Expert Systems With Applications 83 (2017) 63–78

Contents lists available at ScienceDirect

Expert Systems With Applications

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

# Memory based Hybrid Dragonfly Algorithm for numerical optimization problems

Sree Ranjini K.S.[a,*], S. Murugan[b]

[a] Homi Bhabha National Institute, Indira Gandhi Centre for Atomic Research, Kalpakkam, India
[b] Remote handling, Irradiation & Robotics Division, Indira Gandhi Centre for Atomic Research, Kalpakkam, India

A R T I C L E I N F O

*Article history:*
Received 5 January 2017
Revised 27 February 2017
Accepted 17 April 2017
Available online 18 April 2017

*Keywords:*
Dragonfly algorithm
Particle Swarm Optimization
Hybridization
Benchmark functions
Engineering problems
Friedman's test

A B S T R A C T

Dragonfly algorithm (DA) is a recently proposed optimization algorithm based on the static and dynamic swarming behaviour of dragonflies. Due to its simplicity and efficiency, DA has received interest of researchers from different fields. However, it lacks internal memory which may lead to its premature convergence to local optima. To overcome this drawback, we propose a novel Memory based Hybrid Dragonfly Algorithm (MHDA) for solving numerical optimization problems. The *pbest* and *gbest* concept of Particle Swarm optimization (PSO) is added to conventional DA to guide the search process for potential candidate solutions and PSO is then initialized with pbest of DA to further exploit the search space. The proposed method combines the exploration capability of DA and exploitation capability of PSO to achieve global optimal solutions. The efficiency of the MHDA is validated by testing on basic unconstrained benchmark functions and CEC 2014 test functions. A comparative performance analysis between MHDA and other powerful optimization algorithms have been carried out and significance of the results is proved by statistical methods. The results show that MHDA gives better performance than conventional DA and PSO. Moreover, it gives competitive results in terms of convergence, accuracy and search-ability when compared with the state-of-the-art algorithms. The efficacy of MHDA in solving real world problems is also explained with three engineering design problems.

## 1. Introduction

Optimization process have become an integral part of engineering and business problems. The purpose of the optimization can be for the maximization of efficiency, performance, productivity or social welfare. In real world, resources, time and money are always limited and hence there is an inevitable need for finding out solutions for optimal usage of these valuable resources under various constraints (Yang, 2014a). In recent years stochastic algorithms have been gaining significance in producing fast, low cost and robust solution to complex optimization problems (Dorigo & Thomas, 2004). Compared to conventional deterministic approach, they don't require any gradient information and are simple and easy to implement (Blum & Li, 2008). Among the stochastic optimization algorithms, swarm intelligence (SI) based optimization techniques have attracted the attention of researchers world wide. A swarm is characterized by a group of self-organized and decentralized system of non-complex individuals or agents interacting among themselves and with their environment for survival, hunting, navigation or foraging. It can be school of fish, flock of birds, colonies of ants etc. SI based algorithms models the collective behaviour of these individuals to solve complex optimization process. Even though as individuals, these agents have limited operational capability, they tend to outperform in accomplishing the desired task by interacting among themselves and with the environment using their own specific behavioural patterns. Literature review on the SI based optimization algorithms reveals their effectiveness in solving complex optimization problems in different fields of study. Ant colony optimization inspired by the foraging behaviour of the ants was found to be very effective in solving structural optimization problems (Luh & Lin, 2009), traffic area control problems (Sattari, Malakooti, Jalooli, & Noor, 2014) and also in the field of genomics (Greene, White, & Moore, 2008). Particle swarm algorithm (PSO) is well known optimization algorithm mimicking the social behaviour of bird flocking or fish schooling (Eberhart & Kennedy, 1995). The effectiveness of PSO in solving bi level programming problems (Kuo & Huang, 2009), electric power systems (AlRashidi & El-Hawary, 2009), offshore heavy oil reservoir (Wang & Qiu, 2013), and image processing (Omran, Engelbrecht, & Salman, 2006) is clearly explained in the literature. Bat

* Corresponding author.
*E-mail addresses:* srks@igcar.gov.in, sreeranjiniks53@gmail.com (S.R. K.S.), murugan@igcar.gov.in (S. Murugan).

http://dx.doi.org/10.1016/j.eswa.2017.04.033
0957-4174/© 2017 Elsevier Ltd. All rights reserved.

algorithm (Yang, 2010b), Firefly algorithm (Yang, 2009), Krill Herd (Gandomi & Alavi, 2012), Whale optimization algorithm (Mirjalili & Lewis, 2016), Grey wolf optimization (Mirjalili, Saremi, Mirjalili, & dos S. Coelho, 2016), Ageist Spider Monkey optimization (Sharma, Sharma, Panigrahi, Kiran, & Kumar, 2016), Moth search optimization (Wang, 2016), Competitive optimization algorithm (Kashani, Gandomi, & Mousavi, 2016)are some of the popular swarm based meta heuristic algorithms.

With development of numerous optimization algorithms, it is difficult to test and determine which algorithm is most suitable for solving a particular optimization problem. This is because most of the algorithms works on generalized concept and don't have domain knowledge specific to each problem. Hybridization process gains importance in this situation, as it combines the desirable properties of different approaches to mitigate their individual weaknesses (Thangaraj, Pant, Abraham, & Bouvry, 2011). Lesser computation, improvement of solution accuracy, enhancement of algorithm stability and the handling of searching convergence can be considered as targets of hybridization and improvement process. A number of hybridized versions of many conventional algorithms have evolved recently as a part of this process. They tend to show shows remarkable improved performance compared to their traditional counterparts. Nasir, Tokhi, and Ghani (2015) proposed adaptive chemotactic step size based bacterial foraging algorithm depending on individual bacterium fitness value, index of iteration and index of chemotaxis. An improved version of Differential Evolution (DE) algorithm combining different mutation operators and empowered by co-variance adaptation matrix evolution strategy algorithm as a local search is introduced by Elsayed, Sarker, and Essam (2013). Improved PSO based on adaptive inertial weight, introduced in the year 2011 (Nickabadi, Ebadzadeh, & Safabakhsh, 2011) and is found to be very effective in solving real engineering problems. Li, Wang, Yan, and Li (2015) proposed a hybrid PS-ABC combining the local search capabilities of PSO and global search capabilities of Artificial Bee Colony (ABC) algorithm and found that the hybrid algorithm is a better solution than the parent algorithms in terms of speed, convergence and robustness. Nabil (2016) investigated the performance of flower pollination algorithm incorporating colonal selection operator and validated the improved performance through standard benchmark functions. Garg (2016) proposed hybrid optimization algorithm based on Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) for solving constrained optimization problems. A hybrid ICE- SA algorithm based on Imperialist Competitive algorithm (ICE) and Simulated annealing (SA) is proposed for multi source multi product location-routing-inventory problem by Ghorbani and Jokar (2016).

The objective of the paper is to introduce a novel hybrid version of Dragonfly algorithm (DA) which is a recently evolved metaheuristic optimization algorithm proposed by Mirjalili (2016). Because of its simple and easy implementation, DA has been used to solve many real world optimization problems such as extension of RFID network lifetime (Hema, Sankar, & Sandhya, 2016), range based wireless node equalization (Daely & Shin, 2016), threshold for multilevel segmentation of digital images (Sambandam & Jayaraman, 2016), and photo-voltaic systems (Raman, Raman, Manickam, & Ganesan, 2016). However DA does not keep track of its best positions in previous generations which limits its exploitation capability and causes premature convergence to local optima. Even though global search capability of DA is good with randomization and static swarm behaviour, the local search capability is limited which causes the solutions to get trapped in local optima. In order to overcome these shortcomings, we propose a novel Memory based Hybrid Dragonfly Algorithm (MHDA). The proposed MHDA works in two stages, in the first stage, memory element is incorporated in DA algorithm so as to store the coordinates in the solution space which are associated with the best solution (fitness) that has achieved so far by the dragonfly and in the second stage PSO is initialized with this best solutions for further exploitation. Thus exploration capability at the initial stages and exploitation capability at the later stages is guaranteed by iteration level hybridization process and ensures to obtain the global optimum with increased accuracy. The performance of MHDA is compared with other state-of-the-art algorithms on two benchmark function suites. Suite-I consist of benchmark functions commonly found in literature and Suite-II consist of CEC 2014 functions. The significance of the experimental results is proved by statistical analysis.

The paper is organized as follows. Section 2 describes the conventional DA and PSO algorithms. Section 3 describes about the proposed MHDA and its functioning. Section 4 describes the performance evaluation and detailed analysis of MHDA. Section 5 discusses the application of MHDA on engineering problems, comparison of its performance with other conventional algorithms and its statistical results. Finally, conclusion and future scope are described in Section 6.

## 2. Related work

### 2.1. DA

Dragonfly algorithm is inspired by the unique and superior swarming behaviour of dragonflies. The dragonfly swarms for hunting and migration. Hunting swarm behaviour which is otherwise known as static swarm behaviour is characterized by the formation of small group of dragonflies moving locally and abruptly changing the steps. Migratory swarm behaviour which is otherwise known as dynamic swarm is characterized by a massive number of dragonflies flying in one direction over long distances. Static Swarm and dynamic swarms represents exploitation and exploration capabilities of DA. The behaviour of dragonfly follows the principles of separation, alignment, cohesion, distraction from the enemies and attraction towards the food. Each dragon fly in the swarm corresponds to the solution in the search space. Swarm movement of dragonfly is determined by five different operators such as Separation, Alignment, Cohesion, Attraction towards food sources and distraction towards enemy sources (Reynolds, 1987). Separation ($S_i$) which refers to the static collision avoidance of individuals from other individuals in the neighbourhood. Alignment ($A_i$) refers to the velocity matching of individuals to other individuals in neighbourhood. Cohesion ($C_i$) refers to the tendency of individuals towards the center of the mass of the neighbourhood. Suitable weights are assigned to each operators and they are adaptively tuned to ensure the convergence of dragonflies towards optimal solution. The neighbouring radius of the dragonflies also increases as the process of optimization progresses. The mathematical implementation of DA can be explained as follows.

Consider population of dragonflies of size N. The position of *i*th dragonfly is given by Eq. (1)

$$X_i = (x_i^1, x_i^d \ldots, x_i^N) \tag{1}$$

where i = 1,2,3... N, $x_i^d$corresponds to the position of the *i*th dragon fly in *d*th dimension of the search space and N is the number of search agents.

The fitness function is evaluated based on the initial position values which is randomly generated between the lower and upper bounds of the variables. The weights for separation (s), alignment (a), cohesion(c), food (f) and enemy (e) factors for each dragonfly is initialized randomly. For updating the position and velocity of dragonflies separation, alignment and cohesion coefficients are calculated using Eqs. (2)–(4)

$$S_i = -\sum_{j=1}^{N} X - X_i \tag{2}$$

$$A_i = \frac{\sum_{i=1}^{N} V_i}{N} \tag{3}$$

$$C_i = \frac{\sum_{i=1}^{N} X_i}{N} - X \tag{4}$$

where $X_i$ and $V_i$ corresponds to the position and velocity of the *i*th individual. X corresponds to the position of the current individual and N denotes the number of neighbouring individuals.

Attraction towards food source,$F_i$ and distraction from enemies $E_i$ is calculated using Eqs. (5) and (6)

$$F_i = X^+ - X \tag{5}$$

$$E_i = X^- + X \tag{6}$$

where X is the position of the current individual and $X^+$denotes the food source and $X^-$denotes the enemy source.

The distance of the neighbourhood is calculated by calculating the Euclidean distance between all the dragonflies and selecting N of them. The distance, $r_{ij}$ is calculated by the Eq. (7)

$$r_{ij} = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \tag{7}$$

If dragon fly has at least one dragonfly in the neighbourhood the velocity of the dragonfly will be updated as per Eq. (8) analogous to velocity equation of PSO and the position of the dragonfly will be updated as per Eq. (9) which is analogous to position equation of PSO.

$$\triangle X_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\triangle X_t \tag{8}$$

$$X_{t+1} = X_t + \triangle X_{t+1} \tag{9}$$

If there is no dragonfly in the neighbourhood radius the position of the dragonfly will be updated using Levy Flight equation (Yang, 2010a) as given in Eq. (10). This improves the randomness, chaotic behaviour and global search capability of dragonflies.

$$X_{t+1} = X_t + Levy(d)X_t \tag{10}$$

The fitness function is then evaluated based on the updated position and velocities. The position updating process is continued till the stop condition is met.

### 2.2. PSO

PSO is swarm intelligence optimization technique based on the social behaviour of organisms living in swarms (Eberhart & Kennedy, 1995). Each individual in swarm is called as particle which can move freely to explore the problem hyperspace. Each particle is associated with position, velocity and fitness function. The velocity of the particle is updated based on its own history of best solution (pbest) as well as from the history of best solution so far found by all the particles in the population (gbest). The information obtained by the particle is shared with other particles in the population and finally the particle is guided towards the optimal solution. PSO is simple, easy to implement and have few parameters to be adjusted.

