantly better on multimodal, composite and high dimensional problems. FSO also demonstrates graceful degradation of performance with increasing dimensions and is the overall best performing algorithm in terms of the average rank sum metric (Tables 2–4). Table 5 below summarizes the results of performance comparison on CEC 2013. An algorithm is listed above another algorithm in Table 5 if it is known to outperform the algorithm listed below it.

Table 5 summarizes the results of performance comparison on CEC 2013. An algorithm is listed above another algorithm in Table 5 if it is known to outperform the algorithm listed below it. No hierarchy is implied if two algorithms are listed in the same cell of Table 5. For example, Table 5 does not imply that PSO outperforms CMA-ES since both are listed in the same cell. However, Table 5 implies that GFWA outperforms both PSO and CMA-ES since these are listed in the row below it.

Table 5
Hierarchy based on Average Rank-Sum metric on CEC 2013.

FSO		
ABCBB	GFWA	DNLPSO
AAA	ABC	PSOLocal
GSO	DE	UPSO
	PSO	FIPS
	CMA-ES	DMSPSO
	EFWA	CLPSO
	AFWA	
	dynFWA	

4. Conclusion

A new biologically inspired global optimization algorithm inspired by the reproductive swarming of firebugs searching for fit mates was proposed. The proposed FSO algorithm outperforms on the average 17 popular global optimization algorithms (PSOLocal, UPSO, FIPS, DMSPSO, CLPSO, DNLPSO, AAA, GSO, ABC, DE, PSO, CMA-ES, EFWA, AFWA, dynFWA, GFWA, ABCBB) on 28 benchmark problems in the CEC 2013 test suite. FSO avoids premature convergence to a local minimum because female bugs are not attracted to a single best male

