Applied Soft Computing 56 (2017) 520–540

Contents lists available at ScienceDirect

Applied Soft Computing

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

Weighted Superposition Attraction (WSA): A swarm intelligence algorithm for optimization problems – Part 1: Unconstrained optimization

Adil Baykasoğlu *, Şener Akpinar

Dokuz Eylül University, Faculty of Engineering, Department of Industrial Engineering, Izmir, Turkey

ARTICLE INFO

Article history:
Received 14 May 2015
Received in revised form
20 September 2015
Accepted 16 October 2015
Available online 27 October 2015

Keywords:
Meta-heuristics
Swarm intelligence
Global optimization

ABSTRACT

This paper is the first one of the two papers entitled "Weighted Superposition Attraction (WSA)", which is based on two basic mechanisms, "superposition" and "attracted movement of agents", that are observable in many systems. Dividing this paper into two parts raised as a necessity because of their individually comprehensive contents. If we wanted to write these papers as a single paper we had to write more compact as distinct from its current versions because of the space requirements. So, writing them as a single paper would not be as effective as we desired.

In many natural phenomena it is possible to compute superposition or weighted superposition of active fields like light sources, electric fields, sound sources, heat sources, etc.; the same may also be possible for social systems as well. An agent (particle, human, electron, etc.) may be supposed to move towards superposition if it is attractive to it. As systems status changes the superposition also changes; so it needs to be recomputed. This is the main idea behind the WSA algorithm, which mainly attempts to realize this superposition principle in combination with the attracted movement of agents as a search procedure for solving optimization problems in an effective manner. In this current part, the performance of the proposed WSA algorithm is tested on the well-known unconstrained continuous optimization functions, through a set of computational study. The comparison with some other search algorithms is performed in terms of solution quality and computational time. The experimental results clearly indicate the effectiveness of the WSA algorithm.

1. Introduction

Algorithmic design is an exploratory research field, since numerous real world problems need to be solved optimality or near-optimality. Numerous researchers attempted to develop effective algorithms within the purpose of optimization, which is the name of the process of systematically making a system or design as efficient as possible [1]. In this respect, effectiveness arises as a key feature for the efforts to design algorithms for optimization purposes.

From the optimization point of view, the concept of effectiveness of a solution approach covers robustness, ability to find optimal or near optimal solutions in a reasonable time limit and having the potential to be used widely. According to these important characteristics, solution approaches may be classified as mathematical programming, heuristics and meta-heuristics methods. Mathematical programming methods are generally developed for formally describing the problem on hand. They guarantee the optimal solution of a problem; however, they require large amount of time for providing the optimal solution as the size of the problem gets larger and mathematically complex (non-convex, ill-defined, etc.). Hence, the mathematical programming approaches have not the desired level of effectiveness for many cases. Heuristic methods may provide optimal or near-optimal solutions for a problem in shorter times. Nevertheless, they are generally developed on a problem dependent basis and this situation prevents widespread implementations, thus decreases the effectiveness level of them. On the other hand, meta-heuristic algorithms which are one of the pillars of soft computing have the ability to provide satisfactory results for different type of optimization problems in reasonable time limits. Thus, it would not be wrong to claim that meta-heuristic algorithms are usually more effective than mathematical programming and heuristic methods for many complex problem settings.

* Corresponding author. Tel.: +90 232 3017600; fax: +90 232 3017608.
E-mail addresses: adil.baykasoglu@deu.edu.tr (A. Baykasoğlu), sener.akpinar@deu.edu.tr (Ş. Akpinar).

http://dx.doi.org/10.1016/j.asoc.2015.10.036
1568-4946/

Nomenclature

Maxiter	iteration number (stopping condition)
Iteration	current iteration number
AA	number of artificial agents
D	number of dimensions of the problem
τ	user defined parameter
λ	user defined parameter
φ	user defined parameter
UL	upper limit for the dimensions
LL	lower limit for the dimensions
$f(i)$	fitness of the current point of agent i
$f(tar)$	fitness of the target point
weight	weight of the current point of an agent
$\vec{x}$	current position vector of an agent
$\rightarrow tar$	position vector of the target point
$\rightarrow gap$	vector combines an agent to target point
$\rightarrow direct$	move direction vector of an agent
sign()	signum function
sl	step length

Meta-heuristic algorithms are generally based on a specific philosophy inspired by a source. Probably, natural processes of evaluation and swarm intelligence are the most popular and widely used sources of inspiration while designing meta-heuristic approaches. Thus, nature inspired meta-heuristics have been mainly grouped as evolutionary algorithms (genetic algorithm (GA) [2] and differential evolution (DE) [3]) and swarm intelligence algorithms (ant colony optimization (ACO) [4], particle swarm optimization (PSO) [5], bee colony optimization (BCO) [6], and bees algorithm (BA) [7]). On the other hand, non-nature inspired meta-heuristic algorithms (Tabu search (TS) [8,9], and simulated annealing (SA) [10]) were also designed and provided satisfactory performances on several optimization problems. In this current study, we propose a new swarm intelligence algorithm and test the effectiveness of this algorithm on a set of standard unconstrained continuous optimization problems.

A great number of real world problems are included in the field of continuous optimization, a branch of applied mathematics. This kind of optimization is different from the combinatorial optimization in the type of decision variables. The optimization problems belonging to this class of optimization may have one or more real continuous decision variables each one may have an infinite number of values from a particular interval. Thus, the search spaces of continuous optimization problems are infinite. For that reason, effectively exploring the search spaces of this type of problems is a challenging research field. Within this context, various meta-heuristic solution approaches (harmony search algorithm (HSA) [11], electromagnetic algorithm (EA) [12], seeker optimization algorithm (SOA) [13], firefly algorithm (FA) [14], gravitational search algorithm (GSA) [15], bat-inspired algorithm (BIA) [16], cuckoo optimization algorithm (COA) [17], exchange market algorithm (EMA) [18], forest optimization algorithm (FOA) [19], bird mating optimizer (BMO) [20], krill herd algorithm (KHA) [21], social spider optimization (SSO) [22], differential search algorithm (DSA) [23], teaching–learning-based optimization (TLBO) [24]) were developed by numerous researchers.

In this paper, we introduce a new swarm based optimization algorithm, which is based on two basic mechanisms, superposition and attracted movement of agents, that are observable in many systems. It should be mentioned here that in most of the swarm based algorithms including PSO it is assumed that an agent can identify the best agent in the population and adjust its direction towards to it. In realty however, it is not easy if not impossible to accurately identify the best source due to many interferences, on the other hand agents may identify the superposition (as it is usually performed to determine the overall effect of more than one source in physics and other related scientific fields under several assumptions "http://en.wikipedia.org/wiki/Superposition_principle") and may decide to direct towards to this position. This basic mechanism is successfully implemented as an optimization algorithm in the present study. Within this framework, WSA algorithm determines a target point within the search space via the superposition principle and directs the agents of the swarm towards this point. That is to say, the target point is the superposition of the currently discovered points of the search space by the agents of the swarm. Once the target point is determined, agents explore the search space whether moving towards target point or a randomly selected direction.

This paper is the first one of a two-part paper which main aim is to introduce a new swarm based solution approach for complex optimization problems. The aim of this current part is to introduce the WSA algorithm and to provide a comprehensive performance evaluation test of WSA on the unconstrained continuous optimization problems. For a new algorithm, its effectiveness must be proved on both unconstrained and constrained optimization problems to our point of view. Within this context, the second part [25] is about the performance evaluation test of WSA on constrained optimization problems and it also has a comprehensive content. Since both parts of the paper have individually comprehensive contents and these contents required to effectively introduce the new algorithm, writing them as separate parts makes the paper much more effective. Additionally, writing them as a single paper would be resulted to shorten their comprehensive contents due to space requirements and to reduce the effectiveness of the paper.

Decision making is a hard process for living beings, especially for human beings. Through this process a decision-maker takes into account some criteria rather than a single criterion and each criterion's features have impacts over the judgement of the decision-maker. These impacts may be thought as the attractiveness of the related criterion. From this point of view, it is possible to say that a decision is a result of the attractiveness of some criteria. In other words, a decision is generally given in accordance with some criteria thus it arises as a superposition of some criteria related to their individual attractiveness. To our point of view, this decision making procedure is also valid and applicable for the optimization purposes and this procedure constitutes the rationale of the newly developed WSA algorithm. In WSA, decision makers are artificial agents and they have to give decisions about their search directions in order to effectively discovering the solution space of an optimization problem. Each artificial agent has its own point over the search space and this point has an attractant effect over the other artificial agents. This attractiveness of an artificial agent is a measure in proportional to its current positions quality, which is actually the objective function value at this point. As mentioned above, an agent determines its search direction while considering all the other agents' attractiveness not a single one's and this process is more realistic from the optimization point of. As a result, WSA tries to simulate this decision making process, a combination of attractant movement and superposition principle, as a new swarm based solution approach.

The remainder of this paper is organized as follows. Some related works are discussed in Section 2. The newly developed swarm based algorithm is defined in Section 3. Comparative computational study is given in Section 4, and the discussions and conclusions are presented in Section 5.

1. Initialize the algorithm parameters, the best solution and the best fitness
2. Generate a pre-defined number of initial solutions
3. Evaluate fitness values of the initial solutions, and update the best solution and the best fitness
4. *Iteration* = 1

while *Iteration* <= *Maxiter*

- rank solutions according to their fitness values
- assign a weight to each solution by considering their ranks
- determine a target point (superposition) to move the solutions towards it
- evaluate the fitness value of the target point
- determine search directions for each solution by considering the target point and its fitness value
- move each solution towards its determined direction
- evaluate fitness values of each solution, and update the best solution and the best fitness

Iteration = *Iteration* + 1

end while

Fig. 1. Main steps of the WSA algorithm.

2. Related works

As mentioned above algorithmic design is a challenging research field, since many real world problems required to be solved as effectively and efficiently as possible. Within this context, some researchers attempt to improve the existing algorithms by modification or hybridization while some others attempt to develop new search strategies. This current paper is also an attempt to develop a new swarm based solution approach of WSA algorithm. Before introducing the WSA algorithm, some related works are analysed in the following sub-sections in order to highlight how WSA innovate from the previous related approaches is. These analyses will be done on PSO, EA and FA, since PSO tries to discover the search space via its agents like WSA; EA realizes superposition principle during its search as WSA does; FA has a search strategy on the basis of attractiveness as WSA has. At first glance, it may be thought that WSA is a variant of PSO, EA or FA because of the abovementioned similarities; however WSA is a new, different and innovative search procedure as discussed below.

2.1. Particle swarm optimization

The PSO algorithm is a stochastic and iterative learning algorithm. PSO was originally developed to solve real-world optimization problems [5] such as numerical optimization [26], scheduling [27–30], assembly line balancing [31–33], power system [34], facility layout [35], lot sizing [36] and network design [37]. On the other hand, premature convergence arises as a major problem of PSO algorithm [38] and therefore many researchers tried to overcome this drawback of PSO while tuning the parameters [39,40], designing different population topologies [41–43], hybridizing PSO with other evolutionary optimization operators [44,45], adopting new learning strategies [46,47] and using multi-swarm techniques [48,49].

PSO has a search strategy based on the agents' movements as WSA; however the movement strategies are different. In WSA, all agents have effect on the determination of search direction of an agent, while in PSO search direction is determined via the guidance of the best previous position of the related particle and the best previous position among all the particles in the population. Besides this main difference, PSO simulates the social behaviour of birds while WSA inspires by the superposition principle. So, the rationales of these algorithms are different.

2.2. Electromagnetic algorithm

The EA is inspired by the electromagnetism theory and it search strategy based on an attraction–repulsion mechanism and the superposition principle of electromagnetism theory [12], and has been successfully applied to a variety of optimization problems [50–55]. The standard EA has the advantages of multi-point search and faster convergence procedure [12] while its major drawback is computational complexity due to its stochastic and coordinate by coordinate local search procedure [51]. Therefore many researchers tried to overcome this drawback of EA while hybridizing EA with other meta-heuristic algorithms [52,56,57] and modifying the basic EA [58].

EA calculates total force for each point according to the superposition principle of electromagnetism theory and then every point can move in the direction of its total force by random length. On the other hand, WSA determines a single point according to superposition principle and this point may be a target point due to its attractiveness for the other points. All the other points may or may not move towards this target point. As a result, EA and WSA have different search strategies, however both of them use the superposition principle.

$weight = \text{zeros}\ (1, AA)$

$\overrightarrow{tar} = \text{zeros}\ (1, D)$

for ($i = 1$ to AA) **do**

  $weight\ (1,\ i) = i^{(-1)^* \tau}$

  for ($j = 1$ to D) **do**

    $\overrightarrow{tar}(1, j) = \overrightarrow{tar}(1, j) + \vec{x}(i, j) * weight(1, i)$

  endfor

endfor

Fig. 2. Target point determination procedure.

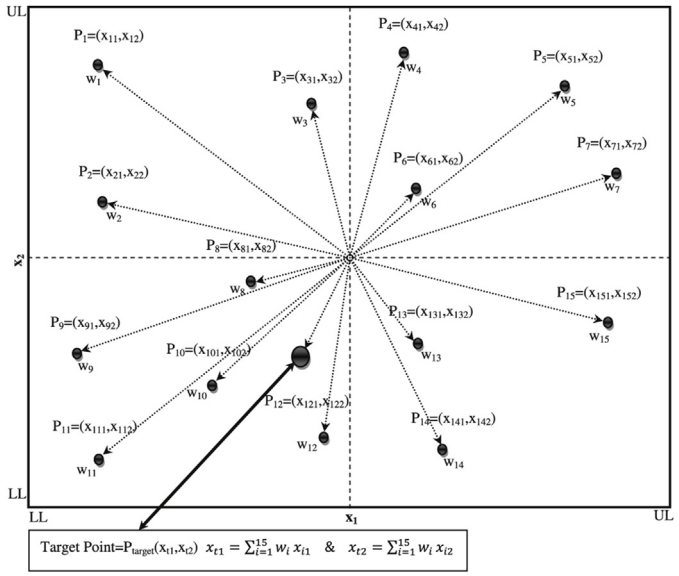

Fig. 3. Target point determination.

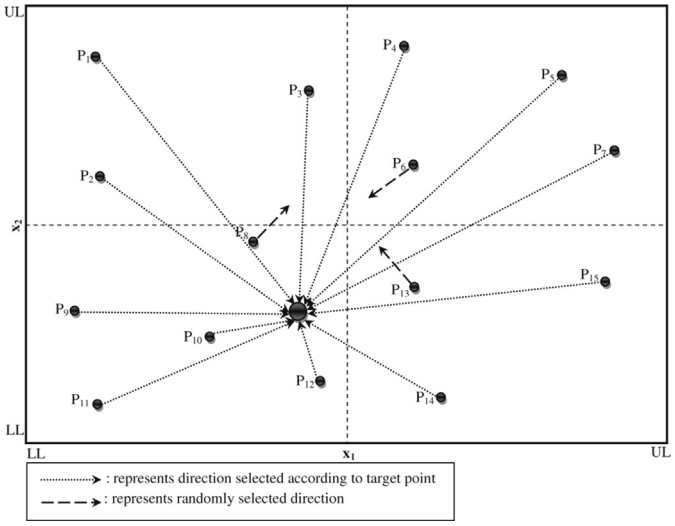

Fig. 5. Search direction determination.