Consider N dimensional search space. Let the position and velocity of the *i*th particle in *k*th iteration be $x_k^i$and velocity $v_k^i$ respectively. The velocity and position of the particle in $(k+1)th$ iteration are updated as per Eqs. (11) and (12) respectively.

$$V_{k+1}^i = wV_k^i + C_1 r_1 (P_k^i - X_k^i) + C_2 r_2 (P_k^g - X_k^i) \tag{11}$$

$$X_{k+1}^i = X_k^i + V_{k+1}^i \tag{12}$$

where $w$ is the inertial weight, $C_1$and $C_2$ represents the cognitive and social parameters, $P_k^i$ and $P_k^g$ represents the pbest of the *i*th particle and gbest of the swarm upto *k*th iteration respectively. $\phi_1$ and $\phi_2$ represents random numbers generated in the range $[0 \quad 1]$.

## 3. Memory based Hybrid DA (MHDA)

For any optimization algorithm, proper balance between exploration and exploitation of the search space is necessary to achieve a global optimal solution. Exploration otherwise known as diversification involves global search in the search space and exploitation otherwise known as intensification involves search in a local region depending upon the current best solution. Too much of exploration and exploitation harmfully affects the performance of the algorithm by increasing the convergence time and increasing the chances to fall into local optima (Yang, 2014b). The conventional DA, operates on randomly generated initial population of search agents or dragonflies and dragon flies explore the search space using Levy flight. This random initialization and levy flight search process increases solution diversity and strengthens the exploration capability of algorithm. Further, DA has only few parameters to adjust and adaptive tuning of these swarming factors helps in balancing local and global search capabilities. However, DA lacks an internal memory which can keep track of previously obtained potential solutions. During the process, DA discards all the fitness values exceeding the global best and never keeps track on possible set of solutions which has a potential to converge to global optima. This weakens the exploitation capability of the DA tending to converge very slowly and sometimes stagnate at local optima. To avoid this, a novel hybrid algorithm based on DA and PSO is proposed. Two features are added to conventional DA algorithm to improvise its performance, they are (i) an internal memory to keep track of possible solutions which has a potential to converge to global optima (ii) iteration level hybridization with PSO which runs on this set of saved solutions.

### 3.1. Implementation of internal memory

With the addition of internal memory, each dragonfly is allowed to keep track of its co-ordinates in the problem hyperspace which are associated with fitness value. This is similar to the *pbest* concept in PSO. During each iteration, the fitness value of search agents in current population is compared with the best fitness value in that iteration. Better solutions are saved and a *DA-pbest* matrix is framed. Dragonflies are also made to track best value obtained so far by any dragon fly in the neighbourhood which is same as the *gbest* concept of PSO and is stored as *DA-gbest*. The concept of *pbest* and *gbest* in DA is novel and enhances the exploitation capability of DA. This feature of internal memory provides capability to escape from local optima and provides greater performance than conventional algorithm (Parouha & Das, 2016).

### 3.2. Iterative level hybridization with PSO

Iteration level hybridization is a straight forward approach of iteratively executing two algorithms in sequence to enhance the optimization performance (Ma, Simon, Fei, Shu, & Chen, 2014). Here DA with internal memory is used to converge the search space to more promising areas and PSO is then allowed to exploit the previously limited area to find better solutions. Due to balance between exploration offered by DA and exploitation capabilities offered by PSO, hybrid algorithm-MHDA performs better than the parent algorithms. PSO is then initialized with *DA-pbest* matrix and *DA-gbest*

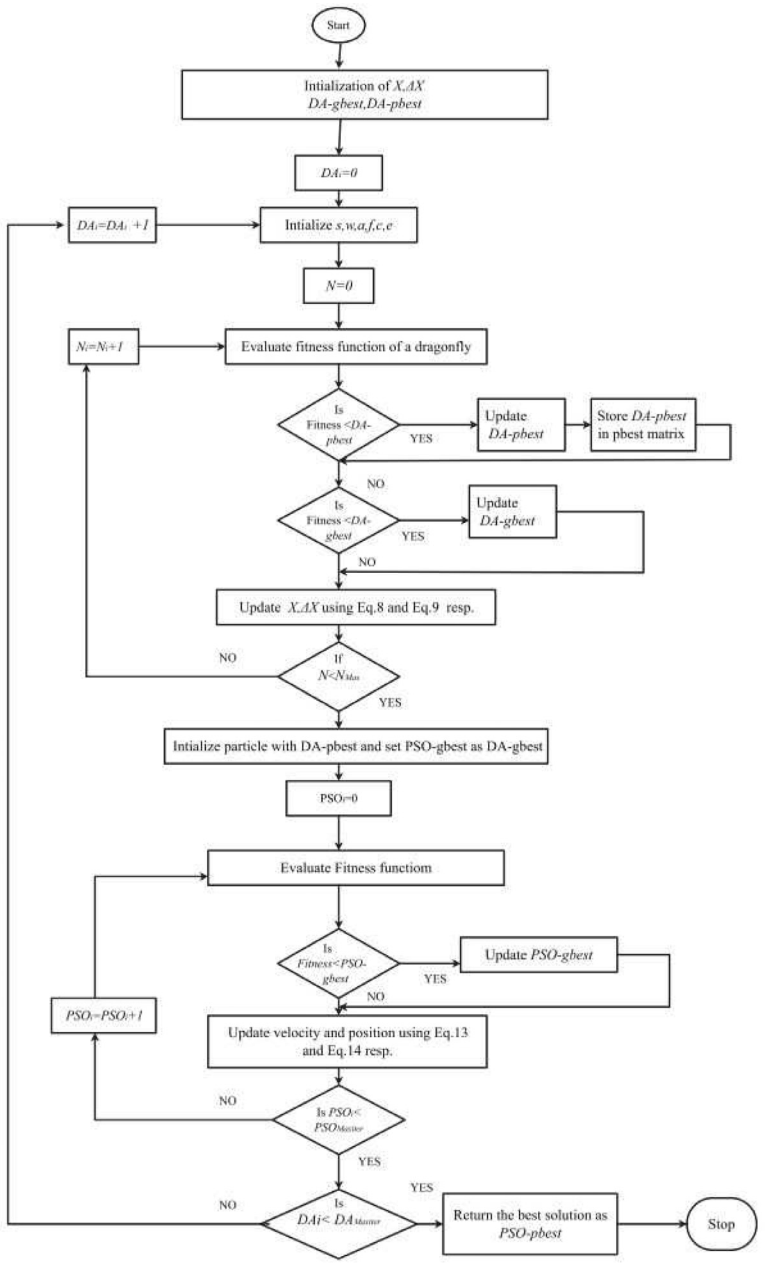
Start
Intialization of X,ΔX DA-gbest,DA-pbest
DAi=0
DAi=DAi +1
Intialize s,w,a,f,c,e
N=0
Ni=Ni+1
Evaluate fitness function of a dragonfly
Is Fitness <DA-pbest
YES
Update DA-pbest
Store DA-pbest in pbest matrix
NO
Is Fitness <DA-gbest
YES
Update DA-gbest
NO
Update X,ΔX using Eq.8 and Eq.9 resp.
NO
If N<NMax
YES
Intialize particle with DA-pbest and set PSO-gbest as DA-gbest
PSOi=0
Evaluate Fitness functiom
Is Fitness<PSO-gbest
YES
Update PSO-gbest
NO
PSOi=PSOi+1
Update velocity and position using Eq.13 and Eq.14 resp.
NO
Is PSOi< PSOMaxiter
YES
NO
Is DAi< DAMaxiter
YES
Return the best solution as PSO-pbest
Stop

**Fig. 1.** Flowchart of MHDA.

is set as the gbest of PSO (*PSO-gbest*). The position and velocity equations of PSO gets modified as

$$V^i_{k+1} = wV^i_k + C_1 r_1 (DA - pbest^i_k - X^i_k) + C_2 r_2 (DA - gbest^g_k - X^i_k) \tag{13}$$

$$X^i_{k+1} = X^i_k + V^i_{k+1} \tag{14}$$

where $DA - pbest^i_k$ is the *pbest* for *i*th particle of PSO and $DA - gbest^g_k$ is *gbest* of the swarm up to *k*th iteration for PSO.

Thus the MHDA combines the exploration features of DA in initial stage and exploitation capabilities of PSO in final stage to achieve global optimal solutions. The flowchart and pseudo-code of proposed optimization algorithm is given in Fig. 1 and Algorithm 1 respectively.

## 4. Performance evaluation

The efficiency of MHDA is proved by testing on standard benchmark functions and comparing its performance with other powerful optimization algorithms. The benchmark function suite used for analysis are explained in the Section 4.1. The experimental results and comparison with other algorithms is explained in Section 4.2. Section 4.3 presents the statistical test results and the analysis of MHDA algorithm is explained in Section 4.4. Section 4.5 explains the computational complexity of MHDA.

### 4.1. Benchmark function suite

To test the performance of MHDA, two test suites are considered. Suite I includes the function set used by the authors of conventional DA and recently proposed swarm based optimization techniques. This avoids studying the best parameter setting

**Algorithm 1** Pseudocode of MHDA.

```
Initialization set of parameters Maximum iteration (Max-iter),maximum number of search agents ($N_{\max}$) number of search agents(N), number of dimensions ($d$), upper boundand lower bound of variables
Initialize the dragonflies populations ($X$) - Initializethe step vectors ($\Delta X$)
while maximum iterations not done
    For each dragonfly
        Calculate fitness value
        if
        Fitness Value < DA-pbestin this iteration
            move the current value to DA-pbestmatrix
        end if
        if
        fitness value < DA-gbest
            set current value as DA-gbest
        end if
    end
    For each dragonfly
        Update the food source and enemy
        Update w, s, a, c, f, and e
        Calculate S, A, C, F, and E using Eqs. (2)–(6)
        Update neighbouring radius
        if a dragonfly has at least one neighbouring dragonfly
            Update velocity vector using Eq. (8)
            Update position vector using Eq. (9)
        else
            Update position vector using Eq. (10)
        end if
        Check and correct the new positions based on the boundariesof variables
    end
------------------Endof DA and Start of PSO------------------
    For each particle
        Initialize particle with DA-pbestmatrix
        Set PSO-gbest asDA-gbest
    end
    while maximum iterations or minimum error criteria is notattained
        For each particle
            Calculate fitness value if fitness value <PSO-pbest in history
            set current value as the new PSO-pbest
            end if
        end
        Choose the particle with the best fitness value of all thepparticles as the PSO-gbest
        For each particle
            Calculate particle velocity according Eq. (13)
            Update particle position according Eq. (14)
        end
    end while
------------------------End of PSO------------------------------
best-fitness = PSO-gbest
end while
```

of each algorithm separately and to conduct a fair comparison. In Suite-I, the performance of MHDA is compared with classical DA, PSO and recently proposed swarm based optimization algorithms such as Ant Lion Optimizer(ALO) (Mirjalili, 2015), Grey Wolf Optimizer(GWO) (Mirjalili, Mirjalili, & Lewis, 2014), Whale Optimization Algorithm (WOA) (Mirjalili & Lewis, 2016). We also intend to prove that MHDA is capable of giving competitive results when compared to recently developed high performance algorithms from different families, hence suite II is considered. In Suite-II, the performance of MHDA is compared with most powerful and standard algorithms such as Cuckoo Search (CS) (Mlakar, Jr., & Fister, 2016), Mean -Variance Mapping Optimization (MVMO) (Erlich, Rueda, Wildenhues, & Shewarega, 2014), Backtracking Search Optimization Algorithm (BSA) (Civicioglu, 2013) self adaptive variants of Differential Evolution (DE) such as JADE (Zhang & Sanderson, 2009), SaDE (Qin, Huang, & Suganthan, 2009) and jDE (Brest, Greiner, Boskovic, Mernik, & Zumer, 2006).

*Suite-I - Basic unconstrained benchmark functions*

Suite-I consist of 19 benchmark functions, out of which 16 are classical benchmark functions found in the literature (Mirjalili, 2015) and other 6 test problems are taken from the novel composite functions proposed in IEEE Swarm Intelligence Symposium 2005 (Liang, Suganthan, & Deb, 2005). The classical benchmark set is classified into two sets- unimodal functions ($F_1 - F_7$) and multimodal functions ($F_8 - F_{13}$). The details of unimodal and multimodal benchmark functions, range, and dimension is shown in Table 1. Unimodal test functions have one global optima and performance of algorithm on these test function reveals its exploitation and convergence capability. Multimodal functions have more than one global optimum in the presence of many local optima. The performance of algorithm on these test function reveals its exploration and local optima avoidance capability. The last set of functions called as composite functions($F_{14} - F_{19}$) comprises of combined, shifted, rotated, biased versions of algorithms and represents the complex search space by providing large number of local minima and changing the shape of search space domain. The details of composite functions ($F_{14} - F_{19}$) are given in the Appendix.

*Suite-II-CEC 2014 unconstrained benchmark functions*

Suite-II comprises 30 CEC 2014 functions with 30 dimensions (30D). The detailed explanation of different functions can be found in the technical report (J.Liang, 2014). CEC 2014 test functions can be categorized into four types-unimodal($F_1 - F_3$), multimodal ($F_4 - F_{16}$), hybrid functions($F_{17} - F_{22}$) and composition functions ($F_{23} - F_{30}$). Unimodal functions in this set are non-separable and rotated where as multimodal functions are either separable or non-separable but are shifted or rotated. Hybrid functions are created by randomly dividing the variables into different sub components and then different basic components are used for different sub components. Composite functions in this set are created by the combination of two or more hybrid functions.