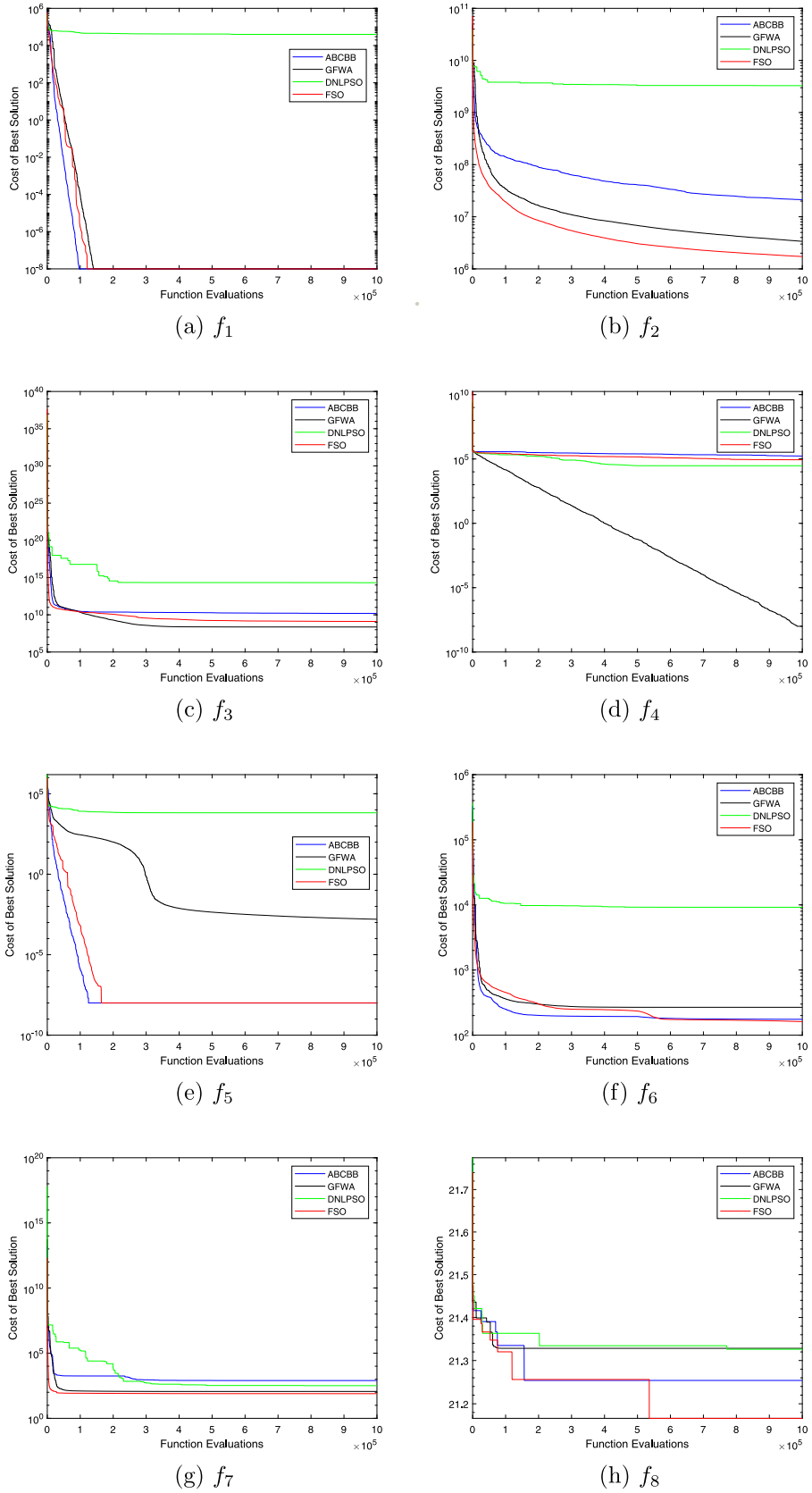


Fig. 4. Convergence characteristics for 100 dimensions.

bug. Also, male bugs are not attracted to other male bugs but only copy the direction of movement of random male bugs. The above features make the FSO more exploratory and suitable for highly multimodal and high-dimensional problems. The computationally costly part of the FSO algorithm uses element-wise (Hadamard) matrix operations to facilitate efficient vectorized implementations. Unlike PSO and similar