$\overrightarrow{gap} = zeros(AA, D)$
$\overrightarrow{direct} = zeros(AA, D)$
for (i = 1 to AA) **do**
  if $f(i) >= f(tar)$
    for (j = 1 to D) **do**
      $\overrightarrow{gap}(i,j) = \overrightarrow{tar}(1,j) - \vec{x}(i,j)$
    endfor
    for (d = 1 to D) **do**
      $\overrightarrow{direct}(i,d) = sign(\ \overrightarrow{gap}(i,d))$
    endfor
  elseif $f(i) < f(tar)$
    if $rand() < e^{(f(i)\ -\ f(tar))}$
      for (j = 1 to D) **do**
        $\overrightarrow{gap}(i,j) = \overrightarrow{tar}(1,j) - \vec{x}(i,j)$
      endfor
      for (d = 1 to D) **do**
        $\overrightarrow{direct}(i,d) = sign(\ \overrightarrow{gap}(i,d))$
      endfor
    else
      for (d = 1 to D) **do**
        $\overrightarrow{direct}(i,d) = sign(\ -1 + (1+1) * rand())$
      endfor
    endif
  endif
endfor

Fig. 4. Search direction determination procedure.

2.3. Firefly algorithm

The FA is a nature-inspired solution procedure which simulates the social behaviour of fireflies [14]. FA has a satisfactory performance in tackling various optimization problems [59–67]; however it may fall to local optima because of its tendency to premature convergence [66]. To overcome this drawback of FA many variants of FA have been developed by various researchers. For the comprehensive surveys about FA the reader can refer to Fister et al. [68,69], Yang and He [70] or Baykasoğlu and Ozsoydan [62].

FA uses an attractiveness based search strategy like WSA. But FA formulates the attractiveness between fireflies as function of distance between each other and the light absorption coefficient, while WSA formulates the attractiveness via the superposition principle. Therefore search ideas and characteristics of these algorithms are different.

3. Weighted Superposition Attraction

This section introduces the newly developed swarm based optimization algorithm, Weighted Superposition Attraction (WSA), which is based on the superposition principle in combination with the attracted movement of agents. A standard search algorithm discovers the search space of an optimization problem via some mechanisms, which provide the algorithm to visit and evaluate different points of the search space, as WSA does. Fig. 1 illustrates the main steps of the proposed WSA algorithm.

In the relevant literature, search algorithms are usually classified into two main groups, improvement and constructive type of algorithms. An improvement type of algorithm starts with one or more solutions and tries to improve these solutions via its specific mechanisms. On the other hand, a constructive type of algorithm generates solutions and gains some information from each solution. After that, this information is used to generate new solutions. In other words, an improvement type of algorithm discovers the search space by differentiating the existing solution/solutions, while a constructive type of algorithm discovers the search space by generating new solutions during each step in line with the formerly generated information. Within the framework of these explanations, WSA algorithm must be included in the class of improvement type of algorithms.

As all of the improvement type of algorithms, WSA also initiates its search via a pre-defined number of solutions of an optimization problem and tries to improve the quality of these solutions with some specific mechanisms. From the optimization point of view, an artificial agent may correspond to a solution of an optimization problem. An artificial agent can be defined via its own position and the quality of its current position, while a solution of an optimization problem can be defined via its location and fitness value.

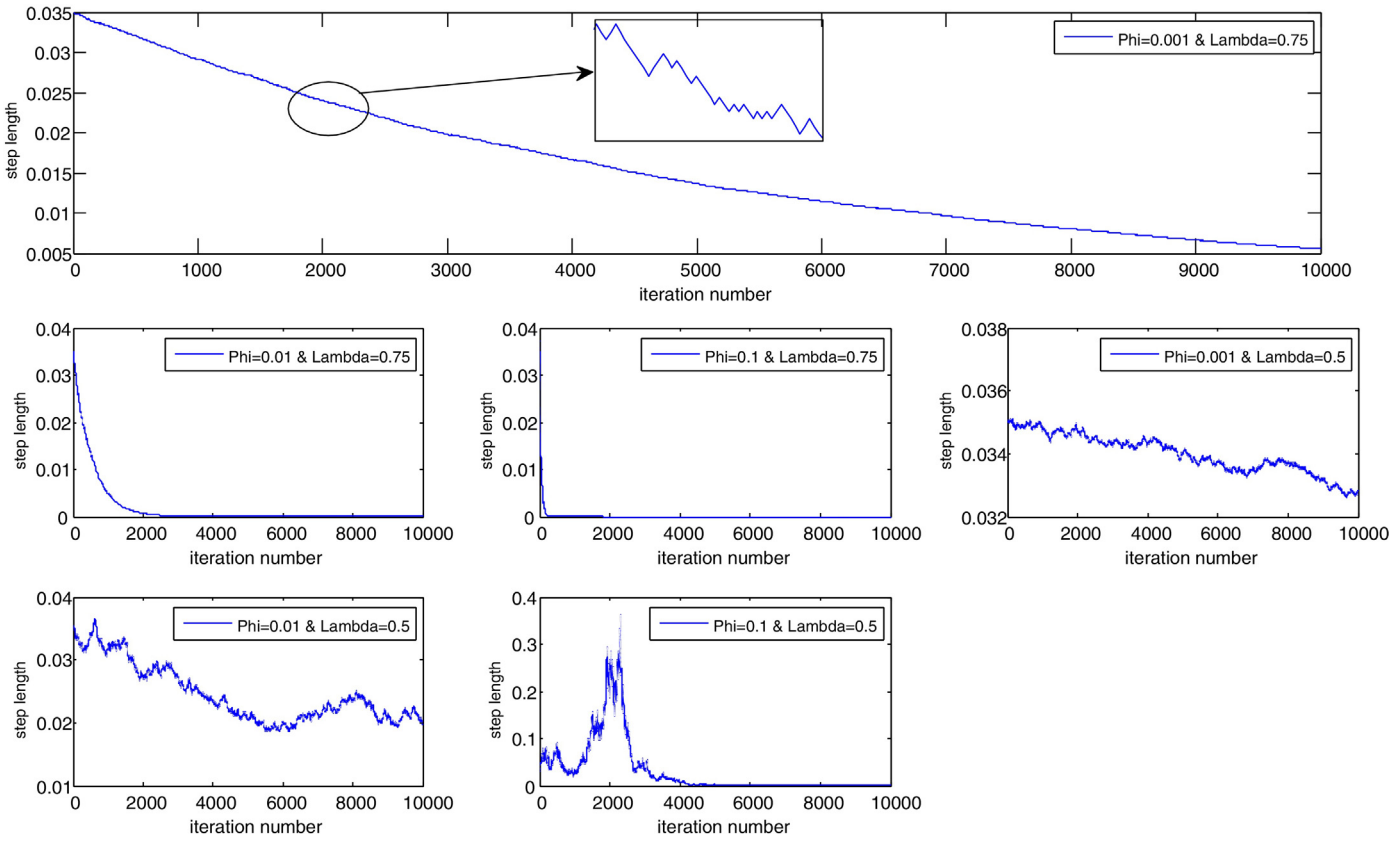

Fig. 6. General behaviour of the step sizing function.

WSA algorithm determines the search directions of the agents of a swarm by realizing the superposition principle in combination with the attracted movement of agents through a neighbour generation mechanism which will be explained in Section 3.2.

3.1. Initializing WSA algorithm

Initialization phase of WSA algorithm consist of determining the values of the required parameters and generating a pre-defined number of initial solutions. Like other search algorithms, WSA also requires some parameters to be set. The parameters and their definitions related to WSA are going to be discussed in the subsequent sections, where necessary. Furthermore, WSA algorithm starts its search from the points regarding to the randomly generated initial solutions. And it must be noted that the number of these randomly generated solutions is one of the parameters related to WSA algorithm.

if $rand() <= \lambda$
  $sl = sl - e^{(mod((k/k+1),1))} * \varphi * sl$
else
  $sl = sl + e^{(mod((k/k+1),1))} * \varphi * sl$
endif
for (i=1 to AA) **do**
  for (d=1 to D) **do**
    $\vec{x}(i,d) = \vec{x}(i,d) + sl * \overrightarrow{direct}(i,d) * abs(\vec{x}(i,d))$
    if $initial\ (i,d) < LL$
      $initial(i,d) = LL$
    elseif $initial(i,d) > UL$
      $initial(i,d) = UL$
    endif
  endfor
endfor

Fig. 7. Position update procedure.

3.2. Neighbour generation

As mentioned earlier, artificial agents discover the search space by moving from one point to another until a satisfactory point is found and this behaviour of the artificial agents constitutes the rationale of WSA algorithm like other search procedures. At this point, the questions are “How they choose their move patterns?” and “How can it be modelled within the optimization point of view?” The artificial agents of WSA algorithm choose their moving patterns via the guidance of the superposition of the currently discovered points. The superposition principle states that the individual responses caused by two or more stimuli at a given place and time may be modelled as a single response, which is the sum of the responses caused by each stimulus individually. The stimuli and responses may be some objects, which satisfy certain axioms, such as numbers, functions, vectors, vector fields, time-varying signals, etc. It should be noted here that a superposition is interpreted as a vector sum when vectors or vector fields are involved, since WSA algorithm deals with continuous optimization problems and each point belonging to such a problem’s search space is actually defined as a vector within the framework of continuous optimization. According to this superposition principle, WSA algorithm determines the superposition of its agent’s positions in each iteration. As systems status changes the superposition also changes; so it need to be recomputed. Additionally, superposition is also a point thereby a solution with its fitness value and an agent

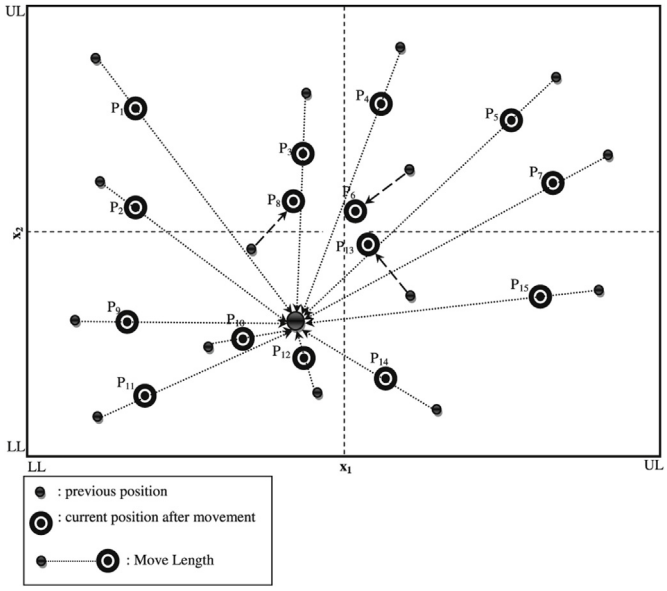

Fig. 8. Position updating.

of WSA algorithm usually moves towards to this point if it is attractive to it. This decision making mechanism could be discussed as follows.

Each currently discovered point by an agent has a fitness value and would have some attractiveness over the other agents in proportion to its fitness value. It could be supposed that this proportional attractiveness of the agents form an artificial agent discovered a point in the search space with its fitness value. WSA algorithm determines this artificial agent's position via the superposition principle such that the weighted sum of the agents' position vectors forms the position vector of this artificial agent and the weights are proportionally determined according to agents' current position finesses. Thus, this superposition would attract the artificial agents while choosing their own move direction. From the optimization point of view, this superposition could be assumed as a single point that should be determined to direct the agents towards this point during the exploration of the search space. At this stage, another question, "How long will the artificial agents move from its current location?" stands out. To respond to this question, a distance must be determined by WSA algorithm. More specifically, the neighbour generation mechanism of WSA algorithm has two major components; search direction (Section 3.2.1) and step length (Section 3.2.2), which have significant effects on the performance of the WSA algorithm.

3.2.1. Search direction

At the initializing phase, WSA algorithm positions a pre-defined number of artificial agents to the randomly selected points within the search space. Each artificial agent explores its own point and must gain some necessary information, fitness value and position as a vector, about the related point. After that, WSA uses all of the information collected by the artificial agents in order to determine the aforementioned superposition and direct the artificial agents towards this point. Hereafter, this superposition will be named as the target point.

While determining the target point, WSA algorithm assigns some weights to each artificial agent with regard to their current point's fitness value. In line with this purpose, WSA algorithm firstly ranks all of the artificial agents according to their fitness values in ascending and descending orders for minimization and maximization purposes, respectively. And, the weights are assigned to artificial agents in accordance with their ranks; the lower the rank has an artificial agent, the higher the weight will be assigned to it. The procedure that determines the target point by using the weights is depicted in Fig. 2 and portrayed in Fig. 3.

Table 1
Parameters of WSA.

Parameter	Definition	Levels
τ	User defined parameter	$\tau(1)=0.2-\tau(2)=0.5-\tau(3)=0.8$
sl_0	Initial step length	$sl_0(1)=0.035-sl_0(2)=0.05-sl_0(3)=0.065$
φ	User defined parameter	$\varphi(1)=0.01-\varphi(2)=0.001-\varphi(3)=0.0001$
λ	User defined parameter	$\lambda(1)=0.75-\lambda(2)=0.85-\lambda(3)=0.95$

It should be noted here that the parameter τ plays a critical role in determining the target point as it controls the influence of artificial agents (vectors) to the target point. The best artificial agent will certainly be used in determining the target point as any powers of one is always one. Rest of the artificial agents will have an effect which is based on their rank and τ. As τ gets bigger their influence will disappear. If τ is zero all of the artificial agents will have the same influence. Once the target point is determined, each artificial agent would decide whether to move towards to this point or not. Every artificial agent gives this decision while comparing its own fitness value to the target point's fitness value. An artificial agent will certainly move towards to the target point, if the target point is better than its current position. Otherwise, an artificial agent may move towards the target point or a randomly selected direction. The related artificial agents give this decision by using a randomly generated number and it compares this number via the value computed as $e^{(f(i)-f(tar))}$, where $f(i)$ is fitness value of the current position of the related artificial agent i and $f(tar)$ is the fitness value of the target point. If the random number is lower than the obtained value, then the related artificial agent decides to move towards the target point. Otherwise, it decides to move towards a randomly selected direction. The procedure used by the artificial agents for determining their search direction is given in Fig. 4 and portrayed in Fig. 5.

3.2.2. Step length

Neighbour generation, consist in determining a search direction with a step length, which is a crucial step in developing effective search algorithms [71] and denotes updating the decision variables according to the newly determined values. By this way, the artificial agents will be able to move from one position to another for discovering its current position's neighbour area. Within the scope of this paper, we deal via the multi-dimensional continuous optimization problems and WSA algorithm determines search directions for all dimensions of an optimization problem as mentioned above. Now, it is required to determine the step lengths for all dimensions and update each dimension by using a position updating mechanism [72]. This mechanism is given by Eq. (1) and artificial agent i ($1\le i\le AA$) update its position on dimension j ($1\le j\le D$) at iteration t as formalized by Eq. (1).

$$x_{ij}(t+1)=x_{ij}(t)+sl(t)\times d_{ij}(t)\times[\![x_{ij}(t)]\!] \quad (1)$$

where $x_{ij}(t)$ is the value of the position of artificial agent i on dimension j at iteration t, $sl(t)$ is the value of step length at iteration t, $d_{ij}(t)$ is the search direction of artificial agent i on dimension j at iteration t, $[\![x_{ij}(t)]\!]$ is the norm of the position vector of artificial agent i on dimension j at iteration t, and $d_{ij}(t)\in\{-1,0,1\}$.

WSA updates the position of an artificial agent via the second term ($sl(t)\times d_{ij}(t)\times[\![x_{ij}(t)]\!]$) of the right hand side of Eq. (1). From the optimization point of view, properly determining the step length provides the algorithm to have a satisfactory performance by providing a high level of intensification. Mainly two types of

Table 2
Analysis of variance for S/N ratios and RPD values.