### 4.2. Experimental results

All the experiments were executed on personal computer (Core i7, 3.4 GHz, 32GB RAM) using MATLAB. The performance of MHDA was compared with other algorithms in terms of mean and standard deviation of objective function value and objective function error value on Suite-I and on Suite II respectively.

#### 4.2.1. On Suite-I

In this section the performance of MHDA on benchmark function Suite-I is explained. For a fair comparison among the algorithms, maximum number of iteration and search agents was set to 1000 and 30 respectively as followed in the literature (Mirjalili, 2015). The levy flight constant and inertial weights of MHDA[$w_{\max}, w_{\min}$] was set to 1.5 and [0.9, 0.2] respectively. Each function was run independently for 30 runs and mean and standard deviation (std) for unimodal, multimodal and composite functions are reported in Table 2. The stopping criterion was set to maximum number of iterations. The results of other algorithms have been taken from the reference (Mirjalili, 2015; Mirjalili & Lewis, 2016).

**Table 1**
Description of unimodal functions.

| Function | Dim | Range | $f_{\min}$ |
|---|---|---|---|
| Unimodal Functions | | | |
| $F_1(x) = \sum_{i=1}^{n} x_i^2$ | 30 | [−100 100] | 0 |
| $F_2(x) = \Sigma_{i=1}^{n} \lvert x_i \rvert + \Pi_{i=1}^{n} \lvert x_i \rvert$ | 30 | [−100 100] | 0 |
| $F_3(x) = \Sigma_{i=1}^{n} (\Sigma_{j-1}^{i} X_j)^2$ | 30 | [−100 100] | 0 |
| $F_4(x) = max\{\lvert x_i \rvert, 1 \le i \le n\}$ | 30 | [−100 100] | 0 |
| $F_5(x) = \Sigma_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$ | 30 | [−30 −30] | 0 |
| $F_6(x) = \Sigma_{i=1}^{n} [(x_i + 0.5)]^2$ | 30 | [−100 100] | 0 |
| $F_7(x) = \Sigma_{i=1}^{n} x_i^4 + random(0\ 1)$ | 30 | [−1.28 1.28] | 0 |
| Multimodal Functions | | | |
| $F_8(x) = \Sigma_{i=1}^{n} - x_i sin(\sqrt{\lvert x_i \rvert})$ | 30 | [−500 500] | −418.982D |
| $F_9(x) = \Sigma_{i=1}^{n} [x_i^2 - 10cos(2\Pi x_i) + 10]$ | 30 | [−5.12 5.12] | 0 |
| $F_{10}(x) = -20exp(-0.2\sqrt{\frac{1}{n}\Sigma_{i=1}^{n} x_i^2} - exp\left(\frac{1}{n}\Sigma_{i=1}^{n} cos(2\Pi x_i)\right) + 20 + e$ | 30 | [−32 32] | 0 |
| $F_{11}(X) = \frac{1}{4000}\Sigma_{i=1}^{n} x_i^2 - \Pi_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$ | 30 | [−600 600] | 0 |
| $F_{12}(x) = \frac{\Pi}{n}\{10\sin(\Pi y_1) + \Sigma_{i=1}^{n-1} (y_i - 1)^2 [1 + 10\sin^2(\Pi y_{i+1})] + (y_n - 1)^2\} + \Sigma_{i=1}^{n} u(x_i, 10, 100, 4)$ | 30 | [−50 50] | 0 |
| $F_{13}(x) = 0.1\{sin^2(3\Pi x_1) + \Sigma_{i=1}^{n} (x_i - 1)^2 [1 + \sin^2(3\Pi x_i + 1) + (x_n - 1)^2 [1 + sin^2(2\Pi x_n)]\} + \Sigma_{i=1}^{n} u(x_i, 5, 100, 4)]$ | 30 | [−1.28 1.28] | 0 |

**Table 2**
Comparison between MHDA and other algorithms on optimizing Suite-I benchmark functions in terms of mean and std.

| F | Meas. | MHDA | DA | ALO | GWO | WOA | PSO |
|---|---|---|---|---|---|---|---|
| $F_1$ | Mean | **4.07E−42** | 5.150E−07 | 2.59E−10 | 6.59E−28 | 1.41E−30 | 2.70E−09 |
| | Std | **2.22E−41** | 2.82E−06 | 1.65E−10 | 6.34E−05 | 4.91E−30 | 1.00E−09 |
| $F_2$ | Mean | 6.62E−15 | 4.818E−06 | 1.842E−06 | 7.18E−17 | **1.06E−21** | 7.15E−05 |
| | Std | 3.61E−14 | 2.50E−05 | 6.58E−07 | 2.91E−02 | **2.39E−21** | 2.26E−05 |
| $F_3$ | Mean | **2.55E−50** | 5.366E−07 | 6.068E−10 | 3.29E+06 | 5.39E−07 | 4.71E−06 |
| | Std | **1.3E−49** | 2.939E−06 | 6.34E−10 | 79.14958 | 2.93E−06 | 1.49E−06 |
| $F_4$ | Mean | 4.989E−05 | 1.349E−04 | **1.361E−08** | 5.61E−07 | 7.26E−02 | 3.25E−07 |
| | Std | 2.73E−04 | 4.569E−04 | **1.81E−09** | 1.315E+00 | 3.98E−11 | 1.02E−08 |
| $F_5$ | Mean | **3.34E−22** | 6.71E−01 | 3.46E−01 | 2.65E+01 | 27.87E+00 | 1.23E−01 |
| | Std | **5.67E−22** | 3.66E+00 | 0.109584 | 69.90499 | 7.636E−01 | 2.16E−01 |
| $F_6$ | Mean | **0.00E+00** | 9.047E−06 | 2.562E−10 | 8.166E−01 | 3.116E+00 | 5.23E−07 |
| | Std | **0.00E+00** | 3.305E−05 | 1.09E−10 | 1.26E−04 | 5.324E−01 | 2.74E−06 |
| $F_7$ | Mean | **5.25E−05** | 4.5E−04 | 4.29E−03 | 2.22E−02 | 1.425E−03 | 1.39E+00 |
| | Std | **5.02E−05** | 5.71E−04 | 5.08E−03 | 1.003E−01 | 1.15E−03 | 0.001269 |
| $F_8$ | Mean | −2957.34 | −3932.76 | −**2247.86** | −6123.1 | −5080.76 | −4841.29 |
| | Std | 3.86E+02 | 2.18E+02 | **5.29E+03** | 4.08E+04 | 6.95E+00 | 1.15E+03 |
| $F_9$ | Mean | **5.901E−07** | 3.36E−02 | 7.71E−06 | 3.12E−01 | 0.00E+00 | 2.78E−01 |
| | Std | **3.23E−06** | 1.81E−01 | 8.45E−06 | 4.74E+01 | 0.00E+00 | 2.18E−01 |
| $F_{10}$ | Mean | **6.34E−15** | 2.66E−04 | 3.71E−15 | 1.06E−13 | 7.40E+00 | 1.11E−09 |
| | Std | **2.720E−14** | 8.59E−04 | 1.5E−15 | 7.78E−02 | 9.90E+00 | 2.39E−11 |
| $F_{11}$ | Mean | **2.397E−04** | 3.83E−03 | 1.86E−02 | 4.49E−03 | 2.89E−04 | 2.73E−01 |
| | Std | **2.25E−02** | 7.154E−02 | 9.54E−03 | 6.66E−03 | 1.59E−03 | 2.04E−01 |
| $F_{12}$ | Mean | **2.34E−31** | 7.48E−04 | 9.75E−12 | 5.34E−02 | 3.40E−01 | 9.42E−09 |
| | Std | **4.45E−47** | 3.75E−04 | 9.33E−12 | 2.07E−02 | 2.15E−01 | 2.31E−10 |
| $F_{13}$ | Mean | **1.39E−32** | 1.06E−03 | 2.01E−11 | 6.54E−01 | 1.88E+00 | 1.35E−07 |
| | Std | **5.57E−48** | 3.99E−04 | 1.13E−11 | 4.47E−03 | 2.66E−01 | 2.88E−08 |
| $F_{14}$ | Mean | **5.75E−15** | 1.04E+02 | 1.51E−04 | 4.38E+01 | 5.68E−01 | 100 |
| | Std | **2.85E−04** | 9.12E+01 | 3.82E−04 | 6.98E+01 | 5.05E−01 | 8.16E+01 |
| $F_{15}$ | Mean | 1.40E+02 | 2.13E+02 | **1.45E+01** | 9.18E+01 | 7.53E+01 | 1.55E+02 |
| | Std | 2.43E+01 | 1.27E+05 | **3.22E+01** | 9.55E+01 | 4.31E+01 | 1.13E+01 |
| $F_{16}$ | Mean | **1.00E+01** | 5.58E+02 | 1.75E+02 | 6.14E+01 | 5.56E+01 | 1.72E+01 |
| | Std | **3.44E+01** | 1.65E+02 | 4.65E+01 | 6.86E+01 | 2.18E+01 | 3.27E+01 |
| $F_{17}$ | Mean | 1.00E+02 | 2.20E−03 | 3.16E+02 | **1.23E+01** | 5.38E+01 | 3.14E+02 |
| | Std | 5.63E−03 | 4.63E−03 | 1.30E+01 | **1.63E+02** | 2.16E+01 | 2.00E+01 |
| $F_{18}$ | Mean | 3.03E+02 | 2.50E+02 | 4.41E+01 | **1.02E+01** | 7.78E+01 | 8.34E+01 |
| | Std | 8.88E+00 | 1.85E+02 | 1.66E+00 | **8.12E+01** | 5.22E+01 | 1.01E+02 |
| $F_{19}$ | Mean | **5.00E+02** | 6.79E+02 | 5.003E+02 | 4.31E+01 | 5.78E+01 | 8.61E+02 |
| | Std | **1.36E−03** | 1.99E+02 | 2.06E−01 | 8.44E+01 | 3.44E+01 | 1.25E+02 |

#### 4.2.2. *On Suite-II*

The experiments are done on 30 CEC 2014 benchmark functions with 30 dimensions. The parameter setting of MHDA was kept same as in Suite-I. The parameter setting and results of other algorithms were taken from the reference (Chen, Zou, Lu, & Wang, 2017; Mlakar et al., 2016). The population size was set to 50 and maximum number of function evaluation ($MaxFE = D * 10^3$) is set as stopping criteria where D is the number of dimensions. The algorithm was run independently 51 times. The mean and standard deviation(std) of function error values between the best fitness value and true optimal value in each independent runs are reported in Table 3.

### 4.3. *Statistical results*

Mean and standard deviation of results give general idea about the performance of the algorithm. In order to prove that results are generated not by chance, statistical tests must be also carried out.

**Table 3**
Comparison between MHDA and other algorithms on optimizing Suite-II benchmark functions in terms of mean and std.