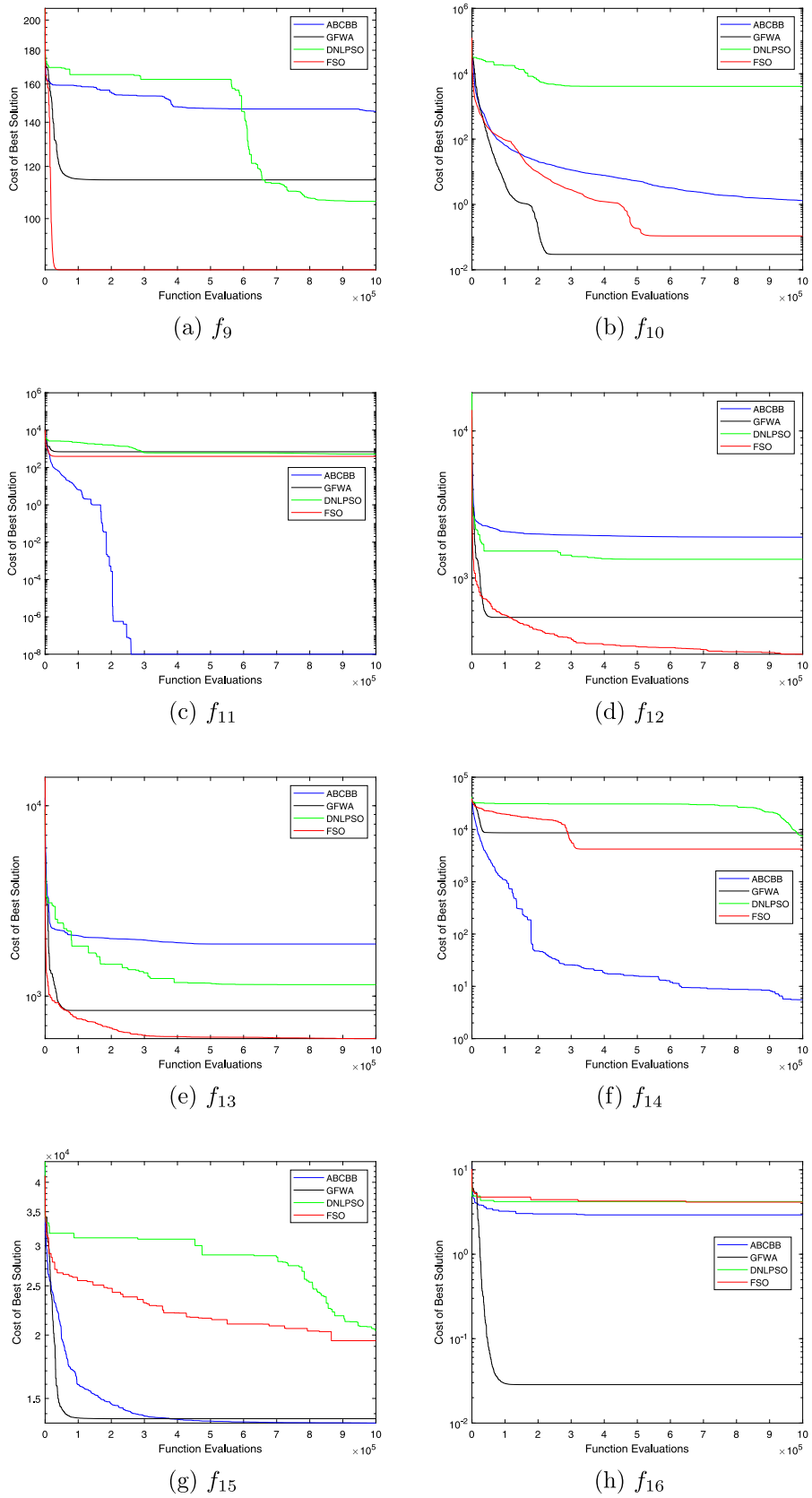


Fig. 5. Convergence characteristics for 100 dimensions (contd.).

algorithms that use the same random number to update the particle's location along different independent dimensions, the FSO uses a random matrix to significantly improve exploration and allow independent exploration in different dimensions. Future work can investigate whether more realistic Firebug behaviour models improve performance: for example, in the FSO all firebugs are assumed to be identical.