	Parameter	Degree of freedom	Sum of squares	Mean of squares	*F* distribution	*p* Value
ANOVA results for S/N ratios	τ	2	3.189	1.594	244.37	0.000*
	sl_0	2	0.577	0.288	44.26	0.000*
	φ	2	0.136	0.068	10.43	0.001*
	λ	2	0.150	0.075	11.56	0.001*
	Error	18	0.117	0.006		
	Total	26	4.171			
ANOVA results for RPD values	τ	2	24.972	12.486	167.72	0.005*
	sl_0	2	25.287	12.644	169.83	0.005*
	φ	2	7.641	3.821	51.32	0.019*
	λ	2	7.731	3.587	48.17	0.020*
	Error	18	1.340	0.074		
	Total	26	66.413			

* The parameter has a meaningful effect at the 5% significance level.

Table 3
The results for Taguchi design.

Parameter level	The mean S/N ratios	The mean RPD values
$\tau(1)$	−43.71	15.36
$\tau(2)$	−41.76	12.52
$\tau(3)$*	−37.00	7.08
$sl_0(1)$*	−39.74	10.10
$sl_0(2)$	−40.61	11.24
$sl_0(3)$	−42.11	13.61
$\varphi(1)$	−41.35	12.39
$\varphi(2)$*	−40.22	10.69
$\varphi(3)$	−40.89	11.84
$\lambda(1)$*	−40.55	10.98
$\lambda(2)$	−41.47	12.69
$\lambda(3)$	−40.75	11.28

* Represents the optimum level.

step sizing strategies (adaptive and variable) were used by various approaches [73–77]; however there is not an agreed method in literature. WSA uses a step sizing function developed by [72] on the basis of the one that represented by [71]. This function is given by Eq. (2) and it starts with an initial step length (sl_0) and varies the step length as the search progress via a proportional rule. This proportional rule uses a randomly generated number r between 0 and 1 and a user defined parameter λ (*Lambda*). The general behaviour of the step sizing function during the iterations is depicted in Fig. 6.

$$sl(t+1) = \begin{cases} sl(t) - e^{t/(t-1)} \times \varphi \times sl(t) & \text{if } r \le \lambda \\ sl(t) + e^{t/(t-1)} \times \varphi \times sl(t) & \text{if } r > \lambda \end{cases} \quad (2)$$

In Eq. (2), t is the number of iteration and φ (*Phi*) is a user defined parameter. The step sizing function has a decreasing trend as the iteration number increases; however it increases the step length for some iteration during the search. By this way, WSA achieves the capabilities of intensively exploring the search space and easily jumping out of the local extremum points. WSA updates the positions of artificial agents during their search by using the procedure presented in Fig. 7, and portrayed in Fig. 8. At this point, it must be noted that the procedures given in Figs. 4 and 7 specify the moving patterns of the artificial agents in the search space during their explorations. If WSA executes these two procedures in accordance with each other, then it will be achieve a significant level of performance.

The position update procedure as mentioned above may cause the agents to exceed the search limits for some dimensions of the problem on hand. Then, this situation reflects as a feasibility problem. WSA overcomes this problem by equalizing the related dimension to UL if it exceeds UL and LL if it exceeds LL.

Table 4
Test problems for the first part of the performance evaluation of WSA.

	Name of the problem	Dimension	Type
1	Ackley's Problem (ACK)	10	Multimodal
2	Aluffi-Pentini's Problem (AP)	2	Multimodal
3	Becker and Lago Problem (BL)	2	Multimodal
4	Bohachevsky 1 Problem (B1)	2	Multimodal
5	Bohachevsky 2 Problem (B2)	2	Multimodal
6	Branin Problem (BR)	2	Multimodal
7	Camel Back-3 Three Hump Problem (CB3)	2	Multimodal
8	Camel Back-6 Six Hump Problem (CB6)	2	Multimodal
9	Cosine Mixture Problem (CM)	2, 4	Unimodal
10	Dekkers and Aarts Problem (DA)	2	Multimodal
11	Easom Problem (EP)	2	Unimodal
12	Epistatic Michalewicz Problem (EM)	5	Multimodal
13	Exponential Problem (EXP)	10	Unimodal
14	Goldstein and Price Problem (GP)	2	Multimodal
15	Griewank Problem (GW)	10	Multimodal
16	Gulf Research Problem (GRP)	3	Unimodal
17	Hartman 3 Problem (H3)	3	Multimodal
18	Hartman 6 Problem (H6)	6	Multimodal
19	Helical Valley Problem (HV)	3	Unimodal
20	Hosaki Problem (HSK)	2	Multimodal
21	Kowalik Problem (KL)	4	Unimodal
22	Levy and Montalvo 1 Problem (LM1)	3	Multimodal
23	Levy and Montalvo 2 Problem (LM2)	5, 10	Multimodal
24	McCormick problem (MC)	2	Multimodal
25	Meyer and Roth Problem (MR)	3	Unimodal
26	Miele and Cantrell Problem (MCP)	4	Multimodal
27	Modified Langerman Problem (ML)	10	Multimodal
28	Modified Rosenbrock Problem (MRP)	2	Multimodal
29	Multi-Gaussian Problem (MGP)	2	Multimodal
30	Neumaier 2 Problem (NF2)	4	Unimodal
31	Neumaier 3 Problem (NF3)	10	Multimodal
32	Odd Square Problem (OSP)	10	Multimodal
33	Paviani's Problem (PP)	10	Unimodal
34	Periodic Problem (PRD)	2	Multimodal
35	Powell's Quadratic Problem (PQ)	4	Unimodal
36	Price's Transistor Modelling Problem (PTM)	9	Multimodal
37	Rastrigin Problem (RG)	10	Multimodal
38	Rosenbrock Problem (RB)	10	Unimodal
39	Salomon Problem (SAL)	5, 10	Multimodal
40	Schaffer 1 Problem (SF1)	2	Multimodal
41	Schaffer 2 Problem (SF2)	2	Multimodal
42	Shubert Problem (SBT)	2	Multimodal
43	Schwefel Problem (SWF)	10	Multimodal
44	Shekel 5 Problem (S5)	4	Multimodal
45	Shekel 7 Problem (S7)	4	Multimodal
46	Shekel 10 Problem (S10)	4	Multimodal
47	Shekel's Foxholes Problem (FX)	5, 10	Multimodal
48	Sinusoidal Problem (SIN)	10, 20	Multimodal
49	Storn's Tchebychev Problem (ST)	9, 17	Multimodal
50	Wood's Problem (WP)	4	Multimodal

Table 5
Computational results for the benchmark set.

Function	Range	D	$f(x^*)$	Results	BPA	IBPA	LADA	TS	SA	WSA
1-ACK	[−30,30]	10	0	Best	0.00411	0.00815	8.822E-4	0.14185	0.01261	**0.888E−15**
				Mean	0.02843	0.02260	0.00473	0.38528	0.05488	**0.888E−15**
				StdDev	0.01338	0.01021	0.00157	0.07488	0.05456	1.0029E−31
				AvgTime	112	102	68	108	115	**33**
2-AP	[−10,10]	2	≈−0.3523	Best	−0.35238	−0.35238	−0.35238	−0.35238	−0.35238	**−0.35238**
				Mean	−0.35238	−0.35238	**−0.35238**	−0.35228	−0.35213	−0.35236
				StdDev	1.449E−6	1.067E−6	5.576E−7	2.183E−5	6.654E−5	8.761E−6
				AvgTime	71	65	36	66	75	**20**
3-BL	[−10,10]	2	0	Best	1.017E−8	3.217E−9	**1.259E−9**	3.955E−7	1.374E−6	5.589E−8
				Mean	2.697E−7	2.826E−7	2.486E−7	7.637E−6	2.772E−5	**1.267E−7**
				StdDev	2.323E−7	2.838E−7	2.704E−7	6.302E−6	2.970E−5	3.877E−8
				AvgTime	55	47	22	48	57	**17**
4-BF1	[−50,50]	2	0	Best	3.656E−5	2.362E−7	6.628E−6	1.123E−4	2.592E−5	**0.00**
				Mean	0.00179	6.419E−4	5.250E−4	0.02104	0.00276	**0.00**
				StdDev	0.00159	7.611E−4	0.00122	0.01801	0.00306	0.00
				AvgTime	62	55	29	57	70	**24**
5-BF2	[−50,50]	2	0	Best	2.927E−6	3.397E−6	4.911E−5	6.223E−4	1.589E−4	**0.00**
				Mean	0.00103	0.00786	0.00324	0.02354	0.00396	**0.00**
				StdDev	0.00144	0.03976	0.00643	0.02656	0.00667	0.00
				AvgTime	64	57	29	58	71	**24**
6-BP	$-5 \le x_1 \le 10$ $0 \le x_2 \le 15$	2	$\frac{5}{4\pi}$	Best	0.39788	0.39788	0.39788	0.39788	0.39788	**0.39788**
				Mean	0.39788	**0.39788**	0.39791	0.39792	0.39796	0.39790
				StdDev	2.361E−6	2.156E−6	9.373E−5	3.543E−5	8.443E−5	9.475E−6
				AvgTime	60	53	24	53	63	**18**
7-CB3	[−5,5]	2	0	Best	6.035E−10	5.405E−9	7.522E−9	1.385E−8	1.174E−6	**0.00**
				Mean	4.285E−7	3.199E−7	8.626E−7	1.503E−5	3.471E−5	**0.00**
				StdDev	6.255E−7	3.123E−7	1.752E−6	1.437E−5	3.458E−5	0.00
				AvgTime	94	86	58	86	96	**25**
8-CB6	[−5,5]	2	≈−1.0316	Best	**−1.03162**	−1.03162	−1.03162	−1.03162	−1.03162	−1.03161
				Mean	−1.03162	−1.03162	**−1.03162**	−1.03160	−1.03155	−1.03161
				StdDev	2.255E−6	2.221E−6	1.747E−6	2.060E−5	5.615E−5	4.516E−16
				AvgTime	118	111	82	112	122	**19**
9-CM	[−1,1]	2	0.2	Best	0.19999	0.19999	0.19999	0.19999	0.19999	**0.2**
				Mean	0.19999	0.19999	0.19999	0.19998	0.19984	**0.2**
				StdDev	2.647E−7	1.646E−7	6.431E−8	4.271E−6	3.309E−5	8.469E−17
				AvgTime	71	65	33	59	78	**25**
		4	0.4	Best	0.39999	0.39999	0.39987	0.39994	0.39991	**0.4**
				Mean	0.39999	0.39999	0.39873	0.39937	0.39855	**0.4**
				StdDev	4.354E−6	4.951E−6	6.909E−4	3.604E−4	7.050E−4	1.693E−16
				AvgTime	89	82	59	78	97	**26**
10-DA	[−20,20]	2	−24,777	Best	−24,776.51	−24,776.51	−24,776.51	−24,776.51	−24,776.51	**−24,776.52**
				Mean	−24,776.47	−24,776.47	−24,776.39	−24,776.24	−24,776.46	**−24,776.49**
				StdDev	0.05370	0.05579	0.25194	0.27919	0.09251	0.01570
				AvgTime	74	67	39	67	83	**26**
11-EP	[−10,10]	2	−1	Best	−0.99999	−0.99999	−0.99999	−0.99999	−0.99999	**−0.99999**
				Mean	**−0.99999**	−0.83334	−0.99999	−0.46667	−0.99991	−0.99957
				StdDev	2.003E−6	0.37901	2.885E−6	0.50733	1.011E−4	2.025E−4
				AvgTime	73	66	37	66	73	**27**
12-EM	[0,π]	10	≈−9.660152	Best	**−9.55986**	−9.52608	−9.27356	−8.97041	−9.52701	−7.20847
				Mean	−9.19363	**−9.28089**	−8.59255	−8.72569	−9.14595	−6.74067
				StdDev	0.21804	0.16721	0.26812	0.13066	0.22567	7.65571
				AvgTime	394	379	336	390	399	**54**
13-EXP	[−1,1]	10	1	Best	0.99998	0.99998	0.99986	0.999936	0.99207	**1.00**
				Mean	0.99994	0.99995	0.99968	0.99752	0.98487	**1.00**
				StdDev	2.469E−5	1.634E−5	1.123E−4	7.975E−4	0.00446	0.00
				AvgTime	97	89	39	87	103	**30**
14-GP	[−2,2]	2	3	Best	3.00000	3.00000	3.00000	3.00000	3.00000	**3.00000**
				Mean	**3.00002**	5.70001	10.0071	3.00053	3.00049	3.00032
				StdDev	2.586E−5	8.23847	16.46670	5.751E−4	0.00114	1.622E−4
				AvgTime	51	45	**16**	45	58	24
15-GW	[−600,600]	10	0	Best	0.29271	0.31242	0.85297	1.25188	0.32537	**0.00**
				Mean	0.67147	0.59742	1.03064	1.48907	0.76516	**0.00**
				StdDev	0.20485	0.20822	0.06245	0.13153	0.26608	0.00
				AvgTime	147	136	90	139	150	**32**
16-GRP	$0.1 \le x_1 \le 100$ $0 \le x_2 \le 25.6$ $0 \le x_3 \le 5$	3	0	Best	**1.208E−6**	5.399E−6	8.124E−5	3.120E−5	2.352E−5	3.283E+1
				Mean	0.00129	0.00157	5.362E−4	**2.047E−4**	2.868E−4	3.283E+1
				StdDev	0.00130	0.00162	3.456E−4	1.382E−4	2.222E−4	1.445E−15
				AvgTime	5234	5211	5205	5207	5269	**121**

Table 5 (*Continued*)

