Global Journal of Technology & Optimization

Rahman Tanveer et al., Global J Technol Optim 2016, 7:3
DOI: 10.4172/2229-8711.1000203

Research Article Open Access

A Comparative Study on Prominent Swarm Intelligence Methods for Function Optimization

Md. Siddiqur Rahman Tanveer, Md. Julfikar Islam and Akhand MAH*

Department of Computer Science and Engineering, Khulna University of Engineering and Technology, Khulna-9203, Bangladesh

Abstract

Optimization includes finding best available values of some objective function given a defined domain. Function optimization (FO) is the well-studied continuous optimization task which aim is to find best suited parameter values to get optimal value of a function. A number of techniques have been investigated in last few decades to solve FO and recently Swarm Intelligence (SI) methods, imitating power of the collective behavior of insects or animals, become popular to solve it. A number of SI methods have been developed on different time and tested on different test functions; therefore, it is important to compare the algorithms on a common test bench to identify their capability as well as best suited method for FO. The objective of this study is to draw a fair comparison among prominent SI methods in solving benchmark test functions. The SI methods considered in this study are Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC) optimization, Firefly Algorithm (FFA), Cuckoo Search Optimization (CSO), Group Search Optimization (GSO) and Grey Wolf Optimizer (GWO). Among the methods PSO is the pioneer and most popular in recent time; and GWO is the most recently developed method. The performance of the methods is compared in solving a suite of 22 well known benchmark test functions having different ranges, dimensions and types. Experimental results as well as analysis revealed that GWO is the overall best method among the SI methods and PSO is still promising to solve bench mark functions.

Keywords: Function optimization; Fitness function; Swarm intelligence; Particle swarm optimization; Artificial bee colony; Firefly algorithm; Cuckoo search optimization; Group search optimization; Grey wolf optimizer

Introduction

In general, optimization is the selection of a best element with regard to some criterion from some set of available alternatives. Optimization deals with the problem of finding numerically minimums (or maximums or zeros) of some objective functions in a defined domain [1]. In mathematics and computer science, optimization problems can be divided depending on the variables whether they are discrete or continuous [2]. Discrete optimization is also known as combinatorial optimization and its goal is to find best combination among given variables from a finite set. On the other hand, continuous optimization searches best suited parameter values in a given boundary.

Function optimization (FO) is the well-studied continuous optimization task which aim is to find best suited parameter values to get optimal value of a function. FO can refer to either minimization or maximization type: minimization type FO searches minimum function value whereas minimization type searches maximum value. Mathematically, a minimization task is defined as:

Given $F : R^n \rightarrow R$

Find $\hat{x} \in R$ such that $F(\hat{x}) \leq F(x), \forall_x \in R^n$

And, a maximization task is defined as:

Given $F : R^n \rightarrow R$

Find $\hat{x} \in R$ such that $F(\hat{x}) \geq F(x), \forall_x \in R^n$

The domain R^n of F is referred to as the search space. Each element of R^n is called a candidate solution in the search space, with $\hat{x}$ being the optimal solution. The value n denotes the number of dimensions of the search space, and thus the number of parameters involved in the optimization problem. The function F is called the objective function which is used to map the search space to the function space. The function space is then mapped to the one dimensional fitness space, providing a single fitness value for each set of parameters. A fitness function quantifies the optimality of a solution so that particular solution may be ranked against all the other solutions. A fitness value is assigned to each solution depending on how close it to the target or optimal value.

A number of techniques have been investigated last few decades to solve FO and recently nature inspired Swarm Intelligence (SI) methods become popular to solve it. SI is a category of algorithms that imitate the collective behavior of animals, has drawn attraction to solve computational problems. SI algorithms have been gaining much popularity in recent years due to the fact that many real-world optimization problems have become increasingly large, complex and dynamic. Among different SI methods, Particle Swarm Optimization (PSO) [3-5] is well studied for FO which is developed mimicking behavior of bird flocking or fish schooling. Mimicking the behavior of animal or living things, some other popular SI methods are Firefly Algorithm (FFA) [3], Artificial Bee Colony (ABC) optimization [5], Cuckoo Search Optimization (CSO) [6], Group Search Optimization (GSO) [7] and Grey Wolf Optimizer (GWO) [8]. Among the methods, PSO is the pioneer one and GWO is the most recently developed method. Since the SI methods have been developed on different time and tested on different test functions, it is important to compare the algorithms on a common test bench to identify their capability.

***Corresponding author:** Akhand MAH, Department of Computer Science and Engineering, Khulna University of Engineering and Technology, Khulna-9203, Bangladesh, Tel: +8801926203027; E-mail: akhand@cse.kuet.ac.bd

Received October 10, 2016; **Accepted** November 21, 2016; **Published** November 25, 2016

Citation: Md. Rahman Tanveer S, Md. Islam J, Akhand MAH (2016) A Comparative Study on Prominent Swarm Intelligence Methods for Function Optimization. Global J Technol Optim 7: 203. doi: 10.4172/2229-8711.1000203

The objective of this study is to draw a fair comparison among prominent SI methods, in solving benchmark test functions. Algorithms selected in this study are PSO, ABC, FFA, CSO, GSO and GWO. A suite of 22 benchmark test functions are considered in this study. The significance of the selected functions is that they have different numbers of range, dimension and type. The outline of the paper is as follows: Section 2 briefly explains the SI methods considered in this study. Section 3 compares proficiency of the methods in solving benchmark functions, the section also includes in depth analysis on some selected functions. Finally, Section 4 concludes the paper with brief summary.