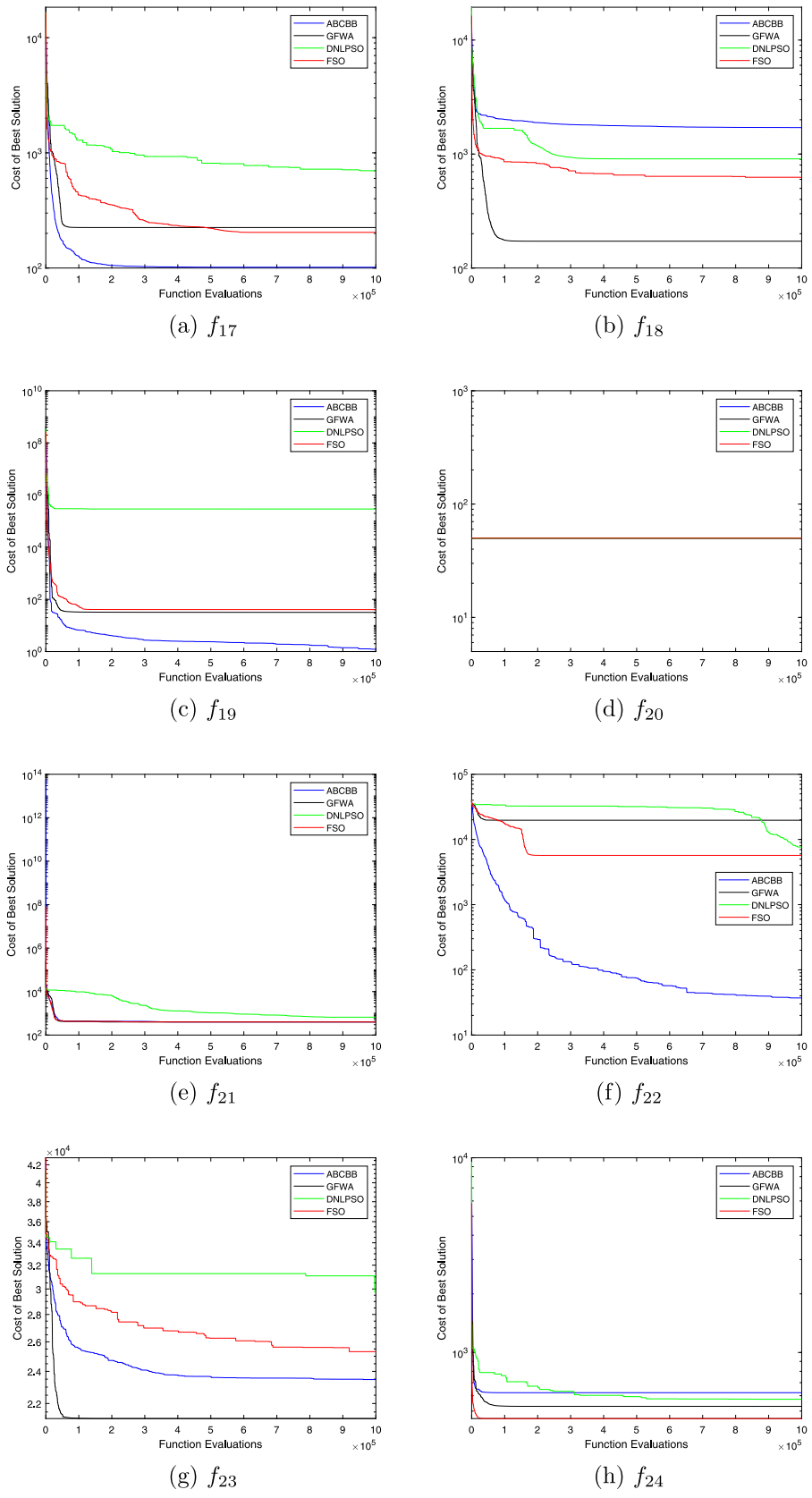
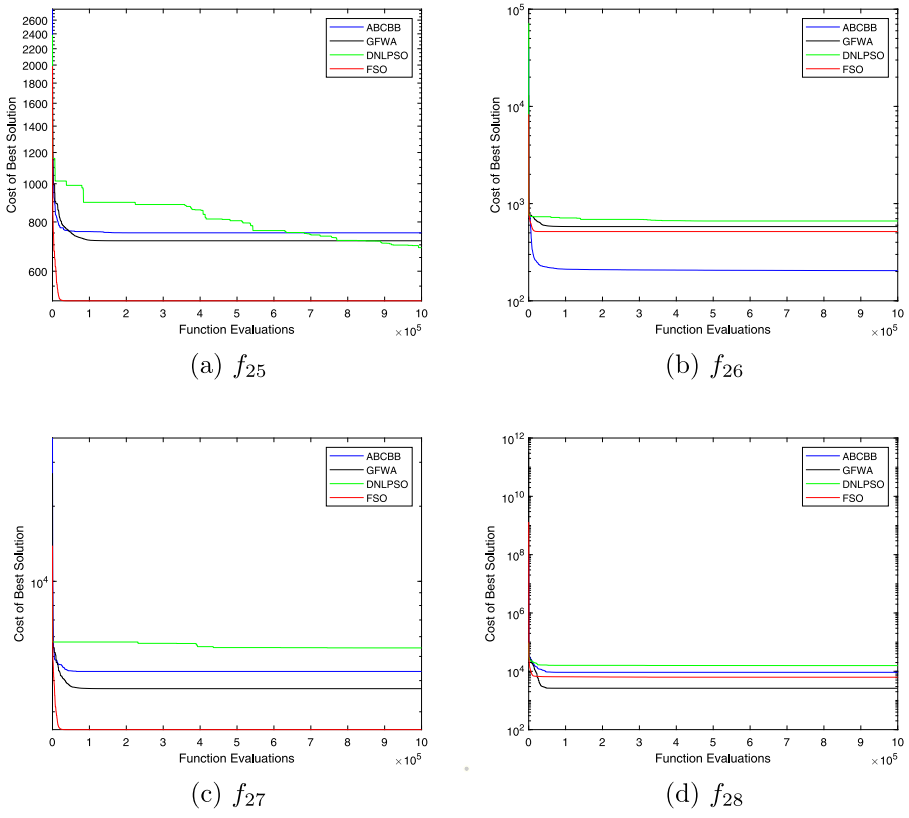


Fig. 6. Convergence characteristics for 100 dimensions (contd.).

However, in nature, certain bugs are observed to be braver and more exploratory. Thus the effect of using a more accurate model that allows different bugs to have different exploratory trait strengths on FSO performance can be studied.



(a) f_{25} (b) f_{26}

(c) f_{27} (d) f_{28}

Fig. 7. Convergence characteristics for 100 dimensions (contd.).