Function	Range	D	$f(x^*)$	Results	BPA	IBPA	LADA	TS	SA	WSA
17-H3	[0,1]	3	≈−3.862782	Best	−3.86278	**−3.86278**	−3.86278	−3.86278	−3.86276	−3.86003
				Mean	−3.86278	−3.86278	**−3.86278**	−3.86277	−3.86230	−3.85943
				StdDev	8.459E−7	3.136E−7	3.591E−7	8.091E−6	1.135E−4	3.204E−4
				AvgTime	117	110	82	110	119	**25**
18-H6	[0,1]	6	≈−3.322368	Best	−3.32236	**−3.32236**	−3.32207	−3.32178	−3.31891	−3.30395
				Mean	−3.27069	−3.25877	**−3.31719**	−3.28901	−3.25266	−3.16019
				StdDev	0.06007	0.06048	0.02160	0.05278	0.04760	0.08199
				AvgTime	165	158	131	159	168	**40**
19-HV	[−10,10]	3	0	Best	1.3243E−5	9.542E−6	2.603E−6	1.010E−4	1.415E−4	**1.7993E−8**
				Mean	2.285E−4	2.003E−4	6.894E−5	0.00674	0.00237	**2.0605E−6**
				StdDev	2.274E−4	1.676E−4	6.454E−5	0.00538	0.00164	8.903E−7
				AvgTime	61	54	**26**	54	67	40
20-HSK	$0 \le x_1 \le 5$ $0 \le x_2 \le 6$	2	≈−2.3458	Best	−2.34581	−2.34581	−2.34581	−2.34581	−2.34581	**−2.34581**
				Mean	−2.34581	−2.34581	**−2.34581**	−2.34579	−2.34573	−2.34558
				StdDev	8.203E−8	1.356E−7	5.327E−8	3.169E−6	1.994E−5	1.631E−4
				AvgTime	57	50	**22**	50	58	27
21-KL	[0,0.42]	4	≈3.0748E−4	Best	3.075E−4	3.075E−4	3.105E−4	**3.075E−4**	3.087E−4	3.079E−4
				Mean	3.105E−4	3.118E−4	3.684E−4	**3.083E−4**	3.204E−4	3.111E−4
				StdDev	3.488E−6	4.096E−6	6.400E−5	6.638E−7	6.842E−6	1.706E−6
				AvgTime	89	84	54	83	91	**57**
22-LM1	[−10,10]	3	0	Best	1.005E−7	1.095E−7	**8.240E−8**	3.124E−6	3.515E−5	3.315E−7
				Mean	7.457E−6	4.979E−6	**2.322E−6**	1.436E−4	3.392E−4	6.705E−6
				StdDev	6.041E−6	6.716E−6	2.867E−6	1.186E−4	2.257E−4	3.392E−6
				AvgTime	84	76	48	76	86	**37**
23-LM2	[−5,5]	5	0	Best	0.00138	0.01882	0.18022	0.17907	0.00706	**0.00103**
				Mean	0.30253	0.55175	2.62975	0.57022	0.27072	**0.02390**
				StdDev	0.27903	0.48275	0.67992	0.32601	0.25005	0.01237
				AvgTime	64	58	36	58	72	**29**
		10	0	Best	0.00387	0.00962	0.06966	0.05080	0.01122	**0.00257**
				Mean	0.35532	0.47384	0.59860	0.45002	0.35846	**0.08629**
				StdDev	0.40486	0.41149	0.41381	0.24229	0.28039	0.05220
				AvgTime	65	58	36	58	72	**33**
24-MC	$-1.5 \le x_1 \le 4$ $-3 \le x_2 \le 3$	2	≈−1.9133	Best	−1.91322	−1.91322	−1.91322	−1.91322	−1.91322	**−1.91327**
				Mean	−1.91322	−1.91322	−1.91321	−1.18018	−1.91320	**−1.91322**
				StdDev	1.898E−7	1.435E−7	1.359E−6	1.35145	1.995E−5	2.738E−5
				AvgTime	59	52	23	52	60	**18**
25-MR	[−10,10]	3	≈0.4E−4	Best	4.356E−5	**4.358E−5**	4.426E−5	4.357E−5	4.574E−5	4.409E−5
				Mean	6.049E−5	6.084E−5	6.496E−5	4.832E−5	5.903E−5	**4.773E−5**
				StdDev	3.054E−5	2.270E−5	2.264E−5	7.299E−6	1.027E−5	2.271E−6
				AvgTime	68	61	34	61	70	**25**
26-MCP	[−1,1]	4	0	Best	6.684E−12	5.606E−12	5.260E−8	3.376E−11	1.098E−9	**8.934E−13**
				Mean	9.185E−9	1.120E−8	2.627E−6	1.353E−9	4.094E−7	**2.414E−10**
				StdDev	8.576E−9	1.235E−8	3.129E−6	2.167E−9	5.113E−7	1.753E−10
				AvgTime	147	140	127	140	152	**79**
27-ML	[0,10]	10	≈−0.965	Best	−0.96021	**−0.96179**	−0.86277	−0.77328	−1.77E−5	−0.95892
				Mean	−0.62031	−0.46008	−0.24771	−0.43739	−6.48E−7	**−0.70597**
				StdDev	0.39698	0.37028	0.29391	0.23061	3.234E−6	0.14413
				AvgTime	178	170	120	171	182	**64**
28-MRP	[−5,5]	2	0	Best	1.960E−6	5.865E−7	3.210E−8	1.623E−5	1.732E−5	**4.291E−9**
				Mean	8.632E−4	0.00211	0.00147	7.907E−4	0.00105	**6.812E−7**
				StdDev	0.00237	0.00403	0.00218	7.948E−4	1.037E−4	3.871E−7
				AvgTime	59	52	23	52	64	**17**
29-MGP	[−2,2]	2	≈1.29695	Best	−1.29695	−1.29695	−1.29695	−1.29694	−1.29694	**1.29695**
				Mean	**−1.29695**	−1.27557	−1.29160	−1.29681	−1.29665	1.29666
				StdDev	4.888E−6	0.03605	0.02033	1.286E−4	3.470E−4	1.778E−4
				AvgTime	121	114	86	114	121	**35**
30-NF2	[0,D]	4	0	Best	4.047E−4	0.00156	0.00480	8.216E−4	0.00339	**3.646E−4**
				Mean	0.04493	0.05163	0.02506	0.02137	0.01717	**0.01200**
				StdDev	0.03509	0.05279	0.01496	0.01166	0.01012	0.00708
				AvgTime	243	236	215	237	250	**87**
31-NF3	[$-D^2$,D^2]	10	−210	Best	−209.161	−209.133	−206.148	−180.266	**−209.910**	−174.025
				Mean	−181.816	−191.591	−150.027	−154.833	**−208.398**	−118.754
				StdDev	37.4011	19.5298	49.3904	15.6459	1.96285	29.4082
				AvgTime	108	101	44	104	117	**23**
32-OSP	[−15,15]	20	≈−1.143833	Best	−0.01385	−0.01007	−0.01001	−0.01780	−0.00494	**−0.17028**
				Mean	0.00454	−0.00199	−0.00152	−0.00615	−0.00214	**−0.14246**
				StdDev	0.00375	0.00277	0.00189	0.00393	9.081E−4	0.01729
				AvgTime	174	165	67	169	177	**49**

Table 5 (*Continued*)

Function	Range	D	$f(x^*)$	Results	BPA	IBPA	LADA	TS	SA	WSA
33-PP	[2,10]	10	≈−45.778	Best	−45.7761	**−45.7769**	−45.7626	−45.6833	−45.7642	−30.7878
				Mean	−45.7724	**−45.7741**	−45.7353	−45.5117	−45.7403	−27.3848
				StdDev	0.00226	0.00152	0.01270	0.06388	0.01351	2.03634
				AvgTime	190	182	136	189	196	**55**
34-PRD	[−10,10]	2	0.9	Best	0.90000	0.90000	0.90000	0.90000	0.90001	**0.90000**
				Mean	0.90000	0.90000	0.90000	0.90003	0.90017	**0.90000**
				StdDev	2.132E−6	1.632E−6	6.619E−7	3.217E−5	1.845E−4	4.516E−16
				AvgTime	72	65	36	66	74	**21**
35-PWQ	[−10,10]	4	0	Best	1.651E−5	2.294E−7	3.530E−4	0.00132	1.113E−4	**0.00**
				Mean	0.00136	0.00102	0.03571	0.01224	0.00356	**0.00**
				StdDev	0.00133	0.00136	0.03693	0.00897	0.00330	0.00
				AvgTime	97	91	69	92	106	**29**
36-PTM	[−10,10]	9	0	Best	4.85590	2.40392	6.72115	2.01215	**1.25577**	10.7313
				Mean	54.9967	67.3443	351.792	**27.7876**	45.5377	31.4807
				StdDev	33.8398	46.2403	693.422	40.9989	132.032	14.4199
				AvgTime	141	134	91	135	150	**45**
37-RG	[−5.12,5.12]	10	0	Best	0.16961	0.08790	0.00606	4.58753	0.03932	**0.00**
				Mean	0.43289	0.29275	0.01584	6.35541	0.15916	**0.00**
				StdDev	0.16469	0.12481	0.00554	0.89405	0.09522	0.00
				AvgTime	133	125	61	134	141	**32**
38-RB	[−30,30]	10	0	Best	3.30050	1.65780	13.1161	24.7395	**0.91868**	8.91674
				Mean	13.5681	12.1420	26.4740	66.1024	**6.41087**	8.94485
				StdDev	17.9707	14.9202	14.9521	19.1763	1.81797	0.016029
				AvgTime	89	83	34	83	98	**25**
39-SAL	[−100,100]	5	0	Best	0.09987	0.09987	0.09987	0.09996	0.09987	**0.00**
				Mean	0.20329	0.18655	0.14497	0.30807	0.20668	**0.00**
				StdDev	0.09609	0.07301	0.04966	0.08101	0.09040	0.00
				AvgTime	67	60	26	60	69	**20**
		10	0	Best	0.29988	0.29987	0.30989	1.03152	0.30006	**0.00**
				Mean	0.45003	0.47596	0.57499	1.39242	0.59059	**0.00**
				StdDev	0.08947	0.12510	0.08872	0.16112	0.15625	0.00
				AvgTime	91	84	40	86	97	**24**
40-SF1	[−100,100]	2	0	Best	1.2051E−4	1.979E−5	1.535E−6	1.409E−4	6.671E−4	**0.00**
				Mean	0.00749	0.00536	0.00557	0.00621	0.00840	**0.00**
				StdDev	0.00362	0.00474	0.00444	0.00343	0.00277	0.00
				AvgTime	65	62	28	56	63	**25**
41-SF2	[−100,100]	2	0	Best	0.06332	0.08329	0.21116	0.58920	0.06096	**8.937E−77**
				Mean	0.16367	3.21774	3.24847	3.63356	0.30990	**1.201E−76**
				StdDev	0.05498	1.40596	1.33647	0.77425	0.35340	1.343E−77
				AvgTime	102	103	67	95	104	**23**
42-SBT	[−10,10]	2	≈−186.7309	Best	−186.730	−186.730	−186.730	−186.730	−186.730	**−186.730**
				Mean	−186.730	−186.730	**−186.730**	−186.723	−186.729	−186.727
				StdDev	3.828E−4	2.462E−4	2.114E−4	0.00742	0.00128	0.00175
				AvgTime	105	98	69	97	111	**27**
43-SWF	[−500,500]	10	≈−418.9829D	Best	−4188.83	−4188.80	**−4189.79**	−4146.86	−4189.64	−4080.13
				Mean	−4186.57	−4187.48	**−4189.74**	−4099.80	−4189.39	−3505.52
				StdDev	1.41713	0.90048	0.03670	23.2457	0.23796	3.282E+2
				AvgTime	137	129	71	134	145	**27**
44-S5	[0,10]	4	≈−10.1531	Best	**−10.1530**	−10.1525	−10.1530	−10.1475	−10.1516	−10.1307
				Mean	−7.89264	−5.39122	−9.41937	−5.43991	−8.80082	**−9.86326**
				StdDev	3.10827	3.09944	1.92103	2.86814	2.26671	0.17026
				AvgTime	112	115	74	105	114	**33**
45-S7	[0,10]	4	≈−10.4028	Best	**−10.4028**	−10.4027	−10.4027	−10.3701	−10.4022	−10.3277
				Mean	−9.15761	−7.32823	−9.08027	−8.02058	**−9.82482**	−8.60123
				StdDev	2.54598	3.42810	2.70629	3.01025	1.75095	1.18516
				AvgTime	135	138	97	127	138	**35**
46-S10	[0,10]	4	≈−10.5362	Best	−10.5363	−10.5363	**−10.5363**	−10.4661	−10.5361	−10.4698
				Mean	−9.01572	−8.16891	−9.25772	−7.26594	**−9.73094**	−9.29081
				StdDev	2.80903	3.22723	2.60790	3.22036	1.84226	0.66730
				AvgTime	170	173	131	163	172	**41**
47-FX	[0,10]	5	≈−10.4056	Best	**−10.4044**	−10.4013	−10.2586	−10.1727	−10.4014	−10.0635
				Mean	−4.15423	−3.00151	−3.97853	−4.16442	−6.42803	**−8.08892**
				StdDev	3.18847	2.05819	2.51266	3.14428	3.62026	1.12418
				AvgTime	170	161	134	162	171	**43**
		10	≈−10.2088	Best	**−9.89768**	−2.23178	−1.93111	−2.51885	−9.65529	−8.73205
				Mean	−1.75189	−1.59930	−1.46436	−1.48733	−2.47327	**−3.14950**
				StdDev	1.53866	0.25277	0.08844	0.33668	2.24299	3.37647
				AvgTime	224	215	167	218	225	**54**

Table 5 (*Continued*)

Function	Range	*D*	$f(x^*)$	Results	BPA	IBPA	LADA	TS	SA	WSA
48-SIN	[0,180]	10	−3.5	Best	−3.49459	**−3.49671**	−3.43730	−3.27354	−3.48933	−3.49129
				Mean	−3.48543	**−3.49018**	−3.30715	−3.00020	−3.45038	−3.46650
				StdDev	0.00460	0.00425	0.08038	0.12374	0.02185	0.01363
				AvgTime	187	178	134	181	193	**34**
		20	−3.5	Best	−2.98125	−3.05950	−2.10485	−1.86913	−2.45027	**−3.07414**
				Mean	−2.67929	−2.50682	−1.78625	−1.35879	−1.29606	**−2.98542**
				StdDev	0.33983	0.78771	0.21988	0.45664	0.97984	0.04242
				AvgTime	351	338	274	344	356	**40**
49-ST	[−128,128]	9	0	Best	**19.0905**	30.2389	72.3632	71.1200	24.6402	45.8053
				Mean	182.298	165.297	476.741	238.236	105.067	**94.8146**
				StdDev	207.122	120.048	397.055	78.9182	63.7722	22.4577
				AvgTime	8073	8249	8134	8011	8200	**300**
	[−32768,32768]	17	0	Best	6,482,485	3,965,373	1.4447E+7	1.2599E+8	1.2976E+7	**1,119,666**
				Mean	4.3309E+7	3.7505E+7	1.6405E+8	1.8776E+9	5.9167E+7	**1,261,969**
				StdDev	2.7479E+7	2.7370E+7	1.6674E+8	1.6880E+9	4.4379E+7	8.4015E+4
				AvgTime	29,832	30,478	30,346	29,694	30,418	**908**
50-WF	[−10,10]	4	0	Best	0.01755	0.00163	0.19640	0.11683	0.01851	**5.510E−4**
				Mean	0.28819	0.38212	0.75840	0.51335	0.36022	**0.18005**
				StdDev	0.33533	0.36681	0.34910	0.25108	0.35557	0.10121
				AvgTime	65	69	36	58	73	**26**

$f(x^*)$ refers optimum objective value of the functions. The bold values are the best solutions found by the algorithms.

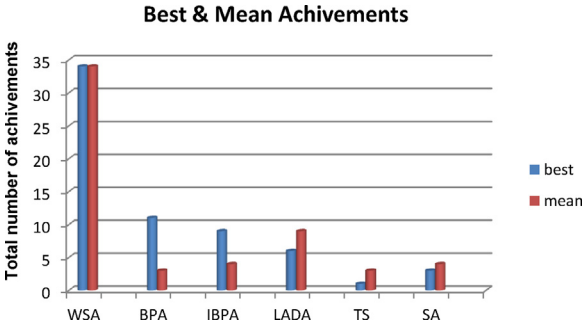

Fig. 9. Visually comparison of the 6 algorithms.

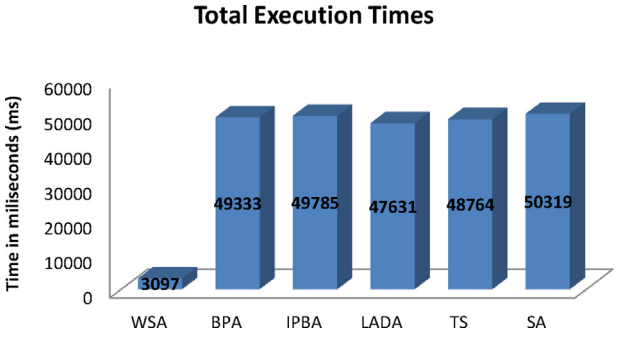

Fig. 10. Total execution times for per algorithm over 56 functions.

4. Computational experience

In this section performance evaluation of the WSA algorithm is carried out into three parts. The first part contains a basic benchmark set that consist of 50 unconstrained continuous global optimization problems, originally compiled by Ali et al. [78] and also used by Chetty and Adewumi [1]. In the second part, we compared the WSA algorithm with several versions of the PSO algorithm, which is a very popular search procedure in the field of continuous optimization. For this purpose, we have based the second part of the computational study on Qin et al.'s work [79], which provides a performance evaluation test over many variants of the PSO algorithm by using some benchmark functions with larger size in comparison to the benchmark set used in the first part of the computational study. Both benchmark sets include unimodal and multimodal functions. A unimodal function has only one single optimum point, while a multimodal has more than one. On the other hand, premature convergence, which may cause the algorithm to trap in local optima or to stagnate, is a challenging problem of the meta-heuristic approaches if an efficient balance between diversification and intensification is not provided. Therefore, multimodal functions provide more consistent information than unimodal functions about the performance of a search algorithm. Finally in the third part, we carried out a performance evaluation test of WSA over some selected complex problems from a recent benchmark set, CEC 2013, [80].