Nature Inspired Prominent SI Methods

Swarm intelligence (SI) is an artificial intelligence discipline that is concerned with the design of intelligent multi-agent systems by taking inspiration from the collective behavior of social insects such as ants, termites, bees, and wasps, as well as from other animal societies such as flocks of birds or schools of fish [9]. Colonies of social insects fascinated researchers many years ago and the mechanisms that govern their behavior remained unknown for a long time. Even though the single members of these colonies are non-sophisticated individuals, they are able to achieve complex tasks in cooperation [9].The term swarm intelligence was first used by Beni [10] in the context of cellular robotic systems where simple agents organize themselves through nearest-neighbor interaction [10]. Meanwhile, the term swarm intelligence is used for a much broader research field [11]. SI methods have been very successful in the area of optimization, which is of great importance for industry and science.

Optimization techniques inspired by SI have become increasingly popular during the last decade. They are characterized by a decentralized way of working that mimics the behavior of swarms of social insects, flocks of birds, or schools of fish [9]. The advantage of these approaches over traditional techniques is their robustness and flexibility. These properties make SI a successful design paradigm for algorithms that deal with increasingly complex problems. A large number of SI methods have been developed in recent years and brief description of the prominent methods is given below to make the paper self-contained.

Particle swarm optimization (PSO)

The PSO algorithm was proposed by Eberhart and Kennedy [12] on the swarming habits of bird flocking and fish schooling. The algorithm works by simultaneously maintaining several candidate solutions in the search space. During each iteration, each candidate solution is evaluated by the objective function being optimized, determining the fitness of that solution. Mimicking physical quantities such as velocity and position in bird flocking, artificial particles search optimum position in the search space. The number of particle is a user defined parameter and each one hold a feasible solution as a position in the search space [13].

Initially, particles are distributed uniformly in the search space, i.e., assign random solutions. In this regard two quantities are associated with each particle, a position vector $\vec{x}_i$ and a velocity $\vec{v}_i$. According to the following formula at each time step, the velocity of particles is updated.

$$\vec{v}_i^{t+1} = w * v_i^t + r_1 * c_1\left(\vec{P}_i - \vec{x}_i^{\,t}\right) + r_2 * c_2\left(\vec{P_g} - \vec{x}_i^{\,t}\right) \quad (1)$$

Where, $\vec{P}_g$ denotes the global best position (i.e., solution) and $\vec{P}_i$ represents the best encountered solution by the particle *i*. c_1 and c_2 are learning parameters; whereas r_1 and r_2 are random parameters in a range of [0,1]. With a view to avoiding the swarm being trapped into a local minimum inertia weight *w* is included. Position of each individual particle is updated adding calculated velocity for it.

$$\vec{x}_i^{t+1} = \vec{x}_i^t + \vec{v}_i^{t+1} \quad (2)$$

Artificial bee colony (ABC) optimization

The ABC algorithm was developed by Karaboga in 2005 mimicking the foraging behavior of honey bee [14]. There are three groups of bees in ABC: employed bees, onlooker bees and scout bees. A bee that goes to the food source visited by itself is called an employed bee. Carrying out random search by another bee called a scout. With a view to making decision to choose a food source an onlooker bee waits on the dance area. In the ABC algorithm, half of the colony is considered as employed artificial bees and the rest half constitutes the onlookers. For every food source, only one employed bee is set. In other words, the number of food sources around the hive is equal to the number of employed bees. An employed bee becomes scout when its food source is exhausted by the employed and onlooker bees.

In ABC algorithm, the position of a food source represents a possible outcome to the optimization problem and the amount of nectar of food source corresponds to the quality (i.e., fitness) of the associated solution. The size of employed bees or onlooker bees is denoted by *SN*. Every solution X_i *(i = 1, 2. . . SN)* is a D-dimensional vector. Solutions (food source positions) of initial population are generated randomly and then search processes performed by these three type of bees., An onlooker bee chooses a food source depending on the probability value associated with a food source p_i by the following expression:

$$p_i = \frac{f_i}{\sum_{i=1}^{SN} f_i} \quad (3)$$

Where f_i represents the fitness value of i^{th} bee which is proportional to the nectar amount of the food source in the position *i* and *SN* is the number of food sources which is equal to the number of corresponding employed or onlooker bees. With a view to producing a candidate food position from the old one in memory, the ABC uses the expression: $V_{ij} = X_{ij} + \Phi_{ij}\left(X_{ij} - X_{kj}\right)$. Where, *i* and *k* are chosen randomly and $j \in \{1, 2. . . D\}$ denotes the dimensions. Though *k* is determined randomly, it needs to be different from *i*. Φ_{ij} is a random number between [-1, 1]. It not only controls the production of neighbor food sources around X_{ij} but also represents the comparison of two food positions visually by a bee. In ABC, a food source is considered to be abandoned if the position which cannot be improved further through a predetermined number of cycles. This predetermined number of cycles is a major control parameter which is called "limit" for abandonment. Assume that the abandoned food source is X_i and $j \in \{1, 2. . . D\}$ after that scout discovers a new food source which is to be replaced with X_i.

$$X_i^j = X_{min}^j + rand[0,\ 1] * (X_{max}^j - X_{min}^j) \quad (4)$$

When each candidate source position v_{ij} is produced and after that it is evaluated by the artificial bee, its performance is compared with of its old one. If the new food source contains equal or better nectar than the old source, it is replaced immediately with the old one in the memory.