The following can be considered for future work:

1. Parallel implementation of the FSO using multi-core architectures to solve large scale optimization problems with dimensions exceeding 100 in a reasonable time.
2. Tuning the parameters of the FSO using Fuzzy logic to improve performance and comparison to other Fuzzy logic tuned optimization algorithms (Bernal et al., 2020; Olivas et al., 2017, 2016, 2019).
3. Application to important real world problems like optimal filter design, optimal design of mechanical structures, electromagnetic optimization, optimal controller design, and power system optimization can be explored.
4. Since FSO uses Hadamard matrix operations to synchronously update all females in a colony, the use of GPUs to reduce computational time can also be explored.

CRediT authorship contribution statement

Mathew Mithra Noel: Conceptualization, Methodology, Writing - reviewing and editing. **Venkataraman Muthiah-Nakarajan:** Formal analysis, Software, Investigation, Writing - original draft. **Geraldine Bessie Amali:** Validation, Resources. **Advait Sanjay Trivedi:** Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to thank Dr. E. P. N. Suganthan, Dr. B. Kocer, and Dr. J. Li for generously providing MATLAB codes for their algorithms which were used for comparison in this paper.

Appendix A. Nomenclature

$\odot$	Hadamard multiplication $C = A \odot B \implies C_{ij} = A_{ij} B_{ij}$
D	Dimension of the search space
N_M	Number of male Firebugs
N_F	Number of female Firebugs
$\mathbf{m}.\mathbf{F}$	Matrix whose columns contain female bug positions associated with male bug $\mathbf{m}$
f	Vectorized cost function to be minimized
$\mathbf{m}.\mathbf{F}_f$	Row vector of function values corresponding to the female bug positions in the $\mathbf{m}$th male's colony
$\mathbf{m}.\mathbf{x}$	Position of the $\mathbf{m}$th male bug in the search space, $\mathbf{x} \in \Re^{n\times 1}$
$\mathbf{m}.\mathbf{x}_f$	Cost function value at the location of the $\mathbf{m}$th male bug
$\mathbf{g}$	Location of the best solution computed so far $\mathbf{g} \in \Re^{n\times 1}$
$\mathbf{g}_f$	$f(\mathbf{g})$
$\mathbf{C_i}$	Acceleration coefficients
$rand(M, N)$	Matrix of shape $M \times N$ containing uniformly distributed random numbers in the interval [0, 1]
$repmat(\mathbf{A}, M, N)$	Array copy operation that returns a matrix containing copies of the matrix $\mathbf{A}$. M copies of $\mathbf{A}$ are made along the row dimension and N copies of $\mathbf{A}$ are made along the column dimension.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.eswa.2021.115408.

References

Bernal, E., Castillo, O., Soria, J., & Valdez, F. (2020). Fuzzy galactic swarm optimization with dynamic adjustment of parameters based on fuzzy logic. *SN Computer Science*, *59*, https://doi.org/10.1007/s42979-020-0062-4.

Cang, Y. T., Lin, J., & Shieh, J. S. (2012). Optimization the initial weights of artificial neural networks via genetic algorithm applied to hip bone fracture prediction. *Hybrid Biomedical Intelligent Systems, Open Access*, *2012*, Article 951247, 9 pages. https://doi.org/10.1155/2012/951247.

Chen, X., Liu, N., Su, Y., & Chen, G. (2020). A survey of swarm intelligence techniques in VLSI routing problems. *IEEE Access*, *8*, 26266–26292. https://doi.org/10.1109/ACCESS.2020.2971574.

Chong, E. K. P., & Zak, S. H. (2005). *Penalty methods. An introduction to optimization* (pp. 445–451). John Wiley & Sons, Inc.

Conover, W. J. (1980). Practical nonparametric statistics. *John Wiley and Sons*, 225–226.

Dikin, I. I. (1967). Iterative solution of problems of linear and quadratic programming. *Doklady Akademii Nauk SSSR*, *174*(1), 747–748.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system : a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, *1*, 53–66. https://doi.org/10.1109/4235.585892.

Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machines and human science* (pp. 39–43).

Ghamisi, P., Couceiro, M. S., Martins, F. M. L., & Benediktsson, J. A. (2014). Multilevel image segmentation based on fractional-order darwinian particle swarm optimization. *IEEE Transactions on Geoscience and Remote Sensing*, *52*, 2382–2394. https://doi.org/10.1109/TGRS.2013.2260552.

Goldberg, D. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley Professional.

Gyuris, E., Fero, O., Tartally, A., & Barta, Z. (2011). Individual behaviour in firebugs (Pyrrhocoris apterus). *Proceedings of Royal Society B*, *278*, 628–633. https://doi.org/10.1098/rspb.2010.1326.