4.1. Parameter calibration of WSA

This section is about to select the appropriate parameter values of WSA algorithm. For this purpose we used the Taguchi method [81] due to its fewer experiments requirement in comparison to full factorial design. Taguchi method is able to analyse the effects of a large number of decision variables via a small number of experiments thanks to the orthogonal arrays and it divides the factors into two groups as controllable and noise factors (uncontrollable). Taguchi method aims to determine the optimal levels of controllable factors by trying to minimize the effect of uncontrollable factors. Additionally, Taguchi discovers the relative significance of each factor in terms of their main effects on the objective function, and the aim is to maximize the signal-to-noise ratio. Taguchi evaluates the objective functions into three categories as the smaller-the-better, the larger-the-better, and the nominal-is-the-best. Since the objective functions of the used benchmark sets are the smaller-the-better type, its corresponding S/N ratio is: S/N ratio $= -10\log_{10}(objective)^2$. The control factors of WSA algorithm are maximum number of iterations (*Maxiter*), number of artificial agents at each iteration (*AA*), three used defined parameters (τ, λ, and φ) and initial step length (sl_0). Since *Maxiter* and *AA* are the parameter values should be set as previously used values with the aim of carrying out fairly comparisons with the other approaches. Therefore, WSA has 4 parameters required to properly tune before executing the algorithm and their considered levels, determined through a set of preliminary experiments, are reported in Table 1.

Table 6
Summary of the computational results.

No.	WSA		BPA		IBPA		LADA		TS		SA	
	Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean
1	ACK	ACK	AP	EP	BP	BP	BL	AP	KL	GRP	NF3	NF3
2	AP	BL	CB6	GP	H3	EM	GP	CB6		KL	PTM	RB
3	BF1	BF1	EP	MGP	H6	PP	LM1	H3		PTM	RB	S7
4	BF2	BF2	EM		HSK	SIN(10)	SBT	H6				S10
5	BP	CB3	GRP		MR		SWF	HSK				
6	CB3	CM(2)	MGP		ML		S10	LM1				
7	CM(2)	CM(4)	S5		PP			PRD				
8	CM(4)	DA	S7		PRD			SBT				
9	DA	EXP	FX(5)		SIN(10)			SWF				
10	EP	GW	FX(10)									
11	EXP	HV	ST(9)									
12	GP	LM2(5)										
13	GW	LM2(10)										
14	HV	MC										
15	HSK	MR										
16	LM2(5)	MCP										
17	LM2(10)	ML										
18	MC	MRP										
19	MCP	NF2										
20	MRP	OSP										
21	MGP	PRD										
22	NF2	PWQ										
23	OSP	RG										
24	PRD	SAL(5)										
25	PWQ	SAL(10)										
26	RG	SF1										
27	SAL(5)	SF2										
28	SAL(10)	S5										
29	SF1	FX(5)										
30	SF2	FX(10)										
31	SBT	SIN(20)										
32	SIN(20)	ST(9)										
33	ST(17)	ST(17)										
34	WF	WF										

A value of τ near to one gives the best performance. In our computational work we did not realize any considerable difference for the values of τ between 0.7 and 1.2, hence we fix upper level of τ as 0.8 for all tests. In another meta-heuristic algorithm which is known as extremal optimization [82–84] a similar parameter was also used with a similar setting with high success. This also attracted our attention in setting up this parameter. After an experimental selection, the lower level of the parameter sl_0 was chosen as 0.035. The centre point for the user defined parameter φ was set to as 0.001 [71,72] and the lower level of the user defined parameter λ was set to as 0.75 [72].

As it can be seen in Table 1, WSA requires tuning 4 parameters each one has three levels. Thus, the L27 orthogonal array, which fulfils all our requirements, can be selected. The Shekel's Foxholes Problem (FX) with 5 dimensions was chosen to evaluate the effects of the parameters of WSA. For each combination, 10 independent runs are conducted to determine the variation in the results and the relative percentage deviation (RPD) was used as the performance measure [85].

To discover the relative significance of each parameter in terms of their main effects on the objective function, an analysis of variance (ANOVA) was conducted for the selected problem. The Taguchi method has a main aim of robustness and it secondly tries to select the optimum levels of parameters. That is to say, if there is no significant different between the S/N ratios, the RPD values should be considered as the second criteria [86]. Tables 2 and 3 indicate the ANOVA results for S/N ratios and RPD values obtained at each level of parameters.

According to the S/N ratios it is obvious that all 4 parameters have a meaningful effect at the 5% significance level over the objective value. Therefore, the parameter levels of $\tau(3)^*$, $sl_0(1)^*$, $\varphi(2)^*$ and $\lambda(1)^*$ were selected on the basis of both S/N ratios and RPD values given in the second and third columns of Table 3, respectively. In other words, the parameters τ, sl_0, φ and λ were set to 0.8, 0.035, 0.001 and 0.75, respectively.

Table 7
Test problems for the second part of the performance evaluation of WSA.

	Name of the problem	Dimension	Type
51	Schwefel's Function P1.2	30	Unimodal
52	Rosenbrock Function	30	Unimodal
53	Schwefel's Function P2.21	30	Unimodal
54	Schwefel's Function P2.22	30	Unimodal
55	Zakharov Function	30	Unimodal
56	Elliptic Function	30	Unimodal
57	Rastrigin Function	30	Multimodal
58	Ackley Function	30	Multimodal
59	Griewank Function	30	Multimodal
60	Schwefel Function	30	Multimodal
61	Noncontinuous Rastrigin Function	30	Multimodal
62	Weierstrass Function	30	Multimodal
63	Levy Function	30	Multimodal

4.2. Computational study-1

Table 4 presents some characteristics of the benchmark set used for the first part of the performance evaluation test. The reader can refer to [78] or [1] for the mathematical formulations of all of the functions and the parameter values of Hartman 3, Hartman 6, Kowalik, Meyer and Roth, Modified Langerman, Multi-Gaussian, Shubert, Shekel 5, Shekel 7, Shekel 10, Shekel's Foxholes functions.

Table 8
Comparisons of WSA with PSO algorithms (*for 400, 000 function evaluations*).

Function	Mean solution accuracy									
	PSO-w	PSO-cf	FIPS	FDR-PSO	HPSO-TVAC	DMS-PSO	GPSO	CLPSO	PSOPB	WSA
f_{51}	5.56E−002	1.28E−020	6.67E+000	1.15E−006	9.31E−009	1.19E+001	1.49E−009	3.51E+002	4.64E−025	**0.00**
f_{52}	3.43E+001	1.68E+000	2.27E+001	5.99E+000	4.74E−001	3.49+001	3.05E+001	6.81E+000	**1.44E−003**	2.41E+01
f_{53}	1.75E−001	8.57E−013	5.49E−005	2.72E−003	5.65E−006	3.19E−001	2.21E−006	2.80E+000	4.77E−015	**0.00**
f_{54}	2.13E−030	1.90E−039	4.08E−015	1.30E−019	7.95E−029	6.59E−022	2.01E+002	3.21E−016	4.23E−122	**0.00**
f_{55}	2.94E+002	1.49E−026	2.91E+002	1.04E−004	2.20E−015	2.27E+002	7.03E−010	2.06E+003	3.21E−025	**0.00**
f_{56}	1.42E−062	1.23E−184	9.17E−025	5.56E−060	1.38E−043	9.21E−043	1.87E+006	2.18E−024	7.43E−208	**0.00**
f_{57}	2.08E+001	5.73E+001	4.87E+001	1.85E+001	5.02E+000	2.63E+001	7.96E+001	**0.00**	**0.00**	**0.00**
f_{58}	6.82E−015	1.58E+000	7.99E−015	6.57E−015	4.19E−014	1.97E−014	2.98E−009	2.89E−014	8.71E−015	**7.40E−16**
f_{59}	1.92E−002	1.47E−022	3.28E−010	1.89E−002	6.33E−003	1.61E−002	1.34E−002	**0**	4.62E−003	**0.00**
f_{60}	1.17E+003	3.75E+003	1.41E+002	1.11+003	7.51E+002	8.50E+002	1.33E+003	**3.82E−004**	**3.82E-004**	5.85E+03
f_{61}	7.48E+000	4.73E+001	5.82E+001	2.29E+001	3.35E+000	2.91E+001	7.20E+001	**0.00**	**0.00**	**0.00**
f_{62}	1.90E−005	4.53E+000	1.82E−002	6.63E−003	2.03E−014	9.41E−007	3.68E+000	**0.00**	**0.00**	**0.00**
f_{63}	2.65E−002	3.47E+000	1.01E−027	2.51E−031	4.41E−018	5.88E−002	6.96E+001	1.03E−026	**1.35E−031**	2.40E+01

The best results are marked in boldface.

Table 9
Comparisons of WSA with the state of the art PSO algorithms (*for 20, 000 function evaluations*).

Function	Mean solution accuracy								
	ALC-PSO	OPSO	FPSO	AIWPSO	APSO	OLPSO-G	OLPSO-L	PSOPB	WSA
f_{51}	1.79E−011	NA	1.73E+002	1.96E−010	1.0E−010	NA	NA	2.36E−012	**1.54E−253**
f_{52}	7.61	49.61	28.16	2.50	2.84	21.52	1.26	**0.58**	28.95
f_{54}	116E−090	1.26E−010	1.58E−011	1.65E−062	5.15E−084	9.85E−030	7.67E−022	3.82E−057	**4.66E−127**
f_{57}	2.53E−014	6.97	7.38E+001	1.66E−001	5.80E−015	1.07	**0.00**	**0.00**	**0.00**
f_{58}	1.15E−014	6.23E−009	2.18E−009	6.99E−015	1.11E−014	7.98E−015	4.14E−015	9.41E−015	**7.40E−016**
f_{59}	1.22E−002	2.29E−003	1.47E−003	2.85E−002	1.67E−002	4.83E−003	**0.00**	8.32E−003	**0.00**
f_{60}	21.03	2.93E+003	1.35E+003	**3.82E−004**	**3.82E−004**	3.84E+002	**3.82E−004**	**3.82E−004**	6.04E+003
f_{61}	1.25E−011	NA	7.03+001	11.18E−016	4.14E−016	NA	NA	**0.00**	**0.00**
f_{63}	NA	NA	5.51E−018	**1.50E−032**	NA	NA	NA	1.36E−031	2.87E+001

The best results are marked in boldface.

4.2.1. Results and discussion for computational study-1

We have been conducted a direct comparison between the obtained results of WSA and the results provided by Chetty and Adewumi for the algorithms of best performance algorithm (BPA), iterative best performance algorithm (IBPA), largest absolute difference algorithm (LADA), Tabu search (TS) and simulated annealing (SA) on the benchmark functions listed in Table 4. WSA was coded in Matlab 7.9, and then the results were obtained by running the coded WSA on a Core(TM) i7-2640 CPU 2.80 GHz personal computer. To compare the algorithms fairly, the initial values of *Maxiter* and *AA* were set to 5000 and 20 so as to perform 100,000 objective function evaluations for every run as Chetty and Adewumi did.

The obtained results of WSA and the provided results of the other 5 algorithms on the 50 benchmark functions are reported in Table 5. Since, Chetty and Adewumi had run per algorithm for per function 30 times, we also ran WSA 30 times for all the test functions and the best and mean values of the solutions have been documented. Additionally, the mean execution times (AvgTime) for all algorithms have also been given in Table 5. The best values found by the algorithms have been emphasized in bolt font. The reader should also be noticed that the problems of CM, LM2, SAL, FX, SIN and ST have been evaluated for two different dimension settings. Thus, WSA evaluated 56 functions and therefore executed totally 1680 function evaluations, since we ran WSA 30 times for each test function.

From the observation of Table 5, WSA produced the best "Mean" and "Best" values for the 34 functions out of 56 functions and this refers to 60.7% success level. Additionally, WSA has the lowest "AvgTimes" for 53 functions out of 56 and this refers to 94.6% success level. On the other hand, WSA was not able to produce the best "Best" values for 22 functions, however 17 of them has reasonable levels. For the other 5 functions (16-GRP, 31-NF3, 33-PP, 36-PTM and 38-RB) WSA was not able to produce good enough results and

Table 10
Test problems for the third part of the performance evaluation of WSA.

	Name of the problem	Type	n
64	Composition Function 1	Rotated	5
65	Composition Function 2	Unrotated	3
66	Composition Function 3	Rotated	3
67	Composition Function 4	Rotated	3
68	Composition Function 5	Rotated	3
69	Composition Function 6	Rotated	5
70	Composition Function 7	Rotated	5
71	Composition Function 8	Rotated	5

n = Number of different functions used to construct the related composition function.

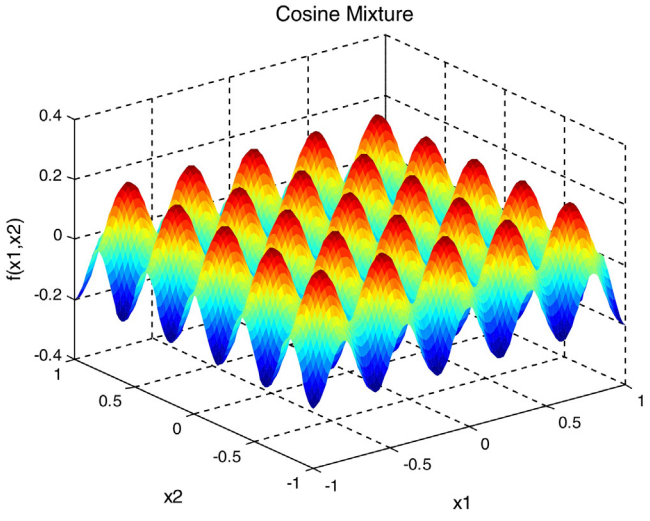

Fig. 11. 3-D plot for the two dimensional Cosine Mixture Function.

Table 11
Computational results for the composition functions.