Firefly algorithm (FFA)

The FFA was developed in 2009 by Yang based on the flashing light

behavior of fireflies [15]. Fireflies produce short and rhythmic flashes and the pattern of flashes is often unique for a particular species. There are two fundamental functions of such flashes which help them to attract mating partners (i.e., communication), and to attract potential prey. The rhythmic flash, the rate of flashing and the amount of time help to bring both male and female fireflies together. In the same species females give the response to a male's unique pattern of flashing. There are three idealized rules in FFA: 1) all fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex; 2) attractiveness is proportional to their brightness, as a result the less brighter one will move towards the brighter one; and 3) the brightness of a firefly is affected or determined by the landscape of the objective function. In the FFA, there are also two important issues which are the variation of light intensity and formulation of the attractiveness.

In the simplest case, the brightness I of a firefly at a particular location x can be chosen as $I(x) / f(x)$. However, the attractiveness β is relative and it is seen by the eyes of the beholder or judged by the other fireflies. As a result, it will vary with the distance r_{ij} between firefly i and firefly j. For a given medium where the light absorption coefficient is fixed, the light intensity I varies with the distance r. That is $I = I_0 e^{-\gamma r}$, where I_0 represents the actual light intensity. A firefly's attractiveness β is proportional to the light intensity which is seen by the adjacent fireflies and it is defined by $\beta = \beta_0 e^{-\gamma r^2}$. The distance between any two fireflies i and j at x_i and x_j in Cartesian form

$$r_{ij} = x_i - x_j = \sqrt{\sum_{k=1}^{d} \left(x_{i,k} - x_{j,k}\right)^2} \tag{5}$$

where, $x_{i,k}$ is the k^{th} component of spatial coordinate x_i of i^{th} firefly. In 2-D case it is

$$r_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \tag{6}$$

The movement of a firefly i is attracted to another more attractive (brighter) firefly j which is determined by the following equation:

$$x_i = (1-\beta) * x_i + \beta * x_j + \alpha * \left(rand - \frac{1}{2}\right) \tag{7}$$

Here, the second term is due to the attraction and third term is randomization with α being the randomization parameter and *rand* is a random number generated uniformly in [0, 1]. For most cases it is taken β = 1 and $\alpha \in [0, 1]$. The parameter αdetermines the maximum radius of the random step. The parameter γ characterizes the variation of the attractiveness and its value is crucially important with a view to determining the speed of the convergence and how the FFA algorithm behaves.

Cuckoo search optimization (CSO)

CSO was invented by Yang and Deb in 2009 inspiring the brood parasitism behavior seen in some species of cuckoo [6]. Cuckoo laid their eggs in other bird's nest. When the host bird finds out this, it will either throw away the cuckoo's egg or simply abandon the whole nest and build a new nest. However, some species of cuckoo are very good at making their eggs as like the host's egg, as a result the survival probabilities of their eggs increase greatly. Naturally the cuckoo eggs hatch slightly earlier than their host eggs. When the first cuckoo chick is hatched, its first instinct action is to evict the host eggs by blindly propelling the eggs out of the nest. This action results in increasing the cuckoo chick's share of food provided by its host bird. Moreover, studies show that a cuckoo chick can imitate the call of host chicks to gain access to more feeding opportunity.

In CSO, firstly each cuckoo lays one egg at a time and dumps it in a randomly chosen nest. Secondly, the best nests with the best quality of eggs will be brought to the next generation. The number of host nests is fixed. The probability of a host bird will discover the egg is laid by cuckoo is defined by $p_\alpha \in (0,1)$. The host bird can get rid of the cuckoo egg or it can build a new nest. Any egg x_i which is in the nest represents a solution to the problem and at each time step the following equation is used to generate new solutions:

$$x_i^{\bar{t}+1} = x_i^{\bar{t}} + Randomwalk\left(Levy\ flight\right) * x_i^{\bar{t}} \tag{8}$$

The *Lévy flight* essentially provides a random walk while the random step length is drawn from a Levy distribution with a heavy tail [16-19]. In fact, *Lévy flight* have been observed among foraging patterns of albatrosses, fruit flies and spider monkeys. Figure 1 shows a sample *Lévy flight* pattern.

Group search optimizer (GSO)

GSO is developed He, et al. in 2009 based on animal searching behavior [7]. Animal searching behavior may be described as an active movement through which animal can find resources such as foods, mates, nesting sites. One major consequence of living together is its group searching strategy which allows group members to increase patch finding rates [16]. Simply this has led to the adoption of two foraging strategies: 1) producing, e.g., searching for food; and 2) scrounging, e.g., joining resources uncovered by others. The second one is also referred to as conspecific attraction, kleptoparasitism [17].