Hansen, N. (2006). The CMA evolution strategy: a comparing review. In J. A. Lozano, P. Larrañaga, I. Inza, & E. Bengoetxea (Eds.), *Studies in fuzziness and soft computing*: *Vol. 192*, *Towards a new evolutionary computation* (pp. 1769–1776). https://doi.org/10.1007/3-540-32494-1_4.

Karaboga, D. (2005). *An Idea Based on Honey Bee Swarm for Numerical Optimization: Technical report-TR06*, Erciyes University, Engineering Faculty, Computer Engineering Department.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671.

Kocer, B. (2016). Bollinger bands approach on boosting ABC algorithm and its variants. *Applied Soft Computing*, *49*, 292–312. https://doi.org/10.1016/j.asoc.2016.08.023.

Li, X. D., & Engelbrecht, A. P. (2007). Particle swarm optimization: an introduction and its recent developments. In *Proceedings of genetic evolutionary computation conference* (pp. 3391–3414). https://doi.org/10.1145/1274000.1274118.

Li, J., Zheng, S., & Tan, Y. (2014). Adaptive fireworks algorithm. In *IEEE congress on evolutionary computation 2014* (pp. 3214–3221). https://doi.org/10.1109/CEC.2014.6900418.

Li, J., Zheng, S., & Tan, Y. (2016). The effect of information utilization: Introducing a novel guiding spark in the fireworks algorithm. *IEEE Transactions on Evolutionary Computation*, *21*, 153–166. https://doi.org/10.1109/TEVC.2016.2589821.

Liang, J. J., Qin, A. K., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, *10*, 281–295. https://doi.org/10.1109/TEVC.2005.857610.

Liang, J. J., Qu, B., Suganthan, P., & Hernandez, A. G. (2013). *Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization: Technical report 2012-13*, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China And Technical Report, Nanyang Technological University, Singapore.

Liang, J. J., & Suganthan, P. N. (2005). Dynamic multi-swarm particle swarm optimizer. In *Proceedings of swarm intelligence symposium* (pp. 124–129).

Mandal, P., Sarkar, B. K., Saha, R., Mookherjee, S., Acharyya, S. K., & Sanyal, D. (2015). GA-optimized fuzzy-feedforward-bias control of motion by a rugged electro-hydraulic system. *IEEE/ASME Transactions on Mechatronics*, *20*, 1734–1742. https://doi.org/10.1109/TMECH.2014.2352156.

Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, *8*, 204–210. https://doi.org/10.1109/TEVC.2004.826074.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, *69*, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007.

Mohamed, A. A. S., Berzoy, A., & Mohammed, O. A. (2017). Design and hardware implementation of FL-MPPT control of PV systems based on GA and small-signal analysis. *IEEE Transactions on Sustainable Energy*, *8*, 279–290. https://doi.org/10.1109/TSTE.2016.2598240.

Muthiah-Nakarajan, V., & Noel, M. M. (2016). Galactic swarm optimization: A new global optimization metaheuristic inspired by galactic motion. *Applied Soft Computing*, *38*, 771–787. https://doi.org/10.1016/j.asoc.2015.10.034.

Nasir, M., Das, S., Maity, S., Halder, U., & Suganthan, P. N. (2012). A dynamic neighbourhood learning based optimizer for global numerical optimization. *Information Sciences*, *209*, 16–36. https://doi.org/10.1016/j.ins.2012.04.028.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, *7*(4), 308–313. https://doi.org/10.1093/comjnl/7.4.308.

Olivas, F., Valdez, F., Castillo, O., Gonzalez, C. I., & Melin, P. (2017). Ant colony optimization with dynamic parameter adaptation based on interval type-2 fuzzy logic systems. *Applied Soft Computing*, *53*, 74–87. https://doi.org/10.1016/j.asoc.2016.12.015.

Olivas, F., Valdez, F., Castillo, O., & Melin, P. (2016). Dynamic parameter adaptation in particle swarm optimization using interval type-2 fuzzy logic. *Soft Computing*, *20*, 1057–1070. https://doi.org/10.1007/s00500-014-1567-3.

Olivas, F., Valdez, F., Melin, P., Sombra, A., & Castillo, O. (2019). Interval type-2 fuzzy logic for dynamic parameter adaptation in a modified gravitational search algorithm. *Information Sciences*, *476*, 159–175. https://doi.org/10.1016/j.ins.2018.10.025.