| F | Meas. | MHDA | CS | MVMO | BSA | JADE | SaDE | jDE |
|---|---|---|---|---|---|---|---|---|
| $F_1$ | Mean | 3.20E+03 | 3.50E+07 | **1.07E−03** | 2.04E+01 | 6.09E+02 | 3.73E+03 | 6.12E+04 |
| | Std | 3.02E+03 | 2.49E+07 | **1.09E−03** | 1.56E−02 | 1.18E+03 | 3.26E+03 | 7.64E+04 |
| $F_2$ | Mean | **0.00E+00** | 1.95E+07 | 2.38E−05 | 1.62E+01 | **0.00E+00** | **0.00E+00** | 2.27E−15 |
| | Std | **0.00E+00** | 5.49E+07 | 1.19E−05 | 9.69E−01 | **0.00E+00** | **0.00E+00** | 7.87E−15 |
| $F_3$ | Mean | **0.00E+00** | 3.10E+04 | 1.11E−03 | 4.19E−03 | 9.86E−04 | **0.00E+00** | 4.09E−14 |
| | Std | **0.00E+00** | 1.36E+04 | 1.03E−03 | 1.32E−02 | 5.95E−03 | **0.00E+00** | 2.60E−14 |
| $F_4$ | Mean | **0.00E+00** | 2.03E+02 | **0.00E+00** | 2.93E+00 | **0.00E+00** | **0.00E+00** | 8.53E+00 |
| | Std | **0.00E+00** | 6.69E+01 | **0.00E+00** | 1.46E+00 | **0.00E+00** | **0.00E+00** | 2.16E+01 |
| $F_5$ | Mean | **2.00E+01** | 2.00E+01 | **2.00E+01** | 5.95E+01 | 2.03E+01 | 2.03E+01 | 2.03E+01 |
| | Std | **1.57E−02** | 2.28E−03 | **2.00E+01** | 7.94E+00 | 3.23E−02 | 4.03E−02 | 3.26E−02 |
| $F_6$ | Mean | **1.76E+00** | 3.23E+01 | 3.62E+00 | 3.22E+01 | 9.15E+00 | 1.49E+01 | 5.31E+00 |
| | Std | **2.87E+00** | 3.27E+00 | 3.04E+00 | 6.57E+00 | 2.21E+00 | 9.42E−01 | 4.04E+00 |
| $F_7$ | Mean | **0.00E+00** | 1.79E+00 | 2.99E−03 | 2.56E+03 | **0.00E+00** | **0.00E+00** | 2.96E−04 |
| | Std | **0.00E+00** | 2.19E+00 | 0.00E+00 | 2.56E+02 | **0.00E+00** | **0.00E+00** | 1.48E−03 |
| $F_8$ | Mean | **0.00E+00** | 1.71E+02 | 8.58E−01 | 4.37E−01 | **0.00E+00** | **0.00E+00** | 1.19E−01 |
| | Std | **0.00E+00** | 3.46E+01 | 9.95E−01 | 7.85E−02 | **0.00E+00** | **0.00E+00** | 3.30E−01 |
| $F_9$ | Mean | 3.00E+01 | 2.80E+02 | 2.51E+01 | **2.84E−01** | 2.62E+01 | 3.58E+01 | 3.81E+01 |
| | Std | 1.82E+01 | 5.16E+01 | 2.39E+01 | **4.68E−02** | 4.96E+00 | 7.01E+00 | 5.71E+00 |
| $F_{10}$ | Mean | 1.10E+03 | 2.66E+03 | 1.79E+01 | **2.45E−01** | 8.16E−03 | 1.11E+00 | 3.17E+00 |
| | Std | 8.24E+02 | 5.34E+02 | 9.76E+00 | **4.02E−02** | 1.18E−02 | 2.02E+00 | 3.18E+00 |
| $F_{11}$ | Mean | 1.41E+02 | 4.13E+03 | 1.54E+03 | **7.06E+00** | 1.67E+03 | 2.28E+03 | 2.71E+03 |
| | Std | 4.35E+02 | 5.35E+02 | 1.59E+03 | **1.07E+00** | 2.13E+02 | 3.45E+02 | 2.75E+02 |
| $F_{12}$ | Mean | 1.44E−01 | 5.11E−01 | **7.21E−02** | 1.07E+01 | 2.67E−01 | 4.59E−01 | 4.77E−01 |
| | Std | 7.19E−02 | 2.56E−01 | **6.24E−02** | 2.71E−01 | 3.57E−02 | 5.23E−02 | 5.41E−02 |
| $F_{13}$ | Mean | 4.59E−01 | 4.81E−01 | **1.57E−01** | 1.54E+05 | 2.20E−01 | 3.02E−01 | 2.84E−01 |
| | Std | 1.23E−01 | 1.17E−01 | **1.62E−01** | 8.75E+04 | 3.25E−02 | 3.69E−02 | 3.55E−02 |
| $F_{14}$ | Mean | 2.04E−01 | 3.08E−01 | **1.99E−01** | 9.10E+02 | 2.41E−01 | 2.68E−01 | 3.02E−01 |
| | Std | 3.33E−01 | 5.64E−02 | **1.99E−01** | 1.05E+03 | 3.18E−02 | 1.40E−01 | 4.15E−02 |
| $F_{15}$ | Mean | **2.33E+00** | 9.80E+01 | 2.86E+00 | 6.91E+00 | 3.20E+00 | 4.86E+00 | 5.36E+00 |
| | Std | **7.67E−01** | 3.02E+01 | 2.69E+00 | 6.25E−01 | 4.55E−01 | 4.17E−01 | 7.43E−01 |
| $F_{16}$ | Mean | 9.50E+00 | 1.27E+01 | 1.02E+01 | 1.68E+02 | **9.30E+00** | 1.03E+01 | 1.03E+01 |
| | Std | 1.18E+00 | 5.01E−01 | 9.84E+00 | 1.91E+02 | **4.61E−01** | 3.42E−01 | 3.23E−01 |
| $F_{17}$ | Mean | **4.81E+02** | 1.48E+06 | 9.01E+02 | 6.20E+03 | 1.91E+04 | 8.55E+02 | 1.62E+03 |
| | Std | **5.32E+02** | 1.21E+06 | 1.03E+03 | 3.02E+03 | 1.08E+05 | 2.80E+02 | 1.49E+03 |
| $F_{18}$ | Mean | 3.74E+01 | 7.67E+03 | 2.89E+01 | 1.79E+02 | 1.14E+02 | 4.92E+01 | **1.86E+01** |
| | Std | 2.08E+01 | 6.70E+03 | 2.08E+01 | 8.11E+01 | 1.97E+02 | 2.57E+01 | **1.04E+01** |
| $F_{19}$ | Mean | 1.12E+01 | 5.33E+01 | 3.08E+00 | 3.15E+02 | **4.48E+00** | 5.26E+00 | 4.97E+00 |
| | Std | 1.87E+01 | 3.63E+01 | 3.02E+00 | 2.92E−07 | **7.56E−01** | 1.15E+00 | 9.61E−01 |
| $F_{20}$ | Mean | 3.67E+02 | 3.93E+04 | 1.09E+02 | 2.27E+02 | 3.11E+03 | 1.85E+01 | **1.36E+01** |
| | Std | 4.08E+02 | 2.20E+04 | 5.69E+01 | 2.42E+00 | 3.01E+03 | 4.14E+00 | **6.64E+00** |
| $F_{21}$ | Mean | 7.06E+02 | 3.54E+05 | 4.67E+02 | **2.07E+02** | 1.33E+04 | 4.31E+02 | 2.98E+02 |
| | Std | 1.05E+03 | 3.48E+05 | 4.89E+02 | **6.48E−01** | 4.12E+04 | 1.32E+02 | 2.25E+02 |
| $F_{22}$ | Mean | 2.70E+02 | 9.47E+02 | 1.45E+02 | **1.00E+02** | 1.44E+02 | 1.65E+02 | 1.38E+02 |
| | Std | 1.80E+02 | 3.31E+02 | 1.46E+02 | **5.83E−02** | 7.74E+01 | 7.11E+01 | 5.38E+01 |
| $F_{23}$ | Mean | **3.10E+02** | 3.29E+02 | 3.15E+02 | 4.09E+02 | 3.15E+02 | 3.15E+02 | 3.15E+02 |
| | Std | **4.00E+01** | 7.51E+00 | 3.15E+02 | 3.71E+00 | 4.01E−13 | 0.00E+00 | 0.00E+00 |
| $F_{24}$ | Mean | **2.24E+02** | 2.78E+02 | 2.25E+02 | 8.77E+02 | 2.25E+02 | 2.25E+02 | 2.26E+02 |
| | Std | **1.80E+01** | 3.11E+01 | 2.25E+02 | 1.66E+01 | 3.60E+00 | 4.31E+00 | 3.34E+00 |
| $F_{25}$ | Mean | 2.10E+02 | 2.23E+02 | 2.03E+02 | 1.41E+03 | 2.03E+02 | **2.03E+02** | 2.04E+02 |
| | Std | 6.91E+00 | 9.39E+00 | 2.03E+02 | 1.89E+02 | 1.13E+00 | **5.52E−01** | 8.81E−01 |
| $F_{26}$ | Mean | 1.000E+02 | 1.00E+02 | 1.00E+02 | 2.55E+03 | 1.02E+02 | **1.00E+02** | 1.00E+02 |
| | Std | 4.720E−02 | 1.63E−01 | 1.00E+02 | 7.49E+02 | 1.39E+01 | **3.55E−02** | 5.05E−02 |
| $F_{27}$ | Mean | 4.01E+02 | 4.27E+02 | 4.01E+02 | 5.12E+06 | **3.35E+02** | 5.46E+02 | 4.01E+02 |
| | Std | 4.01E+02 | 1.96E+01 | 4.01E+02 | 4.10E+06 | **4.68E+01** | 1.11E+02 | 5.44E+01 |
| $F_{28}$ | Mean | 1.52E+03 | 3.49E+03 | 8.77E+02 | **8.34E−01** | 7.96E+02 | 8.08E+02 | 8.38E+02 |
| | Std | 5.77E+02 | 5.48E+02 | 8.78E+02 | **1.48E+00** | 4.63E+01 | 3.78E+01 | 2.99E+01 |
| $F_{29}$ | Mean | 7.78E+02 | 5.44E+05 | 7.36E+02 | **5.74E−04** | 8.28E+02 | 8.41E+05 | 8.66E+02 |
| | std | 7.5E+02 | 2.61E+06 | 7.42E+02 | **9.44E−04** | 3.27E+02 | 2.66E+06 | 1.62E+02 |
| $F_{30}$ | Mean | 2.37E+03 | 2.49E+04 | 2.00E+03 | **9.83E+01** | 1.66E+03 | 2.34E+03 | 2.79E+03 |
| | Std | 2.80E+03 | 2.26E+04 | 2.08E+03 | **2.96E+01** | 7.61E+02 | 1.38E+03 | 1.22E+03 |

The statistical significance of experimental results on Suite-I and Suite-II is obtained by performing Friedman's test and Wilcoxon ranksum test. Friedman test is a commonly used non parametric statistical method to rank the performance of the algorithms. Friedman's test aims to find whether any significant difference exist between the results of different algorithms. It is based on null hypothesis that there is no variation in the performance of all algorithms (Demsar, 2006). The best performing algorithm gets lowest rank while the worst performing algorithm gets the highest rank. The average rank obtained by each algorithm on all test functions is calculated for determining Friedman's statistic (Li et al., 2015). Friedman statistic is then compared with $\chi^2$ (chi-square) distribution values with $k-1$ degrees of freedom, where $k$ is the number of algorithms compared. If the $p$ value returned by this comparison test is found to be less than or equal to level of significance, null hypothesis is rejected indicating the there exist significant differences between the performance of algorithms. Friedman's test is then followed by post-hoc analysis to test the pair wise comparison of algorithms using Wilcoxon's ranksum test (Derrac, Garcia, Molina, & Herrera, 2011). The lowest ranked algorithm by Friedman's test is used as the control method for post-hoc analysis. The summary of statistical results on Suite-I and Suite-II bench-

**Table 4**
Summary of statistical results on Suite-I.

| Friedman's Test | | Wilcoxon ranksum test | | | |
|---|---|---|---|---|---|
| Algorithm | Rank | MHDA Vs | + | – | Not Sgn |
| MHDA | 2.263 | DA | 2 | 16 | 1 |
| DA | 4.578 | ALO | 4 | 15 | 0 |
| ALO | 3.632 | GWO | 7 | 12 | 0 |
| GWO | 4.842 | WOA | 8 | 11 | 0 |
| WOA | 3.563 | PSO | 3 | 16 | 0 |
| PSO | 4.210 | | | | |

**Table 5**
Summary of statistical results on Suite-II.

| Friedman's Test | | Wilcoxon ranksum test | | | |
|---|---|---|---|---|---|
| Algorithm | Rank | MHDA Vs | + | – | Not Sgn |
| MHDA | 2.230 | CS | – | 24 | 6 |
| CS | 5.400 | JADE | 8 | 16 | 4 |
| JADE | 2.933 | MVMO | 11 | 13 | 6 |
| MVMO | 2.700 | SaDE | 7 | 16 | 7 |
| SaDE | 3.226 | jDE | 8 | 20 | 2 |
| jDE | 3.833 | BSA | 11 | 19 | 0 |
| BSA | 4.800 | | | | |

mark functions is shown in Tables 4 and 5 respectively. + indicates significantly better, − indicates significantly worse and Not Sgn indicates non-significant results produced by given algorithm than MHDA.

From the summary of statistical results of Suite-I and Suite-II, MHDA was the best performing algorithm among all the compared algorithms. Considering 5% level of significance MHDA received lowest rank of 2.263 in Suite-I test functions and 2.23 in Suite-II test functions.WOA was the second best performing algorithm on Suite-I outperforming MHDA in 11 cases. MVMO, which was one of the best qualified algorithms in CEC competition 2014 was a major competitor of MHDA in Suite-II and. Its performances were better than MHDA in 16 cases, worse than MHDA in 9 cases and gave non-significant results in 5 cases. Therefore, MHDA is proved to be a highly competitive optimization algorithm and can be used for solving hardest optimization problems.

### 4.4. Analysis of MHDA

The following observations can be made from the experimental results of Suite-I:

- Unimodal functions($F_1 - F_7$) — The proposed algorithm outperforms other algorithms in five out of seven benchmark problems. In function $F_2$ MHDA becomes third best and in $F_4$ MHDA becomes fourth best out of the six algorithms compared.
- Multimodal Functions ($F_8 - F_{13}$) — The proposed algorithm outperforms other algorithms in five out of six cases. In function $F_8$ MHDA becomes the second best performing algorithm.
- Composite Functions ($F_{14} - F_{19}$) — MHDA outperforms other algorithms in four out of six cases.

The following observations can be made from the experimental results of Suite-II:

- Unimodal functions ($F_1 - F_3$) — For this group functions the proposed MHDA gives best performance in $F_2$and $F_3$
- Multimodal functions ($F_4$–$F_{16}$) — The proposed MHDA performs better on six functions namely$F_4$,$F_5$, $F_6$, $F_7$, $F_8$and $F_{15}$ while it becomes the second performing algorithm for functions $F_{11}$,$F_{12}$, $F_{14}$, $F_{16}$.
- Hybrid functions ($F_{17}$–$F_{22}$) — For this group function the proposed MHDA gives best results in $F_{17}$and for other functions it is marginally worse than the best performing algorithm.
- Composition functions ($F_{23}$–$F_{30}$) — In this group the proposed MHDA performs better than all other algorithms on three functions namely $F_{23}$,$F_{24}$, $F_{26}$. It is second best in functions $F_{27}$ and $F_{29}$.