In GSO, population is called a *group* and each individual animal in the population is called a member. In an n dimensional search space, the i^{th} member at the kth searching bout has a current position which is $X_i \in \mathbb{R}^n$ and a head angle $\varphi_i^k = \left(\varphi_{i_1}^k, \ldots, \varphi_{i_{(n-1)}}^k\right) \in \mathbb{R}^{n-1}$. The search direction of the i^{th} member is a unit vector $D_i^k\left(\varphi_i^k\right) = \left(d_{i_1}^k, \ldots, d_{i_{(n)}}^k\right) \in \mathbb{R}^n$ which is measured from φ_i^k Polar to Cartesian coordinate transformation.

$$d_{i_1}^k = \prod_{q=1}^{n-1} \cos\left(\varphi_{i_q}^k\right) \tag{9}$$

$$d_{i_j}^k = \sin\left(\varphi_{i_{(j-1)}}^k\right) \cdot \prod_{q=j}^{n-1} \cos\left(\varphi_{i_q}^k\right) \quad (j = 2, \ldots, n-1) \tag{10}$$

$$d_{i_n}^k = \sin\left(\varphi_{i_{(n-1)}}^k\right) \tag{11}$$

Each iteration, a group member located in the most promising area (i.e., confers the best fitness value) chosen as the producer. It also scans the environment for better resources. A vision based scanning is considered in GSO and scanning characterized by maximum pursuit angle $\theta_{max} \in \mathbb{R}^1$, maximum pursuit distance $l_{max} \in \mathbb{R}^1$ is illustrated in a 3-D space in Figure 2. At the kth iteration producer X_p will scan at zero degree and then scan laterally by randomly sampling three points in the scanning field:

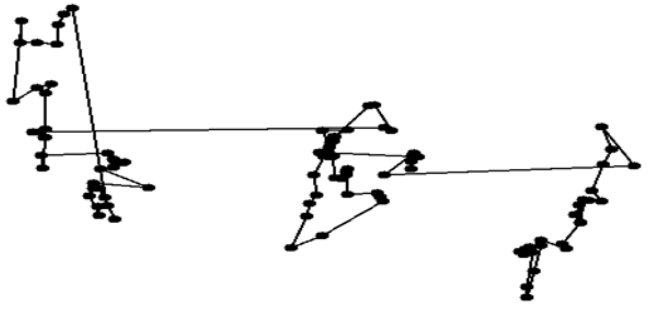


Figure 1: A sample Levy flight pattern.

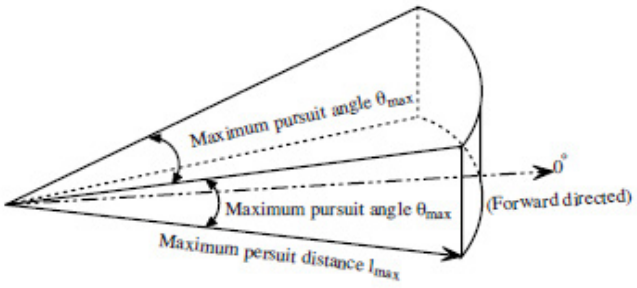


Figure 2: Scanning field in 3-D space [18].

One point at zero degree,

$$X_z = X_p^k + r_1 l_{max} D_p^k\left(\varphi^k\right) \quad (12)$$

One point in the right hand side hypercube,

$$X_r = X_p^k + r_1 l_{max} D_p^k\left(\varphi^k + r_2\theta_{max}/2\right) \quad (13)$$

One point in the left hand side hypercube

$$X_r = X_p^k + r_1 l_{max} D_p^k\left(\varphi^k - r_2\theta_{max}/2\right) \quad (14)$$

Here $r_1 \in \mathbb{R}^1$ is a random number with mean 0 and standard deviation 1 and $r_2 \in \mathbb{R}^{n-1}$ is random number sequence in the range (0, 1). These two numbers are normally distributed. Now the producer will find the best point according to the best resource (i.e., the best fitness value). If the resource of the best point is better than its current position, then it will fly to that point. Otherwise it will have to stay in its present position. It will turn its head to a new randomly generated angle that is $\varphi^{k+1} = \varphi^k + r_2 * a_{max}$ here, $a_{max} \in \mathbb{R}^1$ is considered the maximum turning angle. If the producer fails to find a better area after a iterations, it will have to turn its head back to zero degree $\varphi^{k+a} = \varphi^k$ where $a \in \mathbb{R}^1$ is constant.

A number of group members are selected as scroungers. The scroungers will try to get opportunities sharing the resources found by the producer. At kth iteration the random walk toward the producer of the ith scrounger is modeled as

$$X_i^{k+1} = X_i^k + r_3 o\left(X_p^k - X_i^k\right) \quad (15)$$

Here, $r_3 \in \mathbb{R}^n$ is constant in a uniform random sequence which range is (0, 1). In this equation "o" is the Hadamard product, which calculates the entry wise product of the two vectors. In GSO, a few members are considered as disperse member whose perform random search. The disperse members are also called rangers. At the kth iteration, it generates a random head angle φ_i; and after that it chooses a random distance $l_i = a.r_1 l_{max}$ and move to the new point

$$X_i^{k+1} = X_i^k + l_i D_i\left(\varphi^{k+1}\right) \quad (16)$$

Grey wolf optimizer (GWO)

The GWO was developed in 2014 by Mirjalili, et al. based on the hunting behavior of Grey wolf (*Canis lupus*) which belongs to Canidae family [8]. Naturally grey wolves prefer to live in a pack and are considered as apex predators which meaning that their position in food chain is on the top. The main phases of grey wolf hunting are: i) Tracking, chasing, and approaching the prey; ii) Pursuing, encircling, and harassing the prey until it stops moving; and iii) Attack towards the prey. In the pack, α wolves are the best in the pack and after that β wolves are second best who are abide by the α wolves. The third category of wolves are δ. Rest of the members are supposed to be omega (ω).