Function	$f(x^*)$		$D=2$	$D=5$	$D=10$	$D=30$	$D=50$
f_{64}	700	Best	7.081E+02	9.473E+02	13.817E+02	31.641E+02	53.125E+02
		Mean	7.094E+02	9.487E+02	13.851E+02	31.697E+02	53.218E+02
		Worst	7.101E+02	9.507E+02	13.901E+02	31.759E+02	53.307E+02
		StdDev	0.5684	0.9670	3.0094	3.2095	5.1197
f_{65}	800	Best	8.013E+02	12.182E+02	30.424E+02	96.607E+02	16.519E+03
		Mean	8.022E+02	12.192E+02	30.472E+02	96.660E+02	16.533E+03
		Worst	8.029E+02	12.201E+02	30.518E+02	96.703E+02	16.548E+03
		StdDev	0.5068	0.5995	2.4219	3.1427	7.2564
f_{66}	900	Best	9.113E+02	14.137E+02	29.610E+02	90.671E+02	17.329E+03
		Mean	9.115E+02	14.156E+02	29.641E+02	90.686E+02	17.342E+03
		Worst	9.117E+02	14.177E+02	29.678E+02	90.704E+02	17.352E+03
		StdDev	0.1074	1.0622	2.1312	0.9977	6.6782
f_{67}	1000	Best	10.000E+02	11.100E+02	12.324E+02	13.544E+02	15.162E+02
		Mean	10.000E+02	11.112E+02	12.345E+02	13.566E+02	15.197E+02
		Worst	10.000E+02	11.127E+02	12.371E+02	13.592E+02	15.238E+02
		StdDev	0.0000	0.7899	1.2345	1.3372	2.0877
f_{68}	1100	Best	11.100E+02	12.307E+02	13.215E+02	14.204E+02	15.321E+02
		Mean	11.100E+02	12.323E+02	13.245E+02	14.238E+02	15.247E+02
		Worst	11.100E+02	12.338E+02	13.276E+02	14.271E+02	15.388E+02
		StdDev	0.0000	0.9551	1.9166	1.9806	2.0555
f_{69}	1200	Best	12.000E+02	13.207E+02	14.011E+02	14.604E+02	16.717E+02
		Mean	12.000E+02	13.218E+02	14.033E+02	14.629E+02	16.738E+02
		Worst	12.000E+02	13.232E+02	14.056E+02	14.646E+02	16.759E+02
		StdDev	0.0000	0.6956	1.3613	1.1504	1.2675
f_{70}	1300	Best	13.100E+02	16.921E+02	19.298E+02	28.244E+02	38.548E+02
		Mean	13.100E+02	16.939E+02	19.318E+02	28.264E+02	38.575E+02
		Worst	13.100E+02	16.954E+02	19.337E+02	28.283E+02	38.603E+02
		StdDev	0.0000	0.9581	1.2651	0.9626	1.6281
f_{71}	1400	Best	14.400E+02	16.805E+02	23.788E+02	67.174E+02	11.359E+03
		Mean	14.407E+02	16.822E+02	23.805E+02	67.192E+02	11.367E+03
		Worst	14.416E+02	16.837E+02	23.823E+02	67.206E+02	11.377E+03
		StdDev	0.4433	0.8698	0.9398	1.0242	5.4081

this refers to 8.9% of all the functions. Additionally, from the observations of the standard deviation values provided in Table 5 it is obvious that WSA has the best stability as compared to the other five algorithms. WSA is able to produce results with the smallest standard deviation values among all the algorithms on most test functions. In general, we can conclude that WSA is a highly stable search procedure and is capable of producing reasonable consistent results. Since we compare the results of WSA directly to Chetty and Adewumi's results, we also summarized WSA's results like them as in Table 6.

As indicated in Table 6, it is not clear whether WSA produce better "Best" values then Chetty and Adewumi's results for 8 functions (2-AP, 6-BP, 11-EP, 14-GP, 20-HSK, 29-MGP, 34-PRD, and 42-SBT) or not, since they gave the truncated results for these functions. Therefore, it is possible to say that WSA produced best or close to best "Best" values for these 8 functions considering the truncated results. So, we can conclude that WSA produced the best "Best" values for the 34 functions and this refers to 60.7% success level.

The reported data in Table 6 is also visually compared in Fig. 9, which is formed by simply summing of all the functions under both the "Best" and "Mean" columns in Table 4 for each algorithm as Chetty and Adewumi did. The superiority of WSA over the other 5 algorithms can be clearly and precisely seen in Fig. 9.

Additionally, Fig. 10 visually compares the total "AvgTime" for each algorithm. Fig. 10 also simply sums the "AvgTimes" values of every algorithm over 56 functions. It is also clearly and precisely seen that WSA surpasses all of the other 5 algorithms in case of the execution times over all the benchmark set. So, we can conclude that WSA has the ability of producing satisfactory results in reasonable time limits. On the other hand, the reader should be aware that the algorithms were not compared by using the same machine. So, Fig. 10 should be commented within this direction.

4.3. Performance evaluation of WSA over PSO variants

In the second part of the computational study, it is aimed to indicate the effectiveness of WSA algorithm over the PSO algorithm, which is a popular search strategy in the field of continuous optimization. Within this purpose, we used the benchmark set used by Qin et al. [79], since they also provided a comprehensive performance evaluation test over 16 PSO variants published between the years of 1998 and 2015. The benchmark set used in this part is depicted in Table 7 and the reader can refer to [79] for the mathematical formulations of these problems.

This second part of the computational study involves two subparts. The performance of WSA is compared via 9 PSO variants for 400,000 function evaluations and 8 PSO variants for 20,000 function evaluations for the first and second subparts, respectively. These variants of the PSO algorithm are PSO with inertia weight (PSO-w) [87], PSO with constriction factor (PSO-cf) [88], fitness-distance-ratio based PSO (FDR-PSO) [89], fully informed particle swarm (FIPS) [41], hierarchical PSO with time-varying acceleration coefficients (HPSO-TVAC) [90], Dynamic multi-swarm particle swarm optimizer (DMS-PSO) [42], Gregarious PSO (GPSO) [91], comprehensive learning PSO (CLPSO) [46], orthogonal PSO (OPSO) [92], Frankenstein's PSO (FPSO) [93], adaptive PSO (APSO) [40], PSO with adaptive inertia weight (AIWPSO) [39], orthogonal learning PSO with global star neighbourhood (OLPSO-G) [94], orthogonal learning PSO with local ring neighbourhood (OLPSO-L)

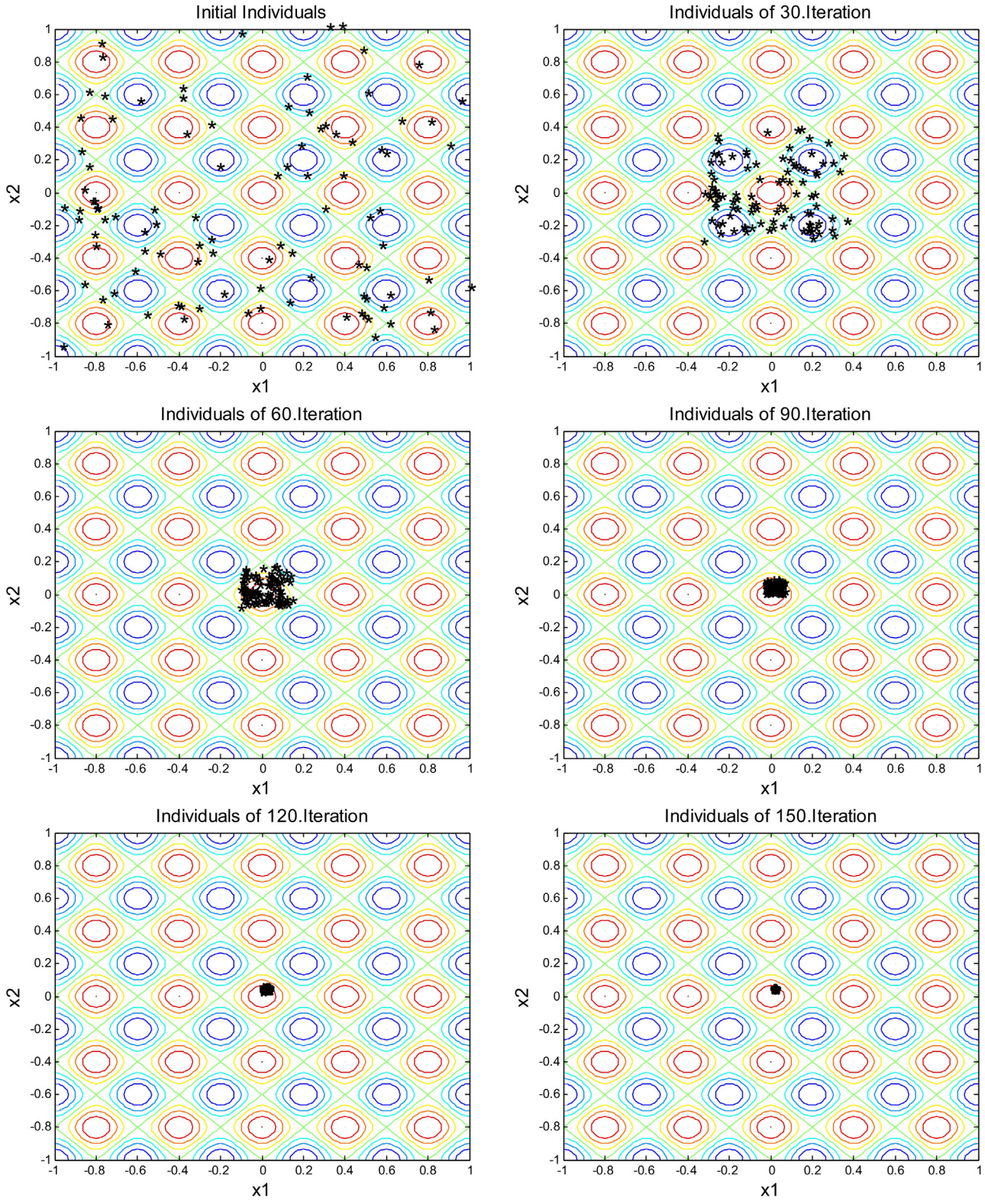

Fig. 12. Convergence behaviour of WSA on Cosine Mixture Function.

[94], PSO with an ageing leader and challenger (ALC-PSO) [95], and coevolutionary particle swarm optimizer with parasitic behaviour (PSOPB) [79]. And obtained result via WSA compared to these algorithms in Tables 8 and 9 for the first and second subparts, respectively. The results given in Tables 8 and 9 are the means of 25 and 30 independent runs, respectively.

The performance evaluation test given in Tables 8 and 9 indicate that WSA algorithm has a superior performance over the different variants of PSO algorithm, since it produced better solutions for 10 problems out of 13 and for 6 problems out of 9 problems as listed in Tables 8 and 9, respectively. At this point we can conclude that WSA prove its effectiveness in solving the unconstrained

Initial top 5 Individuals	
x1	x2
0.0522049	0.0624186
0.370057	0.0612586
0.0675439	-0.368377
0.0990802	0.111476
0.0108563	-0.454122

Top 5 individuals after 30 iterations	
x1	x2
1.822982e-002	2.179641e-002
3.459855e-002	3.892708e-002
6.789189e-002	2.156102e-002
-7.803050e-002	4.072982e-002
5.809648e-002	8.427665e-002

Top 5 individuals after 60 iterations	
x1	x2
6.570476e-003	7.855960e-003
1.247017e-002	1.403028e-002
2.446991e-002	7.771122e-003
-2.812411e-002	1.468003e-002
2.093940e-002	3.037537e-002

Top 5 individuals after 90 iterations	
x1	x2
2.441937e-003	2.919691e-003
4.634574e-003	5.214394e-003
9.094313e-003	2.888160e-003
-1.045241e-002	5.455876e-003
7.782188e-003	1.128910e-002

Top 5 individuals after 120 iterations	
x1	x2
9.349497e-004	1.117868e-003
1.774450e-003	1.996446e-003
3.481959e-003	1.105796e-003
-4.001936e-003	2.088903e-003
2.979584e-003	4.322281e-003

Top 5 individuals after 150 iterations	
x1	x2
3.684378e-004	4.405209e-004
6.992614e-004	7.867442e-004
1.372144e-003	4.357637e-004
-1.577052e-003	8.231788e-004
1.174171e-003	1.703291e-003

Fig. 13. Positions of top 5 artificial agents after iterations.

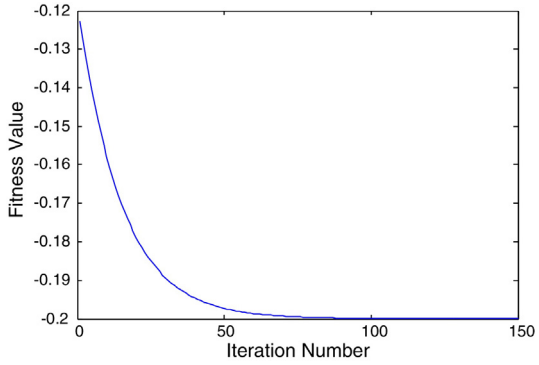

Fig. 14. Fitness value of the best artificial agent versus iteration number for Cosine Mixture Function.

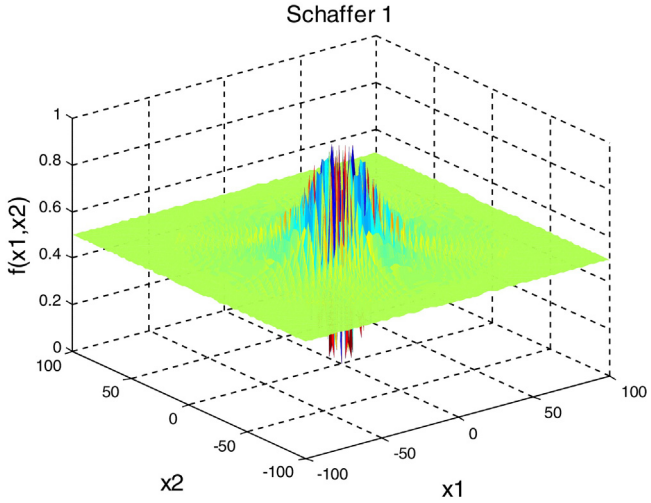

Fig. 15. 3-D plot for the two dimensional Schaffer1 Function.

global optimization problems. Additionally, the effectiveness of WSA algorithm in solving the constrained global optimization problems is introduced in the second part of this paper [25].

4.4. Computational study-3

This final part of the computational study is about the performance evaluation test of WSA over some selected complex problems from a recent benchmark set, CEC 2013 [80]. The selected problems, named as composition functions, were constructed by combining some multimodal and unimodal functions. Thus, these functions have much more complex structures than the basic functions and this property makes them suitable for performance evaluation test. The used composition functions are depicted in Table 10 and the reader can refer to [80] for detailed information about these functions.

The performance of WSA is evaluated over the complex composition functions by using five different dimensions setting: 2, 5 10, 30 and 50. The results, obtained by running WSA 30 times for per function, are given in Table 11 in comparison to the objective function values of these problems, because, there is not a scientific paper available for us used these functions as test bed. Besides, the values of *Maxiter* and *AA* were set to as 5000 and 50, respectively.

From the observation of the results given in Table 11, we can conclude that WSA could produce competitive results; however, we are not able to compare the results of WSA to other researches. Because of this, we cannot produce much concluding remarks about the performance of WSA on these complex composition functions.