Performance of MHDA on unimodal functions shows the exploitation capability of MHDA which helps to converge rapidly and exploit accurately. Integrated internal memory and iterative level hybridization with PSO improves the exploitation capability of MHDA. The superior performance of MHDA on multimodal functions owes to the random initialization and levy flight search process followed in DA. From the results of composite functions, it is evident that MHDA succeeds in avoiding local optima by properly balancing the exploration and exploitation capabilities. MHDA outperformed most of functions in Suite-I while giving competent results in Suite-II.

The mechanism behind the exploration and exploitation of MHDA is graphically represented by tracking the path of search agents in the search space. The Unimodal function $F_1$, multimodal function $F_9$ and composite function $F_{14}$ from Suite-I are solved by 10 search agents for 100 iterations to explain the search process and convergence behaviour of MHDA. It has been found that search agents in MHDA tends to explore the search space very widely and then gradually converges to a point. This is because of exploration capability of DA provided in the first phase of MHDA and subsequent exploitation provided by iterative level hybridization of PSO working on$DA - pbest$ matrix. The position of potential search agents nearer to the food are saved in this memory. This helps in attracting other search agents when exploring the search space. When the search agents approach near global optimal solution they exploit very slowly on the promising area of search solution with embedded PSO operators in MHDA. This is clearly shown in Fig. 2 where the '+' indicates the search agents for DA and green dots indicates the search agents for PSO.

The convergence curves of MHDA is compared with the other six algorithms of Suite-I and is provided in Fig. 3. All the algorithms were executed from the same initial population. It is observed that the convergence accelerates with increase in iterations. MHDA exhibits a rapid convergence in all three cases, which is due to its powerful global search mechanism in the initial stage and local search on best saved positions in the search space. From the Fig. 3 it has been found that MHDA outperforms other algorithms and converges very quickly to global minima or near global minima.

### 4.5. Computational complexity of MHDA

MHDA is realized by combining conventional DA and PSO algorithm. Computational complexity depends on the structure and implementation of algorithm. The overall complexity can be estimated as follows. MHDA consist of four major steps (*i*) Fitness calculation of dragonflies (*ii*) Updation of dragonflies (*iii*) Fitness Calculation of particle (*iv*)Updation of particle. Assuming $N$ as the number of search agents,$t$ as the number of iterations the complexity of first two steps can be estimated as $O(N^2t)$. Considering$K$ as the number of search-agents qualified for being in $DA - pbest$ matrix, the complexity of (*iii*) and (*iv*) together can be estimated as $O(K^2t)$ where $K < N$. Therefore the overall complexity of MHDA can be estimated as $O\left(N^2t + K^2t\right)$. Usually number of search agents required for optimizing problem is small (N = 20 or 40), and t is large (1000 or 2000), the computation cost is relatively inexpensive because the algorithm complexity is linear in terms of t. Even though the complexity of MHDA is more than conventional DA at the same maximum iteration, MHDA can find optimal solutions in lesser number of iterations before it reaches maximum iteration

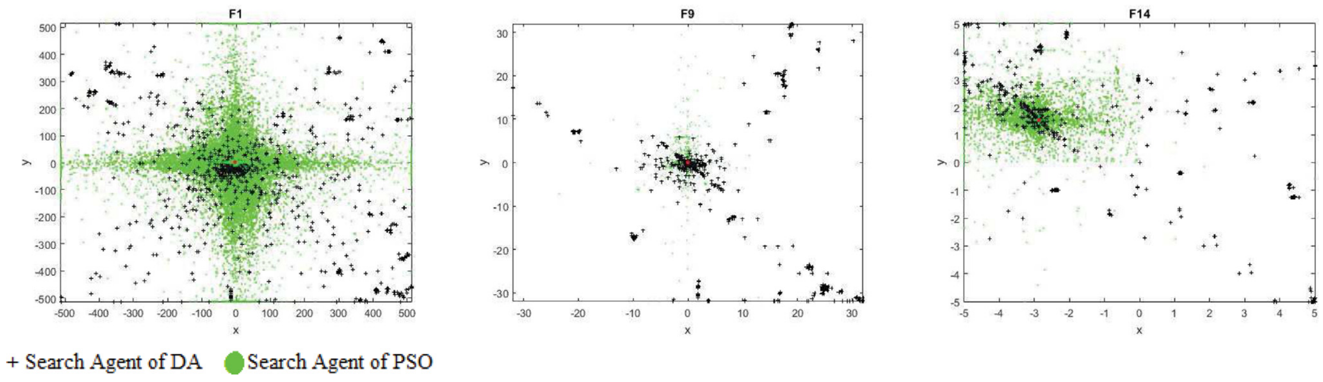

**Fig. 2.** Search process of MHDA.

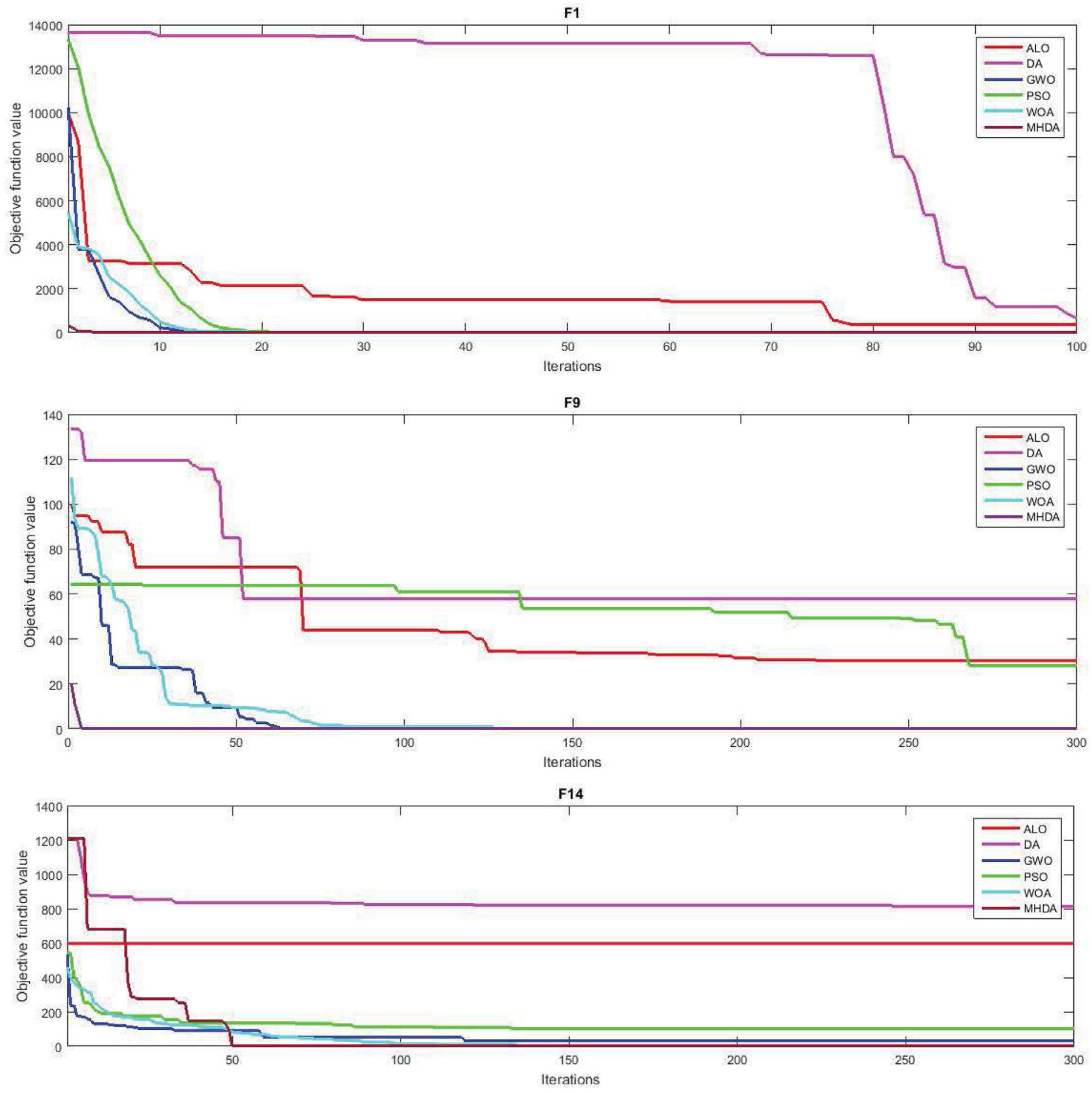

**Fig. 3.** Convergence plot of MHDA.

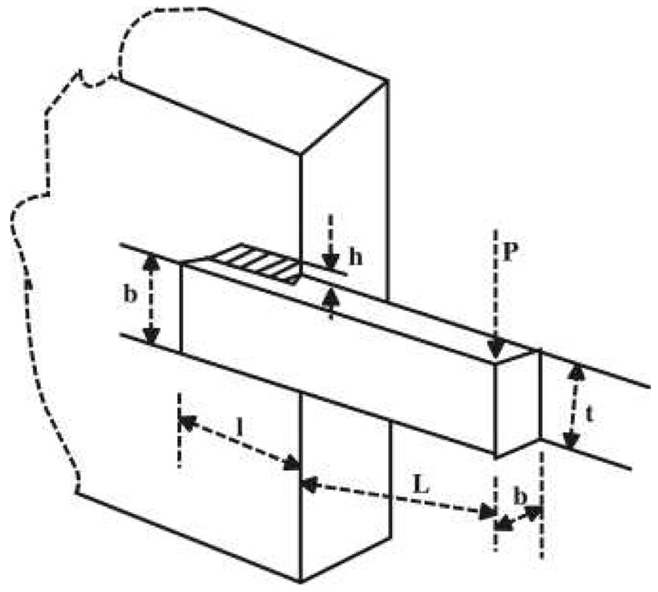

**Fig. 4.** Welded beam design problem.

and with greater accuracy. So, effectively the actual computation time will be reduced relative to conventional DA.

## 5. Application of MHDA

The competence of MHDA in solving real world problems especially non- linear constrained problems is demonstrated by testing on standard engineering design problems and comparing the results with other optimization algorithms. Three well known, engineering problems such as welded beam design, pressure vessel and motor design benchmark study are considered for testing MHDA algorithm. The constraints are usually handled by penalty functions. The idea of penalty functions is to transform a constrained optimization problem into an unconstrained one by adding (or subtracting) a certain value to/from the objective function based on the amount of constraint violation present in a certain solution (Pillo & Grippo, 1989). In this paper death penalty is used for discarding infeasible solutions during optimization.

### *5.1. Welded beam design*

The welded beam design problem, a standard benchmark study, aims at minimizing the fabrication cost of the welded beam by finding feasible set of four structural parameters of the beam: the thickness of the weld ($h$),length of the clamped bar($l$), height of the bar ($t$) and thickness of bar($b$). The related constraints are shear stress($\tau$), bending stress in the beam ($\theta$),buckling load ($P_c$),and end deflection of the beam ($\delta$). The variable vector (in inches) can be written as $\vec{X} = [\vec{x_1}, \vec{x_2}, \vec{x_3}, \vec{x_4}]$ where $\vec{x_1}$, $\vec{x_2}$, $\vec{x_3}$ and $\vec{x_4}$ represents $h$, $l$, $t$ and $b$ respectively. The welded beam structure shown in Fig. 4 is taken from Rao (2009).