The main focus illustrated in GWO algorithm the hunting (i.e., searching in optimal point) is guided by α, β, and δ. The ω wolves have to follow these three types of wolves. During hunting, grey wolves encircle the prey. The mathematical model of encircling behavior is illuminated through the following equations:

$$\vec{D} = \left|\vec{C}.\overrightarrow{X_p}(t) - \overrightarrow{X_p}(t)\right| \quad (17)$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A}.\vec{D} \quad (18)$$

Here, t shows the current iteration, $\vec{A}$ and $\vec{C}$ is both coefficient vectors, $\vec{X}_p$ represents the position vector of the prey, and the position vector of a grey wolf is indicated by $\vec{X}$. The vectors $\vec{A}$ and $\vec{C}$ are calculated through these:

$$\vec{A} = 2\vec{a}.\vec{r}_1 - \vec{a} \quad (19)$$

$$\vec{C} = 2.\vec{r}_2 \quad (20)$$

In the equations components $\vec{a}$ are decreased from 2 to 0 linearly over the course of iterations and r_1, r_2 are both random vectors in [0, 1]. The mathematical simulation of the hunting behavior of grey wolves illustrates that the, β, and δ have fair knowledge about the potential location of prey. As a result, the first three best solutions obtained so far are saved and also oblige the other search agents (including the ω). The following equations are used to fulfill this purpose.

$$\vec{D}_\alpha = \left|\vec{C}_1.\vec{X}_\alpha - \vec{X}\right|, \vec{D}_\beta = \left|\vec{C}_2.\vec{X}_\beta - \vec{X}\right|, \vec{D}_\delta = \left|\vec{C}_3.\vec{X}_\delta - \vec{X}\right| \quad (21)$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1.\left(\vec{D}_\alpha\right), \vec{X}_2 = \vec{X}_\beta - \vec{A}_2.\left(\vec{D}_\beta\right), \vec{X}_3 = \vec{X}_\delta - \vec{A}_3.\left(\vec{D}_\delta\right) \quad (22)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (23)$$

Experimental Studies

This section first gives the description of the benchmark functions and parameter setting of each individual algorithm. After that experimental results comparing performance of SI methods in solving the benchmark functions are presented. Finally, experimental analysis has given on selected functions.

Benchmark functions

To evaluate any system benchmark data is commonly used. Table 1 shows the characteristics of 22 benchmark functions that have been considered in this study. The significance of the selected functions is that they have different numbers of range, dimension and type. In the table last column indicates type of a function; type unimodal, multimodal and fixed dimension multimodal are marked as U, M and FDM, respectively. The characteristics of the functions indicate as a suitable diverse test bed. It is worth mentionable that these functions have been widely used by in many existing studies [7,8].

Experimental setup

An experiment has been conducted in this study considering fair settings. Setting of a method is considered as the setting reported in the original work for better outcome. These settings are also widely used in the existing studies. Population size and total iteration were considered as 100 and 1000 to train a function with an algorithm. In PSO acceleration coefficients were 2.0 and the inertia weight was in range [0.2, 0.9]. In ABC the limit was set as limit = (0.6 × population size × dimension). In CSO discovery rate was 0.25. For FFA, the value

of α was 0.5, β_0 was 0.2 and λ was 1. Here reducing the value of α is optional to reduce randomness. In GSO, the initial head angle of each individual, θ^0 was set to be $(\pi/4, \ldots, \pi/4)$. The constant a was given by $round(\sqrt{n+1})$ where n is the dimension of the search space. The maximum pursuit angle φ_{max} was π/a^2. The maximum turning angle α_{max} was set to be $\varphi_{max}/2$. The maximum pursuit distance l_{max} is calculated from the following equation

$$l_{max} = \|\mathrm{U\text{-}L}\| = \sqrt{\sum_{i=1}^{n}(U_i - L_i)^2} \quad (24)$$

Where L_i and U_i are the lower and upper bounds for the ith dimension. The experiment had been conducted on a PC (Intel Core i3@2.10 GHz CPU, 4GB RAM, Windows 8.1 OS, MATLAB 2015).

Experimental results

This section compares proficiency of PSO, ABC, FFA, CSO, GSO and GWO in solving the selected benchmark functions. Table 2 shows the fitness value achieved by a method for the functions; the presented result is the best result from among five independent runs. The optimal

No.	Function Name	Function Index	Range	Dimension	f_{min}	Type
1	Sphere	F1	[-100,100]	30	0	U
2	Schwefel 2.22	F2	[-10,10]	30	0	U
3	Schwefel 1.2	F3	[-100,100]	30	0	U
4	Rosenbrock	F4	[-30,30]	30	0	U
5	Step	F5	[-100,100]	5	0	U
6	Quartic	F6	[-1.28,1.28]	30	0	U
7	Sum of the Squares	F7	[-10,10]	30	0	U
8	Schwefel	F8	[-500,500]	30	-12569.5	M
9	Rastrigin	F9	[-5.12,5.12]	30	0	M
10	Ackley	F10	[-32,32]	30	0	M
11	Griewank	F11	[-600,600]	30	0	M
12	Penalized	F12	[-50,50]	30	0	M
13	Penalized2	F13	[-50,50]	30	0	M
14	Foxholes	F14	[-65,65]	2	1	FDM
15	Six hump camel back	F15	[-5,5]	2	-1.031	FDM
16	Branin	F16	[-5,5]	2	0.398	FDM
17	GoldStein-Price	F17	[-2,2]	2	3	FDM
18	Hartman3	F18	[1,3]	3	-3.86	FDM
19	Hartman6	F19	[0,1]	6	-3.32	FDM
20	Shekel5	F20	[0,10]	4	-10.1532	FDM
21	Shekel7	F21	[0,10]	4	-10.4028	FDM
22	Shekel10	F22	[0,10]	4	-10.5363	FDM

Table 1: Characteristics of benchmark functions.