4.5. Convergence analysis of WSA

This subsection is about to analyse the convergence behaviour of the introduced WSA algorithm on both unimodal and multimodal functions. Within this scope, we have selected two dimensional functions of Cosine Mixture Problem (CM(2)) as a unimodal function and Schaffer1 Problem (SF1) as a multimodal function. WSA have been initialized with 100 artificial agents and ran for a maximum number of 150 iterations for both functions within the context of this analysis. After that, the initial artificial agents and the artificial agents after every 30 iterations are scattered in order to visualize the convergences of the artificial agents throughout the execution of WSA. This visualization provides us to observe the

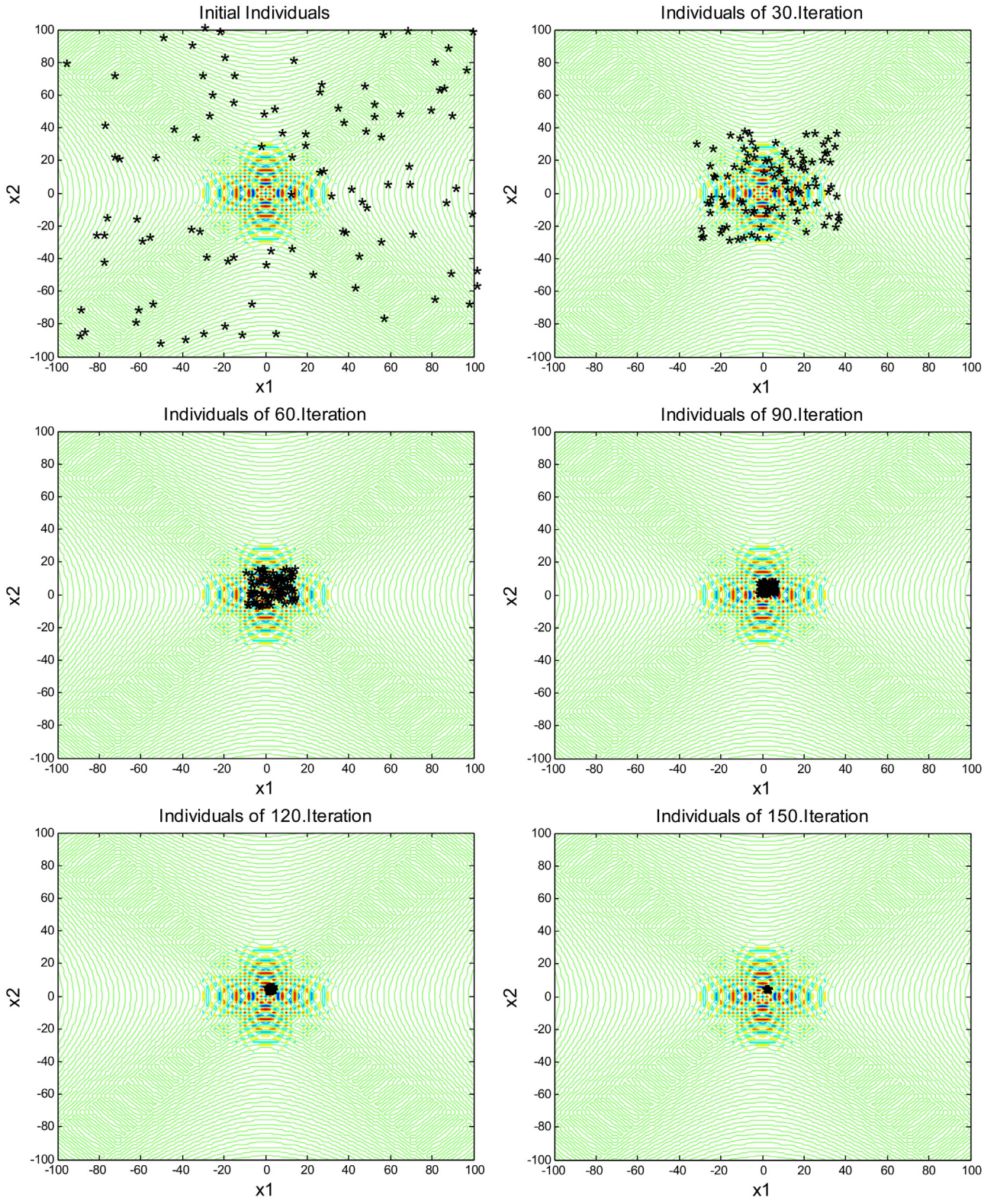

Fig. 16. Convergence behaviour of WSA on Schaffer1 Function.

convergence behaviour of WSA and to control if it converges towards to global optimum or not.

Two dimensional CM(2) is a maximization problem and has a maxima located at the origin with the function value of 0.20, and 3-D plot of this function is represented in Fig. 11.

The contour plot of the CM(2) was firstly constituted, then the artificial agents of the related iterations were scattered on a different contour plot of the function as can be seen in Fig. 12. By this way, we have the possibility to visualize the capability of WSA for converging towards the global optima and this provides us some

Initial top 5 Individuals	
x1	x2
23.6675	8.73265
25.7848	8.98118
5.56937	32.5308
29.2955	-5.87506
-35.5079	29.3619

Top 5 individuals after 30 iterations	
x1	x2
3.392370e+000	-1.680554e+000
3.563265e+000	6.233770e+000
-1.166666e+001	-9.462642e+000
-1.314645e+001	-9.208257e+000
1.579023e+001	1.149768e+001

Top 5 individuals after 60 iterations	
x1	x2
3.477893e+000	-6.974732e-001
6.611823e-001	3.861978e+000
1.229738e+000	2.151370e+000
4.108372e+000	-3.299829e+000
2.436995e+000	-6.401660e+000

Top 5 individuals after 90 iterations	
x1	x2
-8.362103e-001	-1.898272e+000
1.755768e+000	-1.759707e+000
2.053760e+000	2.047153e+000
1.858779e+000	2.521836e+000
-3.724484e-001	-2.977910e+000

Top 5 individuals after 120 iterations	
x1	x2
1.408051e-001	-6.975377e-002
6.098806e-001	-6.112488e-001
7.133907e-001	7.110955e-001
1.478984e-001	2.587415e-001
-1.271834e+000	-1.271334e+000

Top 5 individuals after 150 iterations	
x1	x2
4.906777e-002	-2.430779e-002
5.153962e-002	9.016623e-002
2.125312e-001	-2.130080e-001
8.312530e-002	1.239646e-001
1.757822e-001	1.937923e-001

Fig. 17. Positions of top 5 artificial agents after iterations.

concluding remarks on the convergence behaviour of the artificial agents.

Fig. 12 indicates that, WSA has a satisfactory level of convergence speed and the ability of intensively searching around the global optimum point as the number of iterations increases on the unimodal function of CM(2). The faster speeds of convergence may cause the algorithm to trap in local optima or to stagnate and is a challenging problem should be solved for the search algorithms. At this point, we can conclude that WSA has the ability of coping with the premature convergence problems for the unimodal functions. On the other hand, randomly generated initial artificial agents provide WSA to have a satisfactory level of diversification as can be seen in Fig. 12. For better illustration of the convergence speed of WSA, digital values for the best 5 five artificial agents after every 30 iterations are represented in Fig. 13. Additionally, the fitness value of the best artificial agents versus iteration number is presented in Fig. 14.

Fig. 14 illustrates the variation of the best fitness value as the search progress for 15,000 functions evaluations, 100 artificial agents discovered the search space for 150 iterations. We can claim that WSA converge to a promising area within the search space after about 75–80 iterations for the unimodal function of CM(2). After 15,000 function evaluations, WSA reaches the fitness value of 1.999956e−001 located at 0.3684e−003 for x_1 and 0.4405e−003 for x_2. At this point, the reader should consider that the results given in Table 5 are the values for 100,000 objective function evaluations.

The second part of the convergence analysis was done on a multimodal function of Schaffer1 Function (SF1). The SF1 function is a minimization problem and has an unknown number of local minima, however the global minimum of this function is located at $x^* = (0, 0)$ with $f^*(x_1, x_2) = 0$. 3-D plot of this function is represented in Fig. 15.

For the second part of the convergence analysis, the contour plot of the SF1 was also constituted and the artificial agents were also scattered on different contour plots of SF1 as can be seen in Fig. 16. This part of the analysis provides us more consistent information about the convergence behaviour of WSA, since SF1 has an unknown number of local optima. After this part of the analysis we will visualize the convergence behaviour of WSA and we will conclude consistently if WSA have the ability of coping effectively with the problems of premature convergence, trapping in a local optima and stagnation, or not.

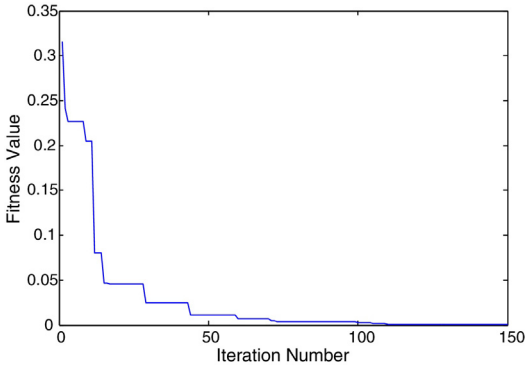

Fig. 18. Fitness value of the best artificial agent versus iteration number for Schaffer1 Function.

In addition to supporting the conclusions achieved from Fig. 8, Fig. 12 proves that WSA has the capability of successfully optimizing the multimodal functions without trapping in local optima. At this point, we can conclude that WSA has an effective balance between diversification (exploration) and intensification (exploitation), since it has not trapped in local optima and not stagnated. For better illustration of the convergence behaviour of WSA on SF1, digital values for the best 5 five artificial agents after every 30 iterations are also represented in Fig. 17. Additionally, the fitness value of the best artificial agent versus iteration number is also presented in Fig. 18.

Fig. 18 represents the variation of the best fitness value as the search progress for 15,000 functions evaluations, 100 artificial agents discovered the search space for 150 iterations. We can claim that WSA converge to a promising area within the search space after about 110–120 iterations for the multimodal function of SF1. After 15,000 function evaluations, WSA reaches the fitness value of 6.299156e−006 located at 0.0491 for x_1 and −0.0243 for x_2. At this point, the reader should also consider that the results given in Table 5 are the values for 100,000 objective function evaluations.

5. Conclusions

In this paper, we have introduced a new swarm based search algorithm named as Weighted Superposition Attraction (WSA), which was based on two basic mechanisms, "superposition" and "attracted movement of agents", that are observable in many systems. In many natural phenomena it is possible to compute superposition or weighted superposition of active fields as it is also applicable for social systems as well. An agent usually moves towards to this superposition due to the attractiveness of this superposition. As systems have dynamic structures their status changes over time, hence the superposition is also dynamic and changes over time; so it is required to be recomputed over time. This phenomenon was the main inspiration source while developing the WSA algorithm, which mainly simulates the dynamically changing superposition due to the dynamic nature of the system in combination with the attracted movement of agents as a search procedure for solving the optimization problems in an innovative manner.

In order to evaluate the capability of WSA in tackling the complex optimization problems, a set of computational experiments was carried out. The computational experiments was done in two parts: the first part evaluated the performance of WSA on a benchmark set contains 50 standard global optimization functions with different dimensions and the second part evaluated the performance of WSA on a benchmark set contains 13 standard global optimization functions with bigger sizes dimensions in comparison to different version of PSO published between the years of 1998 and 2015. The obtained results reveal the effectiveness of WSA in solving global optimization problems in terms of solution quality and execution time.

Future research directions will focus on adapting WSA so as to have the ability of effectively solving different combinatorial and multiple objective optimization problems and constrained optimization problems along with some other real life optimization applications.

Appendix A. Example Matlab code

The reader can refer to *http://web.deu.edu.tr/baykasoglu/wsa.html* for the Matlab code of WSA for Schaffer1 (SF1) function.

Appendix B. A numerical example

In this appendix we present a numerical example on the two dimensional Bohachevsky 1 Problem (BF1) (Eq. (A.1)) for two iterations and 10 function evaluations per iteration. The BF1 function is a minimization problem and has an unknown number of local minima, but the global minimum of this function is located at $x^* = (0, 0)$ with $f^*(x_1, x_2) = 0$.

$$\min_x f(x) = x_1^2 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7 \qquad (A.1)$$

WSA starts its search from the initial points with the fitness values and weights given in Table A.1. The weights values are assigned to each point according to their ranks among all points.

The target point is determined by using the initial points and their ranks as 7.7998 and 6.2517 for x_1 and x_2 respectively. Additionally, the fitness value of the target point is obtained as 140.1973. Now it is required to determine the search directions of each point with regard to target point's fitness value and each point's fitness value. Table A.2 presents the determined move directions for all solutions' dimensions and how each solution to select its move direction.

Table A.1
Initial points, their fitness values and weights.

Rank	x_1	x_2	Fitness value	Weight
1	−3.6492	1.2766	0.0180e+003	1.0000
2	−9.0515	−10.6006	0.3075e+003	0.5743
3	14.5633	7.3967	0.3219e+003	0.4152
4	24.2340	3.2412	0.6096e+003	0.3299
5	−30.0833	3.7657	0.9342e+003	0.2759
6	−17.8308	18.2644	0.9862e+003	0.2385
7	−22.1584	40.4562	3.7647e+003	0.2108
8	14.5718	42.8438	3.8842e+003	0.1895
9	3.7704	44.0982	3.9043e+003	0.1724
10	48.2787	30.1312	4.1476e+003	0.1585

Table A.2
Move directions for all points.

Point	Direction for x_1	Direction for x_2	How to select
1	−1	1	Randomly
2	1	1	Towards target point
3	−1	−1	Towards target point
4	−1	1	Towards target point
5	1	1	Towards target point
6	1	−1	Towards target point
7	1	−1	Towards target point
8	−1	−1	Towards target point
9	1	−1	Towards target point
10	−1	−1	Towards target point

Table A.3
Points after first iteration, their fitness values and weights.

Rank	x_1	x_2	Fitness value	Weight
1	−3.7770	1.3213	0.0189e+003	1.0000
2	−8.7347	−10.2296	0.2864e+003	0.5743
3	14.0536	7.1378	0.2999e+003	0.4152
4	23.3859	3.3547	0.5699e+003	0.3299
5	−29.0305	3.8975	0.8740e+003	0.2759
6	−17.2068	17.6252	0.9180e+003	0.2385
7	−21.3830	39.0404	3.5056e+003	0.2108
8	14.0618	41.3445	3.6171e+003	0.1895
9	3.9023	42.5549	3.6373e+003	0.1724
10	46.5891	29.0767	3.8617e+003	0.1585

After determining the search directions, each solution required to update its current position by using Eq. (1) (Section 3.2.2) according to the initial step length of 0.035. By the way the solutions move to newly determined positions given in Table A.3 after the first iteration. Additionally, the fitness values for the newly determined points are calculated and they ranked again according to their fitness values and weights are assigned to each solution with regard to its rank. Table A.3 also contains the fitness values and weights for the points determined after the first iteration.

The target point for the second iteration is determined as 7.5340 and 6.0329 for x_1 and x_2 respectively with the fitness value of 129.9826. Similar to first iteration move directions are determined for all points' directions as given in Table A.4. Additionally, points after the second iteration with their fitness values are also given in Table A.4.

Table A.4
Move directions for all points of the second iteration and points after second iteration and their fitness values.