Parsopoulos, K. E., & Vrahatis, M. N. (2004). UPSO: a unified particle swarm optimization scheme. In *Lecture Series on Computer and Computational Sciences*: *Vol. 1*, *Proceedings of the international conference of computational methods in sciences and engineering* (pp. 868–873). Zeist, The Netherlands: VSP International Science Publishers, https://doi.org/10.1201/9780429081385-222.

Reams, R. (1999). Hadamard inverses, square roots and products of almost semidefinite matrices. *Linear Algebra and Its Applications*, *288*, 35–43. https://doi.org/10.1016/S0024-3795(98)10162-3.

Ryser, H. J. (1963). *Combinatorial mathematics*. Mathematical Association of America.

Schmuck, R. (1995). Adaptive value of aggregation behavior in the fire bug pyrrhocoris apterus (Heteroptera: Pyrrhocoridae). *Entomologia Generalis*, *19*, 143–156. https://doi.org/10.1127/entom.gen/19/1995/143.

Schofl, G., & Taborsky, M. (2002). Prolonged tandem formation in firebugs (Pyrrhocoris apterus) serves mate-guarding. *Behav Ecol Sociobiol*, *52*, 426–433. https://doi.org/10.1007/s00265-002-0524-9.

Shen, H., Liu, G., & Chandler, H. (2015). Swarm intelligence based file replication and consistency maintenance in structured P2P file sharing systems. *IEEE Transactions on Computers*, *64*, 2953–2967. https://doi.org/10.1109/TC.2015.2389845.

Storn, R., & Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*, 341–359. https://doi.org/10.1023/A:1008202821328.

Styan, G. P. H. (1973). Hadamard products and multivariate statistical analysis. *Linear Algebra and its Applications*, *6*, 217–240. https://doi.org/10.1016/0024-3795(73)90023-2.

Svádová, H., Exnerova, K., & Pavel, A. S. (2014). Gregariousness as a defence strategy of moderately defended prey: Experiments with Pyrrhocoris apterus and avian predators. *Behaviour*, *151*, 1617–1640. https://doi.org/10.1163/1568539X-00003208.

Uymaza, S. A., Tezel, G., & Yel, E. (2015). Artificial algae algorithm (AAA) for nonlinear global optimization. *Applied Soft Computing*, *31*, 153–171. https://doi.org/10.1016/j.asoc.2015.03.003.

Vanderbei, R. J. (2001). A path following method. In *Linear programming: Foundations and extensions* (pp. 269–283). Kluwer.

Wang, W., Lu, Y., Fu, J. S., & Xiong, Y. Z. (2005). Particle swarm optimization and finite-element based approach for microwave filter design. *IEEE Transactions on Magnetics*, *41*, 1800–1803. https://doi.org/10.1109/TMAG.2005.846467.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*, 67–82. https://doi.org/10.1109/4235.585893.

Yun, Y. C., Oh, S. H., Lee, J. H., Choi, K., Chung, T. K., & Kim, H. S. (2016). Optimal design of a compact filter for UWB applications using an improved particle swarm optimization. *IEEE Transactions on Magnetics*, *52*, 1–4. https://doi.org/10.1109/TMAG.2015.2486141.

Zdarek, J. (1970). Mating behaviour in the bug, pyrrhocoris apterus. (Heteroptera): Ontogeny and its environmental control. *Behaviour*, *37*, 253–268.

Zemek, R., & Socha, R. (2010). Habitat selection in the bug pyrrhocoris apterus: Does it minimize the risk of being parasitized by the ectoparasitic mite hemipteroseius adleri. In M. Sabelis, & J. Bruin (Eds.), *Trends in acarology*. Dordrecht: Springer, https://doi.org/10.1007/978-90-481-9837-5_59.

Zhang, B., Zheng, Y., Zhang, M., & Chen, S. (2015). Fireworks algorithm with enhanced fireworks interaction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *14*, 42–55. https://doi.org/10.1109/TCBB.2015.2446487.

Zheng, S., Janecek, A., Li, J., & Tan, Y. (2014). Dynamic search in fireworks algorithm. In *IEEE congress on evolutionary computation 2014* (pp. 3222–3229). https://doi.org/10.1109/CEC.2014.6900485.