Function	f_{min}	PSO	ABC	FFA	CSO	GSO	GWO
F1	0	4.24E-17	0.000133253	0.00284646	*0.0525012*	1.05E-08	**0**
F2	0	1.31E-09	5.06E-05	0.217751	*3.98673*	3.68E-05	3.07E-243
F3	0	0.894981	*18695.8*	968.129	612.874	30.6306	3.42E-216
F4	0	17.4573	*226.237*	27.5278	58.1186	22.4814	25.0925
F5	0	**0**	**0**	**0**	*1*	**0**	**0**
F6	0	0.0149869	*0.0372523*	0.0224415	0.0211475	0.0163996	4.49E-05
F7	0	1.69E-16	8.83E-06	*0.0483066*	0.00545648	2.60E-09	**0**
F8	-12569.5	-8304.92	*-6037.71*	-7772.46	-9219.58	**-12569.5**	-8164.71
F9	0	32.8336	*167.403*	24.543	77.0304	2.01486	**0**
F10	0	2.93E-09	0.00439897	0.0218787	*5.39665*	7.38E-05	7.99E-15
F11	0	**0**	0.00511547	0.00234851	*0.146045*	4.38E-08	**0**
F12	0	7.15E-20	*6.67666*	0.000144563	0.99701	6.34E-11	0.0130216
F13	0	3.82E-18	*10.5903*	0.00129743	0.673448	9.46E-10	1.01E-06
F14	1	0.998004	0.998004	*1.99203*	0.998004	0.998004	0.998004
F15	-1.031	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163
F16	0.398	0.397887	0.397887	0.397887	0.397887	0.397887	0.397887
F17	3	**3**	**3**	**3**	**3**	**3**	**3**
F18	-3.86	-0.300479	-0.300479	-0.300479	-0.300479	*-0.300033*	-0.300479
F19	-3.32	-3.322	-3.322	-3.322	-3.322	-3.322	-3.32199
F20	-10.1532	**-10.1532**	**-10.1532**	**-10.1532**	**-10.1532**	**-10.1532**	**-10.1532**
F21	-10.4028	-10.4029	-10.4029	-10.4029	-10.4029	-10.4029	-10.4029
F22	-10.5363	*-10.5364*	*-10.5364*	*-10.5364*	*-10.5364*	*-10.5364*	**-10.5363**
Optimal Count		4	3	3	2	4	8
Best Count		12	8	7	7	8	18

Table 2: Comparison among the SI methods on the basis of fitness achieved for the benchmark functions.

value of a function is placed from Table 1 for better comparison between optimal fitness and achieved fitness by a method. For a particular function, the best value among the six SI methods is shown underlined and if the best outcome is matched with optimal value it is also shown in bold face. On the other hand, the worst fitness value among the methods is shown in italic type. The method wise summary in solving the 22 functions is presented below the table as best count and optimal count. The best count indicates on how many problems a particular method performed best fitness. On the other hand, the optimal count indicates on how many functions a method is achieved optimal fitness values.

From the Table 2 it is observed that among all these methods, GWO has shown the best: it achieved best fitness value among the methods for 18 cases out of 22 functions; among them eight cases (e.g., F1, F5, F7, F9) it achieved optimal values and some other cases showed fitness value closed to optimal value (e.g., F2, F3, F6, F10). GWO is the recent method and its proficiency is identified as the collaborative actions of different wolves. PSO, the pioneer as well as popular SI method, is also found best for 12 cases with optimal value for four cases (i.e., F5, F11, F17 and F20) and closed to optimal values for several cases (e.g., F12, F13). According to result presented in the table, GSO is the third best showing eight best count and four optimal count. On the other hand, ABC, FFA and CSO also achieved best or optimal fitness for several cases but GWO, GSO or PSO also performed similar outcome for that cases.

Table 3 shows the pair wise Win/Draw/Loss count from Table 2 to identify performance of a method with others individually. According to Table 3, GWO is outperformed ABC, FFA and CSO showing better performance for 14 different cases and competitive performance for several cases. GWO found inferior to FFA and CSO for only one case for each, F12 and F8, respectively. PSO is found competitive to GSO showing Win/Draw/Loss values 9/9/4. GSO is shown better than ABC, FFA and CSO showing Win/Draw/Loss values 12/9/1, 13/8/1 and 13/8/1, respectively. On the other hand, ABC, FFA and CSO are found competitive to one another. Finally, GWO is the overall best method for FO and PSO is still promising to solve bench mark functions.

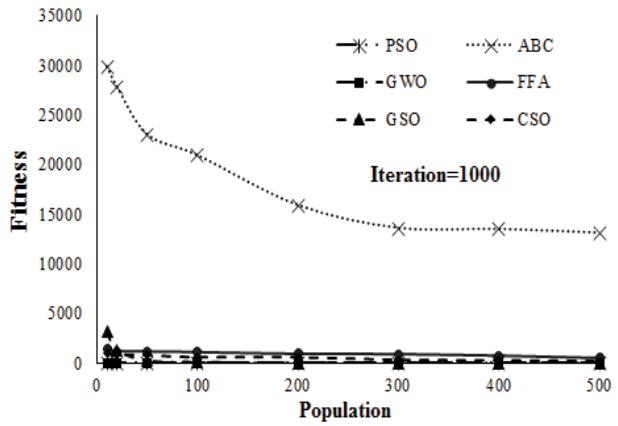
Experimental analysis

This section presents the effect of population and iteration on the SI methods to solve benchmark functions. The experiment has been conducted on selected unimodal (Schwefel 1.2) and multimodal (Schwefel, Rastrigin and Ackley) functions. The functions are selected for better visualization of population and iteration effects on performance. The size of population was varied from 10 to 500 and iteration was varied from 50 to 1000.