The mathematical formulation of the objective function along with the constraints is formulated as follows

Minimize

$$f(\vec{X}) = 1.10471 x_2 x_1^2 + 0.04811 x_3 x_4 (14 + x_2) \tag{15}$$

subject to constraints,

$$g_1(\vec{X}) = \tau(\vec{X}) - \tau_{max} \le 0 \tag{16}$$

$$g_2(\vec{X}) = \sigma(\vec{X}) - \sigma_{max} \le 0 \tag{17}$$

$$g_3(\vec{X}) = \delta(\vec{X}) - \delta_{max} \le 0 \tag{18}$$

$$g_4(\vec{X}) = x_1 - x_4 \le 0 \tag{19}$$

$$g_5(\vec{X}) = P - P_c(\vec{X}) \le 0 \tag{20}$$

$$g_6(\vec{X}) = 0.125 - x_1 \le 0 \tag{21}$$

$$g_7(\vec{X}) = 1.1047 x_1^2 + 0.04811 x_3 x_4 (14 + x_2) - 5 \le 0 \tag{22}$$

where $\tau(\vec{X}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$

$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J},\ M = P(L + \frac{x_2)}{2}$

$R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1+x_3}{2})^2},$

$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\},$

$\sigma(\vec{X}) = \frac{6PL}{x_4x_3^2},\ \delta(\vec{X}) = \frac{6PL^3}{Ex_3^2x_4}$

$P_c(\vec{X}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$

$P = 6000$ lb, $L = 14$ in, $\delta_{\max} = 0.25$ in,

$E = 30E6$ psi, $G = 12E6$ psi, $\tau_{\max} = 13,600$ psi, $\sigma_{\max} = 30,000$ psi

$$\begin{aligned} &0.1 \le x_1 \le 2 \\ &0.1 \le x_2 \le 10 \\ &0.1 \le x_3 \le 10 \\ &0.1 \le x_4 \le 2 \end{aligned} \tag{23}$$

This optimization problem is solved by using different evolutionary algorithms such as GA with self adaptive penalty (Coello, 2000), Evolution Strategy (Mezura-Montes & Coello, 2008), PSO (Hu, Eberhart, & Shi, 2003), Gravitational search algorithm (GSA) (Rashedi, Nezamabadi-pour, & Saryazdi, 2009), Simulated Annealing (SA) (Hedar & Fukushima, 2006), Co-evolutionary Particle Swarm Optimization (PSO) (He & Wang, 2007), Differential Evolution (DE) (Mezura-Montes, Coello, Reyes, & Davila, 2007), HarmonySearch (HS) (Lee & Geem, 2005), Ant Colony Optimization (ACO) (Kaveh & Talatahari, 2010), Simple constrained PSO (L.C. Cagnina, 2008), Improved harmony search algorithm (IHS) (Mahdavi, Fesanghary, & Damangir, 2007), Cuckoo Search (CS) (Gandomi, Yang, & Alavi, 2013), Artificial Bee Colony Algorithm(ABC) (Karaboga & Basturk, 2008), Simplex Search Method (Mehta & Dasgupta, 2012) Whale optimization algorithm (WO) (Mirjalili, 2016), Ray optimization algorithm (RO) (Kaveh & Khayatazad, 2012). The maximum iteration and search agents in MHDA is set to 1500 and 50 respectively. The results of welded beam design problem using different algorithms is shown in Table 6. From the results it is clear that the proposed MHDA produced lowest cost of 1.6952471 and outperformed all other algorithms. The Statistical results after 30 independent runs in terms of best score, worst score, mean and standard deviation for different algorithms is shown in Table 7, out of which Standard Deviation of MHDA was the lowest(about 5.83118E-16). This proves the reliability of MHDA in solving this optimization problem.

### *5.2. Pressure vessel design*

The pressure vessel design is one of the widely used structural design benchmark problem. The objective of this mixed integer optimization problem is to minimize the total cost of materials, forming and welding. The thickness of the shell ($T_s$), the thickness of

**Table 6**
Comparison of optimization results for welded beam design problem by different algorithms.

| Algorithm | Optimum variables | | | | Optimum Cost |
|---|---|---|---|---|---|
| | h | l | t | b | |
| **MHDA** | **0.2057296** | **3.2531200** | **9.0366239** | **0.2057296** | **1.6952471** |
| DA | 0.194288 | 3.46681 | 9.04543 | 0.205695 | 1.70808 |
| GA with Selfadaptive penalty approach | 0.208800 | 3.420500 | 8.997500 | 0.21000 | 1.748309 |
| Evolution Strategy | 0.199742 | 3.612060 | 9.037500 | 0.20682 | 1.73730 |
| SA | 0.20564426 | 3.472578742 | 9.03662391 | 0.2057296 | 1.7250022 |
| Co-evolutionary PSO | 0.20573 | 3.47049 | 9.03662 | 0.20573 | 1.72485084 |
| GSA | 0.18219 | 3.856979 | 10.0000 | 0.202376 | 1.879952 |
| RO | 0.203687 | 3.528467 | 9.0024233 | 0.207241 | 1.735344 |
| WOA | 0.205396 | 3.484293 | 9.037426 | 0.206276 | 1.730499 |
| Simple constrained PSO | 0.205729 | 3.470488 | 9.036624 | 0.205729 | 1.724852 |
| PSO | 0.20573 | 3.47049 | 9.03662 | 0.20573 | 1.72485084 |
| Improved HS | 0.20573 | 3.47049 | 9.03662 | 0.20573 | 1.7248 |
| DE | 0.20573 | 3.470489 | 9.0336624 | 0.205730 | 1.724852 |
| Cuckoo Search | 0.2015 | 3.562 | 9.0414 | 0.2057 | 1.73121 |
| ABC | 0.205730 | 3.470489 | 9.036624 | 0.205730 | 1.724852 |
| ACO | 0.205700 | 3.471131 | 9.036683 | 0.205731 | 1.724918 |
| Simplex Search Method | 0.20572885 | 3.47050567 | 9.03662392 | 0.20572964 | 1.724855 |

**Table 7**
Statistical results of different optimization algorithms for solving welded beam design problem.

| Algorithm | Best Score | Worst Score | Mean | Std. Dev |
|---|---|---|---|---|
| **MHDA** | **1.6952471** | **1.6952471** | **1.6952471** | **5.83118E−16** |
| DA | 1.70808 | 2.52106 | 1.94076 | 0.250234 |
| GA with Self adaptive penalty approach | 1.748309 | 1.771973 | 1.785835 | 0.011220 |
| Evolution Strategy | 1.728226 | 1.993408 | 1.792654 | 0.07471 |
| SA | 1.7250022 | 1.8843960 | 1.7564428 | NA |
| Co-evolutionary PSO | 1.728024 | 1.782143 | 1.748831 | 0.012926 |
| Cuckoo Search | 1.7312065 | 2.3455793 | 1.8786560 | 0.2677989 |
| Simple constrained PSO | 1.724852 | NA | 2.0574 | 0.2154 |
| DE | 1.724852 | NA | 1.725 | 1E-15 |
| Cuckoo Search | 1.7312065 | 2.3455793 | 1.8786560 | 0.2677989 |
| ABC | 1.724852 | NA | 1.741913 | 0.031 |
| ACO | 1.72918 | 1.775961 | 1.729752 | 0.009200 |

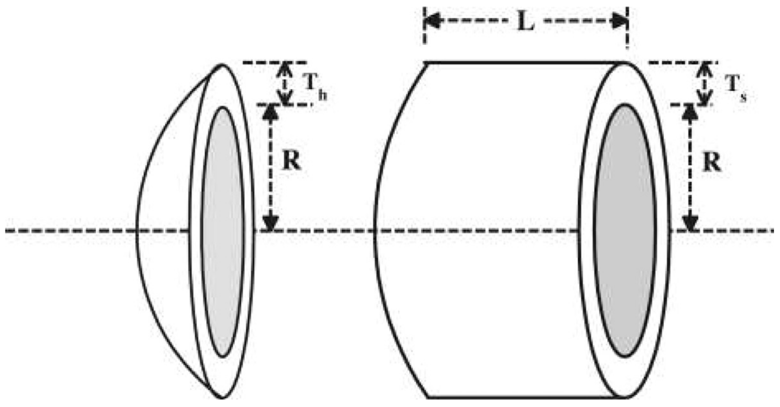

**Fig. 5.** Pressure vessel design problem.

the head ($T_h$), the inner radius ($R$), the length of cylindrical section without considering the head ($L$) are the design variables involved in the optimization. Fig. 5 showing the cross-section of pressure vessel is taken from the reference (Kannan & Kramer, 1994). The variable vector (in inches) can be written as $\vec{X} = [\vec{x_1}, \vec{x_2}, \vec{x_3}, \vec{x_4}]$ where $\vec{x_1}$, $\vec{x_2}$, $\vec{x_3}$ and $\vec{x_4}$ represents $T_s$, $T_h$, $R$ and $L$.

The optimization problem can be written as

Minimize

$$f(\vec{X}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \tag{24}$$

subject to

$$g_1(\vec{X}) = -x_1 + 0.0193x_3 \le 0 \tag{25}$$

$$g_2(\overrightarrow{X}) = -x_3 + 0.00954x_3 \tag{26}$$

$$g_3(\overrightarrow{X}) = -\Pi x_3^2x_4 - \frac{4}{3}\Pi x_3^3 + 1296000 \le 0 \tag{27}$$

$$g_4(\overrightarrow{X}) = x_4 - 240 \le 0 \tag{28}$$

$$\begin{aligned} 0 &\le x_1 \le 99 \\ 0 &\le x_2 \le 99 \\ 10 &\le x_3 \le 200 \\ 10 &\le x_4 \le 200 \end{aligned} \tag{29}$$

**Table 8**
Comparison of optimization results for pressure vessel design problem by different algorithms.

| Algorithm | Optimum variables | | | | Optimum Cost |
|---|---|---|---|---|---|
| | $T_s$ | $T_h$ | $R$ | $L$ | |
| **MHDA** | **0.778169** | **0.384649** | **40.3196** | **200** | **5885.3353** |
| DA | 0.782825 | 0.384649 | 40.3196 | 200 | 5923.11 |
| Non-linear and discrete programming | 1.125000 | 0.625000 | 47.700000 | 117.70100 | 8129.1036 |
| SA | 1.125000 | 0.625000 | 58.290000 | 43.6930000 | 7197.7000 |
| HS | 1.125000 | 0.625000 | 58.278900 | 43.75490000 | 7198.433 |
| Augmented Lagrange multiplier | 1.125000 | 0.625000 | 58.291000 | 43.690000 | 7198.0428 |
| GeneAS | 0.937500 | 0.500000 | 48.329000 | 112.67900 | 6410.3811 |
| GA with Self adaptive penalty approach | 0.812500 | 0.437500 | 40.323900 | 200.0000 | 6288.7445 |
| Co-evolutionary PSO | 0.812500 | 0.437500 | 42.091266 | 176.746500 | 6061.077 |
| GA with dominance based tournament selection | 0.812500 | 0.437500 | 42.097398 | 176.654050 | 6059.946 |
| Evolution Strategy | 0.812500 | 0.437500 | 42.098087 | 176.640518 | 6059.7456 |
| Guassian QPSO | 0.812500 | 0.437500 | 42.098400 | 176.637200 | 6059.7208 |
| Cuckoo Search | 0.812500 | 0.437500 | 42.0984456 | 176.6363595 | 6059.7143348 |
| Simple Constrained PSO | 0.812500 | 0.437500 | 42.0984456 | 176.6363595 | 6059.714335 |
| ABC | 0.812500 | 0.437500 | 42.098446 | 176.636596 | 6059.714339 |
| Improved PSO | 0.812500 | 0.437500 | 42.098445 | 176.6365950 | 6059.7143 |
| DE | 0.812500 | 0.437500 | 42.098446 | 176.6360470 | 6059.701660 |
| PSO | 0.812500 | 0.437500 | 42.098450 | 176.636600 | 6059.131296 |
| Hybrid PSO | 0.812500 | 0.437500 | 42.103566 | 176.573220 | 6059.0925 |
| WO | 0.782825 | 0.384847 | 40.3403 | 200 | 5923.11 |
| Penalty guided ABC | 0.7781686 | 0.3846491 | 40.3210545 | 199.9802367 | 5885.40322828 |
| Hybrid PSO-GA | 0.7781686 | 0.3846491 | 40.3196187 | 200 | 5885.3327736 |

**Table 9**
Statistical results of different optimization algorithms for solving pressure vessel design.

| Algorithm | Best Score | Worst Score | Mean | Std. Dev |
|---|---|---|---|---|
| **MHDA** | **5885.3353** | **5885.3353** | **5885.3353** | **0** |
| DA | 5923.11 | 222536 | 21342.2 | 47044.2 |
| Non-linear and discrete programming | 8129.1036 | NA | NA | NA |
| Augmented Lagrange based method | 7198.0428 | NA | NA | NA |
| GeneAS | 6410.3811 | NA | NA | NA |
| GA with self adaptive penalty approach | 6288.7445 | 6308.1497 | 6293.8432 | NA |
| GA with dominance based tournament selection | 6059.9463 | 6469.3220 | 6177.2533 | 130.9297 |
| Co-evolutionary PSO | 6061.077 | 6363.8041 | 6147.1332 | 86.4545 |
| Evolution Strategy | 6059.7456 | 7332.8798 | 6850.0049 | 426.000 |
| Cuckoo Search | 6059.714 | 6495.3470 | 6447.7360 | 502.693 |
| Improved PSO | 6059.7143 | NA | 6289.92881 | 305.78 |
| ABC | 6059.714339 | NA | 6245.308144 | 205 |
| Penalty guided ABC | 5885.403282 | 5895.126804 | 5887.557024 | 2.745290 |

This optimization problem is solved by many researchers using different algorithms like non linear and discrete programming (Sandgren, 1990), Simulated Annealing (Zhang & Wang, 1993), Harmony Search (HS) (Lee & Geem, 2005), Augmented Lagrange multiplier (Kannan & Kramer, 1994), GeneAS (Deb, 1997), GA with self adaptive penalty approach (Coello, 2000), Guassian QPSO (dos & Coelho, 2010), Co-evolutionary PSO (He & Wang, 2007), GA with dominance based tournament selection (Coello & Montes, 2002), Evolution Strategy (Mezura-Montes & Coello, 2008), GA with self-adaptive penalty approach (Coello, 2000), Cuckoo search algorithm (Gandomi et al., 2013), Artificial Bee Colony algorithm (ABC) (Akay & Karaboga, 2012), Simple constrained PSO (L.C. Cagnina, 2008), Improved PSO (He, Prempain, & Wu, 2004), Differential Evolution(DE), Improved Ant Colony Optimization(ACO) (Kaveh & Talatahari, 2010), Hybrid PSO (Kaveh & Talatahari, 2009), PSO (Hu et al., 2003), Whale Optimization (WO) (Mirjalili & Lewis, 2016), Penalty guided ABC (Garg, 2014). The best solutions of pressure vessel design problem found by different algorithms is shown in Table 8. The maximum iteration and population size is set to1500 and 50 respectively. The statistical results of different algorithms after 30 independent runs are shown in Table 9. From the results it is clear that the result obtained by the performance of the proposed optimization algorithm was better than other algorithms in the literature. However considering the statistical results the performance of MHDA is superior. The standard deviation of the results after 30 independent runs is zero which indicates that the proposed hybrid optimization algorithm is very effective and reliable in solving this optimization problem.