Point	Direction for x_1	Direction for x_2	How to select	Points after second iteration		Fitness values
				x_1	x_2	
1	1	−1	Randomly	−3.6448	1.2751	0.0179e+003
2	1	1	Towards target point	−8.4290	−9.8716	0.2669e+003
3	−1	−1	Towards target point	13.5618	6.8880	0.2796e+003
4	−1	1	Towards target point	22.5676	3.4721	0.5336e+003
5	1	1	Towards target point	−28.0146	4.0339	0.8174e+003
6	1	−1	Towards target point	−16.6047	17.0085	0.8543e+003
7	1	−1	Towards target point	−20.6347	37.6743	3.2651e+003
8	−1	−1	Towards target point	13.5698	39.8977	3.3686e+003
9	1	−1	Towards target point	4.0389	41.0658	3.3893e+003
10	−1	−1	Towards target point	44.9588	28.0592	3.5966e+003

References

[1] S. Chetty, A.O. Adewumi, Three new stochastic local search algorithms for continuous optimization problems, Comput. Optim. Appl. 56 (2013) 675–721.
[2] J.H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, MI, 1975.
[3] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (1997) 341–359.
[4] M. Dorigo, V. Maniezzo, A. Colorni, Positive feedback as a search strategy, Politecnico di Milano, Italy, 1991 (Technical Report No: 91-016).
[5] J. Kennedy, R. Eberhart, Particle swarm optimization, Proc. IEEE Int. Conf. Neural Netw. 4 (1995) 1942–1948.
[6] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Computer Engineering Department Engineering Faculty, Erciyes University, Turkey, 2005 (Technical Report TR06).
[7] D.T. Pham, E. Koc, A. Ghanbarzadeh, S. Otri, S. Rahim, M. Zaidi, The bees algorithm – A novel tool for complex optimisation problems, in: Proceedings of IPROMS 2006 Conference, 2006, pp. 454–461.
[8] F. Glover, Tabu search – part 1, ORSA J. Comput. 1 (2) (1989) 190–206.
[9] F. Glover, Tabu search – part 2, ORSA J. Comput. 2 (1) (1990) 4–32.
[10] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Sci.: New Ser. 220 (1983) 671–680.
[11] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68.
[12] Ş.İ. Birbil, S.C. Fang, An electromagnetism-like mechanism for global optimization, J. Global Optim. 25 (2003) 263–282.
[13] D. Chaohua, Z. Yunfang, C. Weirong, Seeker optimization algorithm, in: Y. Wang, Y. Cheung, H. Liu (Eds.), Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin/Heidelberg, 2007, pp. 167–176 (CIS 2006).
[14] X.S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, 2008.
[15] E. Rashedi, H.S. Nezamabadi-pour, G.S.A. Saryazdi, A Gravitational search algorithm, Inf. Sci. 179 (2009) 2232–2248.
[16] X.S. Yang, A new metaheuristic bat-inspired algorithm, in: J. González, D. Pelta, C. Cruz, G. Terrazas, N. Krasnogor (Eds.), Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Springer, Berlin/Heidelberg, 2010, pp. 65–74.
[17] R. Rajabioun, Cuckoo optimization algorithm, Appl. Soft Comput. 11 (2011) 5508–5518.
[18] N. Ghorbani, E. Babaei, Exchange market algorithm, Appl. Soft Comput. 19 (2014) 177–187.
[19] M. Ghaemi, M-R. Feizi-Derakhshi, Forest optimization algorithm, Expert Syst. Appl. 41 (2014) 6676–6687.
[20] A. Askarzadeh, Bird mating optimizer: an optimization algorithm inspired by bird mating strategies, Commun. Nonlinear Sci. Numer. Simul. 19 (2014) 1213–1228.
[21] A.H. Gandomia, A.H. Alavi, Krill herd: a new bio-inspired optimization algorithm, Commun. Nonlinear Sci. Numer. Simul. 17 (2012) 4831–4845.
[22] E. Cuevas, M. Cienfuegos, D. Zaldívar, M. Pérez, A swarm optimization algorithm inspired in the behavior of the social-spider, Expert Syst. Appl. 40 (2013) 6374–6384.
[23] P. Civicioglu, Transforming geocentric Cartesian coordinates to geodetic coordinates by using differential search algorithm, Comput. Geosci. 46 (2012) 229–247.
[24] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Comput.-Aided Des. 43 (2011) 303–315.
[25] A. Baykasoğlu, Ş. Akpinar, Weighted Superposition Attraction (WSA): a swarm intelligence algorithm for optimization problems – part 2: constrained optimization, 37, 2015, pp. 396–415.
[26] X. Zhao, A perturbed particle swarm algorithm for numerical optimization, Appl. Soft Comput. 10 (1) (2010) 119–124.
[27] A. Azadeh, M.H. Farahani, H. Eivazy, S. Nazari-Shirkouhi, G. Asadipour, A hybrid metaheuristic algorithm for optimization of crew scheduling, Appl. Soft Comput. 13 (1) (2013) 158–164.
[28] G. Moslehi, M. Mahnam, A pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search, Int. J. Prod. Econ. 129 (2011) 14–22.
[29] C.J. Liao, E. Tjandradjaja, T.P. Chung, An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem, Appl. Soft Comput. 12 (2012) 1755–1764.
[30] S.A. Torabia, N. Sahebjamniaa, S.A. Mansourib, M. Aramon Bajestani, A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem, Appl. Soft Comput. 13 (2013) 4750–4762.
[31] Y. Delice, E.K. Aydoğan, U. Özcan, M.S. İlkay, A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing, J. Intell. Manuf. (2014), http://dx.doi.org/10.1007/s10845-014-0959-7.
[32] W.C. Chianga, T.L. Urbana, C. Luob, Balancing stochastic two-sided assembly lines, Int. J. Prod. Res. (2015), http://dx.doi.org/10.1080/00207543.2015.1029084.
[33] J.M. Nilakantan, S.G. Ponnambalam, Robotic U-shaped assembly line balancing using particle swarm optimization, Eng. Optim. (2015), http://dx.doi.org/10.1080/0305215X.2014.998664.
[34] T. Ting, M.V.C. Rao, C. Loo, A novel approach for unit commitment problem via an effective hybrid particle swarm optimization, IEEE Trans. Power Syst. 21 (1) (2006) 411–418.
[35] A.D. Asl, K.Y. Wong, Solving unequal-area static and dynamic facility layout problems using modified particle swarm optimization, J. Intell. Manuf. (2015), http://dx.doi.org/10.1007/s10845-015-1053-5.
[36] Y. Hana, J. Tanga, I. Kakub, L. Mu, Solving uncapacitated multilevel lot-sizing problems using a particle swarm optimization with flexible inertial weight, Comput. Math. Appl. 57 (2009) 1748–1755.
[37] C.F. Juang, A hybrid of genetic algorithm and particle swarm optimization-for recurrent network design, IEEE Trans. Syst. Man Cybern. B: Cybern. 34 (2) (2004) 997–1006.
[38] S.Z. Zhao, J.J. Liang, P.N. Suganthan, M.F. Tasgetiren, Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization, in: Proceedings of 2008 IEEE Congress on Evolutionary Computation (CEC'2008), IEEE, Hong Kong, 2008, pp. 3845–3852.
[39] A. Nickabadi, M.M. Ebadzadeh, R. Safabakhsh, A novel particle swarm optimization algorithm with adaptive inertia weight, Appl. Soft Comput. 11 (4) (2011) 3658–3670.
[40] Z.-H. Zhan, J. Zhang, Y. Li, H.S.-H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybern. B: Cybern. 139 (6) (2009) 1362–1381.
[41] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evol. Comput. 8 (3) (2004) 204–210.
[42] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: Proceedings of 2005 IEEE Swarm Intelligence Symposium (SIS'2005), Pasadena, CA, 2005, pp. 124–129.
[43] CKLT, T. Cao, Tribe-PSO: a novel global optimization algorithm and its application in molecular docking, Chemom. Intell. Lab. Syst. 82 (1) (2006) 248–259.
[44] Y.T. Kao, E. Zahara, A hybrid genetic algorithm and particle swarm optimization for multimodal functions, Appl. Soft Comput. 8 (2) (2008) 849–857.
[45] W.-J. Zhang, X.-F. Xie, DEPSO: hybrid particle swarm with differential evolution operator, in: Proceedings of the IEEE International Conference on Systems, Manand Cybernetics (SMC'2003), vol. 4, Washington, DC, 2003, pp. 3816–3821.
[46] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (2006) 281–295.
[47] H. Huang, H. Qin, Z. Hao, A. Lim, Example-based learning particle swarm optimization for continuous optimization, Inf. Sci. 182 (1) (2012) 125–138.
[48] F. van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 225–239.
[49] B. Niu, Y. Zhu, X. He, H. Wu, MCPSO: A multi-swarm cooperative particle swarm optimizer, Appl. Math. Comput. 185 (2) (2007) 1050–1062.
[50] C.H. Lee, C.T. Li, F.Y. Chang, A species-based improved electromagnetism-like mechanism algorithm for TSK-type interval-valued neural fuzzy system optimization, Fuzzy Set. Syst. 171 (2011) 22–43.
[51] C.H. Lee, F.K. Chang, C.T. Kuo, H.H. Chang, A hybrid of electromagnetism like mechanism and back-propagation algorithms for recurrent neural fuzzy systems design, Int. J. Syst. Sci. 43 (2012) 231–247.
[52] B. Javadi, F. Jolai, J. Slomp, M. Rabbani, R. Tavakkoli-Moghaddam, A hybrid electromagnetism-like algorithm for dynamic inter/intra cell layout problem, Int. J. Comput. Integr. Manuf. (2013), http://dx.doi.org/10.1080/0951192X.2013.814167.

[53] M. Khalili, R. Tavakkoli-Moghaddam, A multi-objective electromagnetism algorithm for a bi-objective flowshop scheduling problem, J. Manuf. Syst. 31 (2012) 232–239.
[54] Z. Naji-Azimi, P. Toth, L. Galli, An electromagnetism metaheuristic for the unicost set covering problem, Eur. J. Oper. Res. 205 (2010) 290–300.
[55] A. Yurtkuran, E. Emel, A new hybrid electromagnetism-like algorithm for capacitated vehicle routing problems, Expert Syst. Appl. 37 (2010) 3427–3433.
[56] A. Jamili, M.A. Shafia, R. Tavakkoli-Moghaddam, A hybridization of simulated annealing and electromagnetism-like mechanism for a periodic job shop scheduling problem, Expert Syst. Appl. 38 (2011) 5895–5901.
[57] P.C. Chang, S.H. Chen, C.Y. Fan, A hybrid electromagnetism-like algorithm for single machine scheduling problem, Expert Syst. Appl. 36 (2009) 1259–1267.
[58] Q. Wu, L. Gao, X. Li, C. Zhang, Y. Rong, Applying an electromagnetism-like mechanism algorithm on parameter optimisation of a multi-pass milling process, Int. J. Prod. Res. 51 (2013) 1777–1788.
[59] T. Mauder, C. Sandera, J. Stetina, M. Seda, Optimization of the quality of continuously cast steel slabs using the firefly algorithm, Mater. Tehnol. 45 (2011) 347–350.
[60] H. Banati, M. Bajaj, Promoting products online using firefly algorithm, in: 12th International Conference on Intelligent Systems Design and Applications (ISDA), 2012, pp. 580–585.
[61] O. Herbadji, K. Nadhir, L. Slimani, T. Bouktir, Optimal power flow with emission controlled using firefly algorithm, in: 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO), 2013.
[62] A. Baykasoğlu, F.B. Ozsoydan, An improved firefly algorithm for solving dynamic multi dimensional knapsack problems, Expert Syst. Appl. 41 (2014) 3712–3725.
[63] N.S. Grewal, M. Rattan, M.S. Patterh, A linear antenna array failure correction with null steering using firefly algorithm, Def. Sci. J. 64 (2014) 136–142.
[64] M. Younes, F. Khodja, R.L. Kherfane, Multi-objective economic emission dispatch solution using hybrid FFA (firefly algorithm) and considering wind power penetration, Energy 67 (2014) 595–606.
[65] A. Mishra, C. Agarwal, A. Sharma, P. Bedi, Optimized gray-scale image water marking using DWT-SVD and firefly algorithm, Expert Syst. Appl. 41 (2014) 7858–7867.
[66] S. Yu, S. Zhu, Y. Ma, D. Mao, A variable step size firefly algorithm for numerical optimization, Appl. Math. Comput. 263 (2015) 214–220.
[67] A. Baykasoğlu, F.B. Ozsoydan, Adaptive firefly algorithm with chaos for mechanical design optimization problems, Appl. Soft Comput. 36 (2015) 152–164.
[68] I. Fister Jr., I. Fister, X.-S. Yang, J. Brest, A comprehensive review of firefly algorithms, Swarm Evol. Comput. 13 (2013) 34–46.
[69] I. Fister, X.S. Yang, J. Brest Jr., I. Fister, Modified firefly algorithm using quaternion representation, Expert Syst. Appl. 40 (2013) 7220–7230.
[70] X.S. Yang, X. He, Firefly algorithm: recent advances and applications, Int. J. Swarm Intel. 1 (2013) 36–50.
[71] A. Baykasoğlu, Design optimization with chaos embedded great deluge algorithm, Appl. Soft Comput. 12 (2012) 1055–1067.
[72] Ş. Akpinar, A. Baykasoğlu, Multiple colony bees algorithm for continuous spaces, Appl. Soft Comput. 24 (2014) 829–841.
[73] J.M. Sutter, J.H. Kalivas, Convergence of generalized simulated annealing with variable step size with application toward parameter estimations of linear and nonlinear models, Anal. Chem. 63 (1991) 2383–2386.
[74] A. Ostermeier, A. Gawelczyk, N. Hansen, Step-size adaptation based on non-local use of selection information, in: Y. Davidor, H.-P. Schwefel, R. Männer (Eds.), Parallel Problem Solving from Nature – PPSN III of LNCS, vol. 866, Springer-Verlag, Berlin, Germany, 1994, pp. 189–198.
[75] C.D. Rosin, R.S. Halliday, W.E. Hart, R.K. Belew, A comparison of global and local search methods in drug docking, in: T. Back (Ed.), Proceedings of the Seventh International Conference on Genetic Algorithms, San Francisco, CA, Morgan Kaufmann Publishers, Inc., 1997, pp. 221–228.
[76] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol. Comput. 2 (2001) 159–195.
[77] S. Kern, S.D. Müller, N. Hansen, D. Büche, J. Očenášek, P. Koumoutsakos, Learning probability distributions in continuous evolutionary algorithms – a comparative review, Nat. Comput. 3 (2004) 77–112.
[78] M.M. Ali, C. Khompatraporn, Z. Zabinsky, A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems, J. Global Optim. 31 (2005) 635–672.
[79] Q. Qin, S. Cheng, Q. Zhang, L. Li, Y. Shi, Biomimicry of parasitic behavior in a coevolutionary particle swarm optimization algorithm for global optimization, Appl. Soft Comput. 32 (2015) 224–240.
[80] J.J. Liang, B.Y. Qu, P.N. Suganthan, G. Alfredo, Hernández-Díaz, Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization, Technical Report 12 Computational Intelligence Laboratory Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2012.
[81] G. Taguchi, Introduction to Quality Engineering: Designing Quality into Products and Processes, 1st ed., White Plains: Asian Productivity Organization, 1986.
[82] S. Boettcher, A.G. Percus, Extremal optimization for graph partitioning, Phys. Rev. E 64 (2001) 026114.
[83] S. Boettcher, Extremal optimization heuristics via coevolutionary avalanches, Comput. Sci. Eng. 2 (2000) 75–82.
[84] I.G. Lopes, F.L. de Sousa, L.C.G. DeSouza, The generalized extremal optimization with real codification, in: Proceedings of EngOpt 2008 – International Conference on Engineering Optimization, 2008.
[85] R. Ruiz, A. Allahverdi, Some effective heuristics for no wait flow shops with setup times to minimize total completion time, Ann. Oper. Res. 156 (2007) 143–171.
[86] H. Mosadegh, M. Zandieh, S.M.T. Fatemi Ghomi, Simultaneous solving of balancing and sequencing problems with station-dependent assembly times for mixed-model assembly lines, Appl. Soft Comput. 12 (2012) 1359–1370.
[87] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Proceedings of the1998 Congress on Evolutionary Computation (CEC'1998), Anchorage, AK, 1998, pp. 69–73.
[88] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (1) (2002) 58–73.
[89] T. Peram, K. Veeramachaneni, C.K. Mohan, Fitness-distance-ratio based particle swarm optimization, in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium (SIS'2003), Indianapolis, IN, 2003, pp. 174–181.
[90] A. Ratnaweera, S.K. Halgamuge, H.C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput. 8 (3) (2004) 240–255.
[91] S. Pasupuleti, R. Battiti, The gregarious particle swarm optimizer (G-PSO), in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'2006), Seattle, WA, 2006, pp. 67–74.
[92] S.-Y. Ho, H.-S. Lin, W.-H. Liauh, S.-J. Ho, OPSO: orthogonal particle swarm optimization and its application to task assignment problems, IEEE Trans. Syst. Man Cybern. A: Syst. Hum. 38 (2) (2008) 288–298.
[93] M.A.M. de Oca, T. Stützle, M. Birattari, M. Dorigo, Frankenstein's PSO: a composite particle swarm optimization algorithm, IEEE Trans. Evol. Comput. 13 (5) (2009) 1120–1132.
[94] Z.-H. Zhan, J. Zhang, Y. Li, Y. Shi, Orthogonal learning particle swarm optimization, IEEE Trans. Evol. Comput. 15 (6) (2011) 832–847.
[95] W.-N. Chen, J. Zhang, Y. Lin, N. Chen, Z.-H. Zhan, H.S.-H. Chung, Y. Li, Y. Shi, Particle swarm optimization with an aging leader and challengers, IEEE Trans. Evol. Comput. 17 (2) (2013) 241–258.