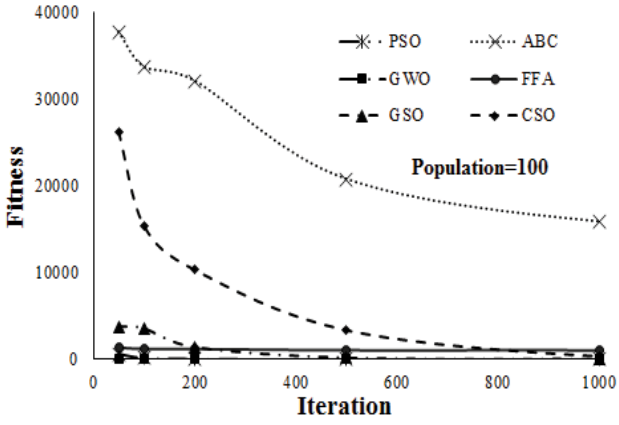
Figure 3 shows the effect of varying population and iteration on Schwefel 1.2. It is observed from Figure 3a that for very small population (e.g., 10) the methods perform worse and improve with population size. Among the methods, ABC shows very bed performance although improves with population size. On the other hand, for fixed 100 population size, ABC and CSO are found to perform very bad for less

	PSO	ABC	FFA	CSO	GSO	GWO
PSO	-	0/10/12	1/9/12	1/9/12	2/9/11	9/9/4
ABC	-	-	7/9/6	7/9/6	12/9/1	14/8/0
FFA	-	-	-	5/8/9	13/8/1	14/7/1
CSO	-	-	-	-	13/8/1	14/7/1
GSO	-	-	-	-	-	11/7/4

Table 3: Pair wise win/draw/loss comparison of result presented in Table 2.



(a) Fitness varying population.



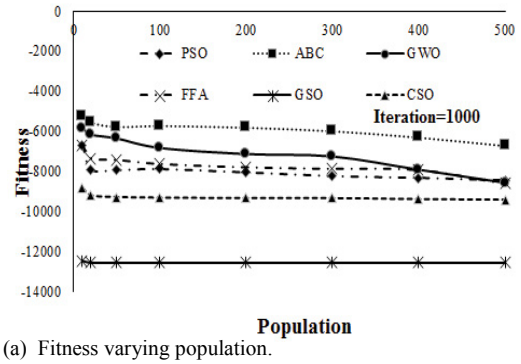
(b) Fitness varying iteration.

Figure 3: Effect of population and iteration on Schwefel 1.2.

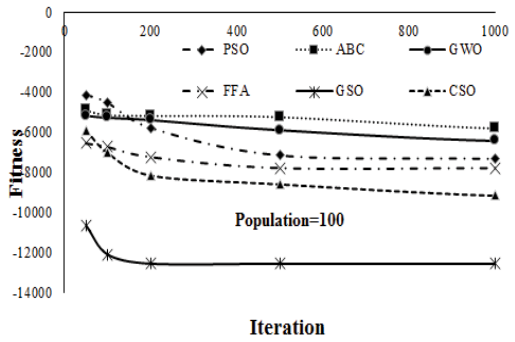
number of iteration (e.g., less than 200) as seen in Figure 3b. Although with iteration performance of ABC and CSO improved but ABC failed to reach other methods. At 1000 iteration, GWO has shown to reach close to the optimal value. For Schwefel 1.2 problem, the sequence of the methods according to performance is GWO, PSO, GSO, CSO, FFA and ABC.

The effect of varying population and iteration on multimodal Schwefel function is shown in Figure 4. As like Figure 3, the effect of population and iteration is that the methods improves performance up to a certain population and iteration. However, GSO is shown to get the optimal -12569.5 for the problem. At a glance, the algorithm sequence according performances on the problem are GSO, CSO, PSO, GWO, FFA and ABC.

The performance on Rastrigin function for the algorithms has been drawn in Figure 5. For the problem ABC and CSO both have showed worse performance at small population. While increasing the population both the methods improved performance but fail to reach other methods. Similar observation of iteration variation is that ABC and CSO are inferior to others. For the problem only GWO achieved optimal value (i.e., 0) with 100 population and 500 (and more) iteration. On the other hand, GSO and PSO showed fair performance for the

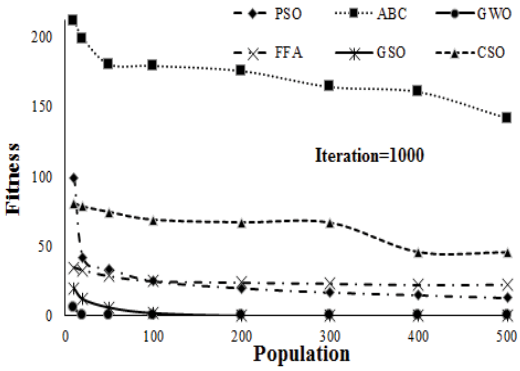


(a) Fitness varying population.