## 5.3. Brushless DC motor optimization benchmark

Brushless DC Motor (BLDC) wheel problem is well known optimization problem in electromagnetism. The objective functions and constraints for design optimization of typical brushless DC motor are available online (Benchmark[Online], 2005). This problem aims to maximize the efficiency of the motor,$\eta$ with five optimization parameters: bore stator diameter ($D_s$), flux density in the air-gap ($B_d$), current density in the conductors ($\delta$), teeth flux density ($B_e$) and back iron flux density ($B_{cs}$). The total mass ($M_{tot}$), inner diameter ($D_{int}$), external diameter ($D_{ext}$), Maximum current ($I_{\max}$), and temperature ($T_a$) and determinant used in the slot height calculation ($Discr$) which depend upon design variables are the constraints in this problem. The decision variable is represented as $X = (D_s,\ B_e,\ \delta,\ B_d, B_{cs}) = (x_1,x_2,\ x_3,\ x_4)$. The problem can be expressed mathematically as follows.

**Table 10**
Comparison of optimization results for BLDC optimization benchmark problem by different algorithms.

| Algorithm | $x_1(mm)$ | $x_2(T)$ | $x_3(A/mm^2)$ | $x_4(T)$ | $x_5(T)$ | $\eta(\%)$ |
|---|---|---|---|---|---|---|
| **MHDA** | **201.5** | **0.6479** | **2.0000** | **1.8** | **0.89496** | **95.32** |
| DA | 201.2 | 0.6481 | 2.0438 | 1.8 | 0.896413 | 95.31 |
| PSO | 202.1 | 0.6476 | 2.0417 | 1.8 | 0.9298 | 95.32 |
| ACO | 201.2 | 0.6481 | 2.0437 | 1.8 | 0.8959 | 95.32 |
| GA | 201.5 | 0.6480 | 2.0602 | 1.799 | 0.8817 | 95.31 |
| GA&SQP | 201.2 | 0.6481 | 2.0615 | 1.8 | 0.8700 | 95.31 |
| BA | 202.2 | 0.6535 | 2.0514 | 1.8 | 0.9792 | 95.31 |
| MSSO | 201.2 | 0.6481 | 2.0437 | 1.8 | 0.8959 | 95.32 |

**Table 11**
Statistical results of different optimization algorithms for solving BLDC optimization benchmark problem.

| Algorithm | Best Score | Worst Score | Mean | Std. Dev |
|---|---|---|---|---|
| **MHDA** | **95.32** | **95.32** | **95.32** | **2.18762E−07** |
| DA | 95.31 | 95.06 | 95.18 | 8.5713E−04 |
| BAT | NA | NA | 95.23 | 0.056 |
| SSO | 94.98 | 94.81 | 94.88 | 0.08 |

Minimize

$$f(x) = 1 - \eta \tag{30}$$

subject to

$$M_{tot} \leq 15 \text{ kg} \tag{31}$$

$$D_{ext} \leq 0.340 \text{ m} \tag{32}$$

$$D_{int} \geq 0.076 \text{ m} \tag{33}$$

$$I_{max} \geq 125 \text{ A} \tag{34}$$

$$T_a < 125 \text{ A} \tag{35}$$

$$discr(D_s, \delta, B_d, B_e) \geq 0 \tag{36}$$

The optimization results using different evolutionary algorithms are shown in the Table 10. It is observed that MHDA gave an efficiency of 95.32% which is probably the global optimal solution of the problem. Ant Colony Optimization (ACO) (Benchmark[Online], 2005), Sequential Quadratic Programming (SQP) algorithm (Benchmark[Online], 2005), PSO (Benchmark[Online], 2005) and Modified Social Spider Optimization algorithm (MSSO) (Klein, Segundo, Mariani, & dos S. Coelho, 2016) was equally efficient in producing the same efficiency. The statistical results of different algorithms for this problem is shown in the Table 11. The average value and standard deviation for 30 independent runs and with maximum iteration of 1000 was 0.0469729 and 0.00045512 respectively. The standard deviation of MHDA was much lower than other algorithms which gave the same efficiency. This highlights the reliable nature of MHDA in optimizing this problem.

## 6. Conclusion

This paper proposes a new efficient "Memory based Hybrid Dragonfly Algorithm (MHDA)" for numerical optimization problems. MHDA is a hybrid optimization algorithm based on classical DA and PSO. It uses the exploration capability of DA to explore the search space effectively and social and cognitive behaviour of PSO for faster convergence and to find the global best solution. In order to validate the effectiveness of this algorithm, the experiments were carried out using two test suites which comprises of basic unconstrained benchmark functions and CEC 2014 benchmark functions. The superior performance of MHDA in optimizing most of the unimodal functions in Suite-I and Suite-II proves its exploitation capability and convergence behaviour. The main contributing element in exploitation capability of MHDA is its iteration level hybridization with PSO which works on the best saved positions obtained so far in the search space. The performance of multimodal functions reveals the exploration capabilities provided by Levy flight search process and random initialization of DA. MHDA also produces competitive results in composite functions which shows the balance between global and local search process. The optimization results of MHDA is also compared with other well known algorithms in the literature in terms of mean and standard deviation. The ranking of different algorithms in solving the benchmark problems is done by Friedman's rank test and post hoc analysis is carried out by Wilcoxon ranksum test. The MHDA algorithm proved to be a powerful candidate surpassing all the other algorithms. The convergence behaviour of MHDA was analyzed and found that MHDA converge very quickly compared to other algorithms. The potential of MHDA for solving real time optimization problems is demonstrated by solving three well known engineering design problems. MHDA algorithm proved to be very effective for locating global optimal solutions or near global optimal solution in all three cases. The simulation results are also carried out in the terms of best score, worst score, mean and standard deviation. In majority of the cases worst solution given by MHDA is better than any of the solutions provided by other algorithms. This supports the application of MHDA as a stable and reliable optimization algorithm in expert systems.

MHDA has very few parameters to fine tune as in DA and self-adaptive nature of these factors makes it easy to use for optimization problems. Being a population based algorithm, MHDA has high capability to avoid local optima like other algorithms. In addition, the integration of internal memory concept borrowed from PSO helps in improving the quality of the solution in updating process and avoids stagnation at local optima. Moreover MHDA does not require any gradient information, so it can be easily applied on real world problems. The future work involves verifying the credibility of the proposed algorithm in solving discrete and multi-objective optimization problems. Other types of hybridization to improve the solution quality and convergence speed of conventional DA can be also considered.

## Acknowledgement

We express our sincere gratitude to Dr. Jayalal M L, Computing Systems Section, Indira Gandhi Centre for Atomic Research (IGCAR), Kalpakkam for his helpful and constructive suggestions. We also thank our colleagues in Remote Handling Inspection Devices Section (RHIDS), Director, MMG and Director IGCAR for their constant support and encouragement towards this work.

## Appendix

In this section we give the details of composite test functions ($F_{14} - F_{19}$) used in Suite-I. The basic functions for each composite function and controlling factors $\sigma$, $\lambda$ for different functions are given below.

- $F_{14}$

$$f_1, f_2 \ldots f_{10} = Sphere\ Function$$

$$[\sigma_{1,}\sigma_2 \ldots \sigma_{10}] = [1, 1, 1, \ldots 1]$$

$$[\lambda_{1,}\lambda_2, \ldots \lambda_{10}] = \left[\frac{5}{100}, \frac{5}{100} \cdots \frac{5}{100}\right]$$

- $F_{15}$

$$f_1, f_2 \ldots f_{10} = Griewank's\ Function$$

$$[\sigma_{1,}\sigma_2 \ldots \sigma_{10}] = [1, 1, 1, \ldots 1]$$

$$[\lambda_{1,}\lambda_2, \ldots \lambda_{10}] = \left[\frac{5}{100}, \frac{5}{100} \cdots \frac{5}{100}\right]$$

- $F_{16}$

$$f1, f2 \ldots f10 = Griewank's\ Function$$

$$[\sigma_{1,}\sigma_2 \ldots \sigma_{10}] = [1, 1, 1, \ldots 1]$$

$$[\lambda_{1,}\lambda_2, \ldots \lambda_{10}] = [1, 1, 1, \ldots 1]$$

- $F_{17}$

$$f_1, f_2 = Ackley's\ Function,$$
$$f_3, f_4 = Rastrigin's\ Function\ f_5, f_6 = Weierstrass\ Function,$$
$$f_7, f_8 = Griewank's\ Function\ f_9, f_{10} = Sphere\ Function$$

$$[\sigma_{1,}\sigma_2 \ldots \sigma_{10}] = [1, 1, 1, \ldots 1]$$

$$[\lambda_{1,}\lambda_2, \ldots \lambda_{10}] = \left[\frac{5}{32}, \frac{5}{32}, 1, 1, \frac{5}{0.5}, \frac{5}{0.5}, \frac{5}{100}, \frac{5}{100}, \frac{5}{32}, \frac{5}{32}, \frac{5}{100}, \frac{5}{100}\right]$$

- $F_{18}$

$$f_1, f_2 = Rastrigin's\ Function,$$
$$f_3, f_4 = Weierstrass\ Function\ f_5, f_6 = Griewank's\ Function$$
$$f_7, f_8 = Ackley's\ Function\ f_9, f_{10} = Sphere\ Function$$

$$[\sigma_{1,}\sigma_2 \ldots \sigma_{10}] = [1, 1, 1, \ldots 1]$$

$$[\lambda_{1,}\lambda_2, \ldots \lambda_{10}] = \left[\frac{1}{5}, \frac{1}{5}, \frac{5}{0.5}, \frac{5}{0.5}, \frac{5}{100}, \frac{5}{100}, \frac{5}{32}, \frac{5}{32}, \frac{5}{100}, \frac{5}{100}\right]$$

- $F_{19}$

$$f_1, f_2 = Rastrigin's\ Function,$$
$$f_3, f_4 = Weierstrass\ Function\ f_5, f_6 = Griewank's\ Function$$
$$f_7, f_8 = Ackley's\ Function\ f_9, f_{10} = Sphere\ Function$$

$$[\sigma_{1,}\sigma_2 \ldots \sigma_{10}] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$$

$$[\lambda_{1,}\lambda_2, \ldots \lambda_{10}] = \Big[0.1 * \frac{1}{5}, 0.2 * \frac{1}{5}, 0.3 * \frac{5}{0.5}, 0.4 * \frac{5}{0.5}, 0.5 * \frac{5}{100}, 0.6 * \frac{5}{100}, 0.7 * \frac{5}{32}, 0.8 * \frac{5}{32} 0.9 * \frac{5}{100}, 1 * \frac{5}{100}\Big]$$

The mathematical formulation of basic functions used in the creation of composite functions are given below.

- Sphere Function

  $f(x) = \sum_{i=1}^{d} x_i^2, x\in[-100, 100]^d$

- Rastrigin's Function

  $f(x) = n * 10 + \sum_{i=1}^{d} (x_i - 10cos(2\Pi x_i))x\in[-5, 5]^d$

- Griewank's Function

  $f(x) = \frac{1}{4000} \sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} cos(\frac{x}{\sqrt{1}}) + 1, x\in[-100, 100]^d$

- Ackley's Function

  $f(x) = -20exp[-\frac{1}{5}\sqrt{\frac{1}{d}\sum_{i=1}^{d} cos(2\Pi x_i)}] + 20 + e, x\in[-32, 32]^d$

- Weierstrass Function

  $f(x) = \sum_{i=1}^{d} \sum_{k=0}^{k=k_{\max}} \left[a^k cos(2\Pi b^k(x_i + 0.5))\right] - D\sum_{k=0}^{k=k_{\max}} \left[a^k cos(2\Pi b^k 0.5)\right]$

## References

Akay, B., & Karaboga, D. (2012). Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing, 23*(4), 1001–1014.

AlRashidi, M. R., & El-Hawary, M. E. (2009). A survey of particle swarm optimization applications in electric power systems. *IEEE Transactions on Evolutionary Computation, 13*(4), 913–918.

Benchmark[Online ] (2005). A benchmark for a mono and multi objective optimization of the brushless dc wheel motor. Accessed 11-November-2016(http://l2ep.univ-lille1.fr/come/benchmark-wheel-motor/OptRest.htm).

Blum, C., & Li, X. (2008). Swarm intelligence in optimization. In C. Blum, & D. Merkle (Eds.), *Swarm intelligence: Introduction and applications* (pp. 43–85)). Berlin, Heidelberg: Springer Berlin Heidelberg.

Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation, 10*(6), 646–657.

Chen, D., Zou, F., Lu, R., & Wang, P. (2017). Learning backtracking search optimisation algorithm and its application. *Information Sciences, 376*, 71–94.

Civicioglu, P. (2013). Backtracking search optimization algorithm for numerical optimization problems. *Applied Mathematics and Computation, 219*(15), 8121–8144.

Coello, C. A. C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry, 41*(2), 113–127.

Coello, C. A. C., & Montes, E. M. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics, 16*(3), 193–203.

Cagnina, L. C., Coello, C. C. L., & Esquivel, S. C. (2008). Solving engineering optimization problems with the simple constrained particle swarm optimizer. *Informatica, 32*, 319–326.

Daely, P. T., & Shin, S. Y. (2016). Range based wireless node localization using dragonfly algorithm. In *2016 eighth international conference on ubiquitous and future networks (ICUFN)* (pp. 1012–1015).

Deb, K. (1997). Geneas: A robust optimal design technique for mechanical component design. In D. Dasgupta, & Z. Michalewicz (Eds.), *Evolutionary algorithms in engineering applications* (pp. 497–514)). Berlin, Heidelberg: Springer Berlin Heidelberg.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

Derrac, J., Garcia, S., Molina, D., & Herrera, F. (2011). 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), 3–18.

Dorigo, M., & Thomas, S. (2004). *Ant colony optimization*. MIT Press eBooks.

dos, L., & Coelho, S. (2010). Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. *Expert Systems with Applications, 37*(2), 1676–1683.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Micro machine and human science, 1995. MHS '95., proceedings of the sixth international symposium on* (pp. 39–43).

Elsayed, S. M., Sarker, R. A., & Essam, D. L. (2013). An improved self-adaptive differential evolution algorithm for optimization problems. *IEEE Transactions on Industrial Informatics, 9*(1), 89–99.

Erlich, I., Rueda, J. L., Wildenhues, S., & Shewarega, F. (2014). Evaluating the mean–variance mapping optimization on the ieee-cec 2014 test suite. In *2014 IEEE congress on evolutionary computation (CEC)* (pp. 1625–1632).

Gandomi, A. H., & Alavi, A. H. (2012). Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation, 17*(12), 4831–4845.

Gandomi, A. H., Yang, X.-S., & Alavi, A. H. (2013). Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Engineering with Computers, 29*(1), 17–35.

Garg, H. (2014). Solving structural engineering design optimization problems using an artificial bee colony algorithm. *Journal of Industrial and Management Optimization, 10*(3), 777–794.

Garg, H. (2016). A hybrid pso-ga algorithm for constrained optimization problems. *Applied Mathematics and Computation, 274*, 292–305.

Ghorbani, A., & Jokar, M. R. A. (2016). A hybrid imperialist competitive-simulated annealing algorithm for a multisource multi-product location-routing-inventory problem. *Computers & Industrial Engineering, 101*, 116–127.

Greene, C. S., White, B. C., & Moore, J. H. (2008). Ant colony optimization for genome-wide genetic analysis. In M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, & A. F. T. Winfield (Eds.), *Ant colony optimization and swarm intelligence: 6th international conference, ANTS 2008, Brussels, Belgium, September 22–24, 2008. Proceedings* (pp. 37–47)). Berlin, Heidelberg: Springer Berlin Heidelberg.

He, Q., & Wang, L. (2007). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence, 20*(1), 89–99.

He, S., Prempain, E., & Wu, Q. H. (2004). An improved particle swarm optimizer for mechanical design optimization problems. *Engineering Optimization, 36*(5), 585–605.

Hedar, A.-R., & Fukushima, M. (2006). Derivative-free filter simulated annealing method for constrained continuous global optimization. *Journal of Global Optimization, 35*(4), 521–549.

Hema, C., Sankar, S., & Sandhya (2016). Energy efficient cluster based protocol to extend the rfid network lifetime using dragonfly algorithm. In *2016 international conference on communication and signal processing (ICCSP)* (pp. 0530–0534).

Hu, X., Eberhart, R. C., & Shi, Y. (2003). Engineering optimization with particle swarm. In *Swarm intelligence symposium, 2003. SIS '03. Proceedings of the 2003 IEEE* (pp. 53–57).

Kannan, & Kramer (1994). An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *ASME Journal of Mechanical Design, 116(2*, 405–411.

Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (abc) algorithm. *Applied Soft Computing, 8*(1), 687–697.

Kashani, A. R., Gandomi, A. H., & Mousavi, M. (2016). Imperialistic competitive algorithm: A metaheuristic algorithm for locating the critical slip surface in 2-dimensional soil slopes. *Geoscience Frontiers, 7*(1), 83–89. Special Issue: Progress of Machine Learning in Geosciences.

Kaveh, A., & Khayatazad, M. (2012). A new meta-heuristic method: Ray optimization. *Computers & Structures, 112–113*, 283–294.

Kaveh, A., & Talatahari, S. (2010). An improved ant colony optimization for constrained engineering design problems. *Engineering Computations, 27*(1), 155–182.

Kaveh, A., & Talatahari, S. T. (2009). Engineering optimization with hybrid particle swarm anoptimization optimization. *Asian Journal of Civil Engineering, 10*, 267–283.

Klein, C. E., Segundo, E. H. V., Mariani, V. C., & dos S. Coelho, L. (2016). Modified social-spider optimization algorithm applied to electromagnetic optimization. *IEEE Transactions on Magnetics, 52*(3), 1–4.

Kuo, R., & Huang, C. (2009). Application of particle swarm optimization algorithm for solving bi-level linear programming problem. *Computers & Mathematics with Applications, 58*(4), 678–685.

Liang, J., Suganthan, P., & Qu, B. (2014). Problem definitions and evaluation criteria for the CEC2014 special session and competition on single objective real parameter numerical optimization. *Technical Report*. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China Technical Report, Nanyang Technological University, Singapore.

Lee, K. S., & Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering, 194*(36–38), 3902–3933.

Li, Z., Wang, W., Yan, Y., & Li, Z. (2015). Ps-abc: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems. *Expert Systems with Applications, 42*(22), 8881–8895.

Liang, J. J., Suganthan, P. N., & Deb, K. (2005). Novel composition test functions for numerical global optimization. In *Proceedings 2005 IEEE swarm intelligence symposium, 2005. SIS 2005* (pp. 68–75). doi:10.1109/SIS.2005.1501604.

Luh, G.-C., & Lin, C.-Y. (2009). Structural topology optimization using ant colony optimization algorithm. *Applied Soft Computing, 9*(4), 1343–1353.

Ma, H., Simon, D., Fei, M., Shu, X., & Chen, Z. (2014). Hybrid biogeography-based evolutionary algorithms. *Engineering Applications of Artificial Intelligence, 30*, 213–224.

Mahdavi, M., Fesanghary, M., & Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation, 188*(2), 1567–1579.

Mehta, V. K., & Dasgupta, B. (2012). A constrained optimization algorithm based on the simplex search method. *Engineering Optimization, 44*(5), 537–550.

Mezura-Montes, & Coello, C. A. C. (2008). An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. *International Journal of General Systems, 37*(4), 443–473.

Mezura-Montes, E., Coello, C. A. C., Reyes, J., & Davila, L. (2007). Multiple trial vectors in differential evolution for engineering design. *Engineering Optimization, 39*(5), 567–589.

Mirjalili, S. (2015). The ant lion optimizer. *Advances in Engineering Software, 83*, 80–98.

Mirjalili, S. (2016). Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications, 27*(4), 1053–1073.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.

Mirjalili, S., Saremi, S., Mirjalili, S. M., & dos S. Coelho, L. (2016). Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. *Expert Systems with Applications, 47*, 106–119.

Mlakar, U., Jr. , I. F., & Fister, I. (2016). Hybrid self-adaptive cuckoo search for global optimization. *Swarm and Evolutionary Computation, 29*, 47–72.

Nabil, E. (2016). A modified flower pollination algorithm for global optimization. *Expert Systems with Applications, 57*, 192–203.

Nasir, A., Tokhi, M., & Ghani, N. (2015). Novel adaptive bacterial foraging algorithms for global optimisation with application to modelling of a trs. *Expert Systems with Applications, 42*(3), 1513–1530.

Nickabadi, A., Ebadzadeh, M. M., & Safabakhsh, R. (2011). A novel particle swarm optimization algorithm with adaptive inertia weight. *Applied Soft Computing, 11*(4), 3658–3670.

Omran, M. G. H., Engelbrecht, A. P., & Salman, A. (2006). Particle swarm optimization for pattern recognition and image processing. In A. Abraham, C. Grosan, & V. Ramos (Eds.), *Swarm intelligence in data mining* (pp. 125–151)). Berlin, Heidelberg: Springer Berlin Heidelberg.

Parouha, R. P., & Das, K. N. (2016). A memory based differential evolution algorithm for unconstrained optimization. *Applied Soft Computing, 38*, 501–517.

Pillo, G. D., & Grippo, L. (1989). Exact penalty functions in constrained optimization. *SIAM Journal on Control and Optimization, 27*(6), 1333–1360.

Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation, 13*(2), 398–417.

Raman, G., Raman, G., Manickam, C., & Ganesan, S. I. (2016). Dragonfly algorithm based global maximum power point tracker for photovoltaic systems. In Y. Tan, Y. Shi, & B. Niu (Eds.), *Advances in swarm intelligence: 7th international conference, ICSI 2016, Bali, Indonesia, June 25–30, 2016, proceedings, part I* (pp. 211–219)). Cham: Springer International Publishing.

Rao, S. (2009). *Engineering optimization: Theory and practice: Fourth edition*. John Wiley and Sons.

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). Gsa: A gravitational search algorithm. *Information Sciences, 179*(13), 2232–2248.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on computer graphics and interactive techniques*. In *SIGGRAPH '87* (pp. 25–34). New York, NY, USA: ACM.

Sambandam, R. K., & Jayaraman, S. (2016). Self-adaptive dragonfly based optimal thresholding for multilevel segmentation of digital images. *Journal of King Saud University - Computer and Information Sciences*. http://dx.doi.org/10.1016/j.jksuci.2016.11.002.

Sandgren (1990). Nonlinear integer and discrete programming in mechanical design optimization. *ASME Journal of Mechanical Design, 112(2)*, 223–229..

Sattari, M. R. J., Malakooti, H., Jalooli, A., & Noor, R. M. (2014). A dynamic vehicular traffic control using ant colony and traffic light optimization. In J. Świątek, A. Grzech, P. Świątek, & J. M. Tomczak (Eds.), *Advances in systems science: Proceedings of the international conference on systems science 2013 (ICSS 2013)* (pp. 57–66)). Cham: Springer International Publishing.

Sharma, A., Sharma, A., Panigrahi, B., Kiran, D., & Kumar, R. (2016). Ageist spider monkey optimization algorithm. *Swarm and Evolutionary Computation, 28*, 58–77.

Thangaraj, R., Pant, M., Abraham, A., & Bouvry, P. (2011). Particle swarm optimization: Hybridization perspectives and experimental illustrations. *Applied Mathematics and Computation, 217*(12), 5208–5226.

Wang, G.-G. (2016). Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Computing*, 1–14.

Wang, X., & Qiu, X. (2013). Application of particle swarm optimization for enhanced cyclic steam stimulation in a offshore heavy oil reservoir. *International Journal of Information Technology, Modeling and Computing (IJITMC), 1*(2), 37–47.

Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. In O. Watanabe, & T. Zeugmann (Eds.), *Stochastic algorithms: Foundations and applications: 5th international symposium, SAGA 2009, Sapporo, Japan, October 26–28, 2009. Proceedings* (pp. 169–178)). Berlin, Heidelberg: Springer Berlin Heidelberg.

Yang, X.-S. (2010a). Firefly algorithm, lévy flights and global optimization. In M. Bramer, R. Ellis, & M. Petridis (Eds.), *Research and development in intelligent systems XXVI: Incorporating applications and innovations in intelligent systems XVII* (pp. 209–218)). London: Springer London.

Yang, X.-S. (2010b). A new metaheuristic bat-inspired algorithm. In J. R. González, D. A. Pelta, C. Cruz, G. Terrazas, & N. Krasnogor (Eds.), *Nature inspired cooperative strategies for optimization (NICSO 2010)* (pp. 65–74)). Berlin, Heidelberg: Springer Berlin Heidelberg.

Yang, X.-S. (2014a). Chapter 1- introduction to algorithms. In X.-S. Yang (Ed.), *Nature-inspired optimization algorithms* (pp. 1–21). Oxford: Elsevier.

Yang, X.-S. (2014b). Chapter 2- analysis of algorithms. In X.-S. Yang (Ed.), *Nature-inspired optimization algorithms* (pp. 23–44). Oxford: Elsevier.

Zhang, C., & Wang, H.-P. (1993). Mixed-discrete non-linear optimization using simulated annealing. *Engineering Optimization, 21*(4), 277–291.

Zhang, J., & Sanderson, A. C. (2009). Jade: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation, 13*(5), 945–958.