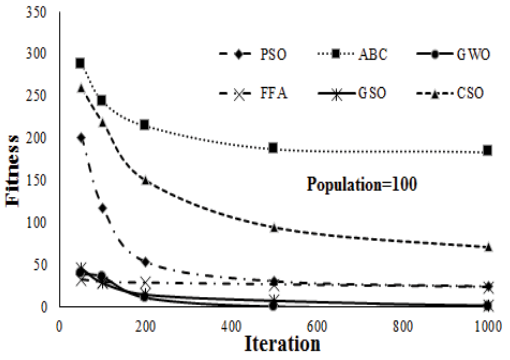


(b) Fitness varying iteration

Figure 4: Effect of population and iteration on Schwefel.



(a) Fitness varying population.



(b) Fitness varying iteration

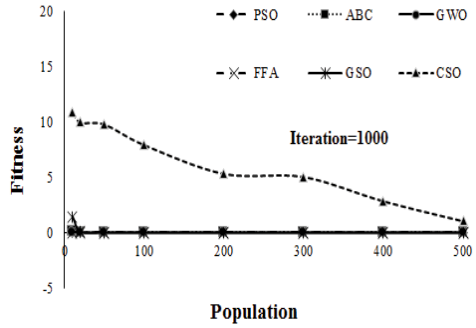
Figure 5: Effect of population and iteration on Rastrigin.

problem. At a glance, the algorithm sequence according performance on the problem is GWO, GSO, FFA, PSO, CSO, and ABC.

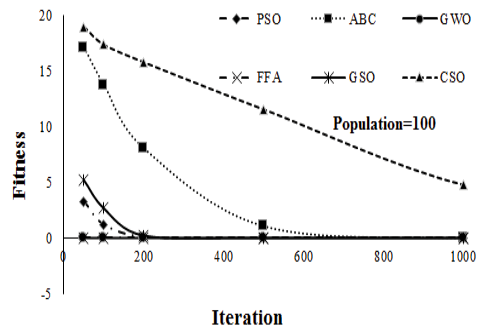
Figure 6 shows the performance of the SI algorithms on Ackley function varying population and iteration. For the function, CSO has showed worse performance at small population and increased with population size. On the other hand, CSO and ABC performed very badly for small number of iteration (e.g., less than 200). Although performance improved for both CSO and ABC with iteration; but CSO is found the worst among the methods. For the function, GWO has showed the best performance achieving fitness closed to optimal. PSO and GSO are also showed fair performance for the function. At a glance, the algorithm sequence according performance on the problems are GWO, PSO, GSO, ABC, FFA and CSO.

Conclusion

Function optimization is the well-studied continuous optimization task which aim is to find best suited parameter values to get optimal value of a function. Recently, nature inspired swarm intelligence methods become increasingly popular to solve various optimization tasks including function optimization. In this study, prominent SI methods have been tested with fair settings and compared their performance in solving a large number of popular benchmark functions. Among the methods, the recently proposed Grey Wolf Optimizer is shown best performance in solving the benchmark functions. The performance of particle swarm optimization, the pioneer SI method, and group search optimization are found relatively better than artificial bee colony, firefly algorithm and cuckoo search optimization.



(a) Fitness varying population.



(b) Fitness varying iteration

Figure 6: Effect of population and iteration on Ackley.

References

1. Varoquaux G (2010) Mathematical optimization.
2. Optimization problem. Retrieved from https://en.wikipedia.org/wiki/Optimization_problem.
3. Pal SK, Rai CS, Singh AP (2012) Comparative study of firefly algorithm and particle swarm optimization for noisy non-linear optimization problems. International Journal on Intelligent Systems and Applications 10: 50-57.
4. Vesterstrøm J, Thomsen R (2004) A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. Evolutionary Computation CEC2004. Congress.
5. Ab Wahab MN, Nefti-Meziani S, Atyabi A (2015) A comprehensive review of swarm optimization algorithms. PLoS ONE 10: e0122827.
6. Yang XS, Deb S (2009) Cuckoo search via Levy flights. Proceedings of World Congress on Nature Biologically Inspired Computing.
7. He S, Wu QH, Saunders JR (2009) Group search optimizer: An optimization algorithm inspired by animal searching behavior. IEEE Transactions on Evolutionary Computation 13: 973-990.
8. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Advances in Engineering Software 69: 46-61.
9. Blum C, Li X (2008) Swarm intelligence in optimization. In: Natural computing series, swarm intelligence. Blum C, Merkle D (Eds) Berlin, Heidelberg: Springer Berlin Heidelberg.
10. Beni G (1988) The concept of cellular robotic system. Intelligent Control, 1988. Proceedings, IEEE International Symposium on, Arlington, VA.
11. Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence: From natural to artificial systems. Oxford University Press, New York.
12. Kennedy J, Eberhart R (1995) Particle swarm optimization. Neural Networks. Proceedings. IEEE International Conference on, Perth.
13. Blodin J (2009) Particle swarm optimization: A tutorial.
14. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.
15. Yang XS (2009) Firefly algorithms for multimodal optimization. In: Stochastic algorithms: Foundations and applications, SAGA 2009, Lecture notes in computer sciences.
16. Pulliam HR, Millikan GE (1983) Social organization in the non-reproductive season. Animal Behavior 6: 169-197.
17. Brockmann J, Barnard CJ (1979) Kleptoparasitism in birds. Animal Behavior 27: 546-555.
18. Bell JW (1999) Searching behavior–The behavioral ecology of finding resources (Chapman and Hall Animal Behavior Series). London, U.K.: Chapman and Hall.
19. Reynolds AM, Rhodes CJ (2009) The Lévy flight paradigm: random search patterns and mechanisms. Ecology 90: 877-887.