Information Sciences 293 (2015) 125–145

Contents lists available at ScienceDirect

Information Sciences

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

A new metaheuristic for numerical function optimization: Vortex Search algorithm

Berat Doğan *, Tamer Ölmez

Istanbul Technical University, Department of Electronics and Communication Engineering, Turkey

ARTICLE INFO

Article history:
Received 4 December 2013
Received in revised form 22 August 2014
Accepted 27 August 2014
Available online 18 September 2014

Keywords:
Metaheuristics
Global optimization
Function optimization
Simulated annealing
Artificial bee colony
Particle swarm optimization

ABSTRACT

In this study, a new single-solution based metaheuristic, namely the Vortex Search (VS) algorithm, is proposed to perform numerical function optimization. The proposed VS algorithm is inspired from the vortex pattern created by the vortical flow of the stirred fluids. To provide a good balance between the explorative and exploitative behavior of a search, the proposed method models its search behavior as a vortex pattern by using an adaptive step size adjustment scheme. The proposed VS algorithm is tested over 50 benchmark mathematical functions and the results are compared to both the single-solution based (Simulated Annealing, SA and Pattern Search, PS) and population-based (Particle Swarm Optimization, PSO2011 and Artificial Bee Colony, ABC) algorithms. A Wilcoxon-Signed Rank Test is performed to measure the pair-wise statistical performances of the algorithms, the results of which indicate that the proposed VS algorithm outperforms the SA, PS and ABC algorithms while being competitive with the PSO2011 algorithm.

1. Introduction

Many real-life optimization problems are complex in their nature and are difficult to solve. Exact optimization methods usually cannot provide a solution for these types of optimization problems. Some of the properties of such problems, such as high dimensionality, multimodality, epistasis (parameter interaction), and non-differentiability, render exact optimization methods impotent [41]. Hence, the use of some approximate algorithms remains as an alternative approach for the solution of these problems.

Approximate algorithms can be further decomposed into two classes: specific heuristics and metaheuristics [41]. Specific heuristics are problems dependent and designed only for the solution of a particular problem. However, metaheuristics represent a family of approximate algorithms that are more general and thus applicable to a large variety of optimization problems.

The words "meta" and "heuristic" both have their origin in the old Greek: "meta" means "upper level", and "heuristic" denotes the art of discovering new strategies [41]. Heuristic methods provide near optimal solutions in a reasonable amount of computational time without guaranteeing the optimality. Thus, the term metaheuristics can be defined as some intelligent strategies that enhance the efficiency of the heuristic methods [6].

A number of classification criteria have been proposed in the literature for the metaheuristics classification, including the search path that they follow, the use of memory, the type of neighborhood exploration used or the number of current

* Corresponding author.
E-mail addresses: bdogan@itu.edu.tr (B. Doğan), olmezt@itu.edu.tr (T. Ölmez).

http://dx.doi.org/10.1016/j.ins.2014.08.053
0020-0255/

Input: Initial solution s_0
$t = 0$;
Repeat
/* Generate candidate solutions (partial or complete neighborhood) from s_t */
Generate($C(s_t)$) ;
/* Select a solution from $C(s)$ to replace the current solution s_t */
s_{t+1} = Select($C(s_t)$) ;
$t = t + 1$;
Until the termination condition is met.
Output: Best solution found

Fig. 1. High-level representation of the single-solution based metaheuristics.

solutions transferred from one iteration round to the next. Among these criteria, the metaheuristic classification, which differentiates between the Single-Solution Based Metaheuristic and the Population-Based Metaheuristic, is often taken to be a fundamental distinction in the literature [41,8]. Single-solution based metaheuristics are also known as trajectory methods, which are based on a single solution at any time and comprise local search-based metaheuristics such as Simulated Annealing (SA) [30,9], Tabu Search (TS) [18], Iterated Local Search (ILS) [33], Guided Local Search (GLS) [3], Pattern Search (PS) [22], Random Search (RS) [36], and Variable Neighborhood Search (VNS) [20]. However, in population-based metaheuristics, a number of solutions is first created and then updated iteratively until the termination condition is satisfied. Population-based metaheuristics are generally studied under two major groups: Evolutionary algorithms and Swarm-based algorithms. Evolutionary algorithms are based on the notion of natural selection, which can be considered a competition between the species during the evolution process. In these algorithms, individual solutions are selected from a population of solutions according to their fitness value to generate new offspring by using some operators, such as the crossover and the mutation operators. Some well-known examples of the evolutionary algorithms are the Genetic Algorithm (GA) [21], Differential Evolution (DE) [39,40], and Estimation of Distribution Algorithms (EDA) [31]. Swarm- based algorithms are another class of population-based metaheuristics, which are inspired by the collective behavior of species, such as ants, bees, fish and birds. Swarm-based algorithms have the following characteristics: their particles are simple and non-sophisticated agents, they cooperate by an indirect communication medium, and they perform movements in the decision space [41]. Ant Colony Optimization (ACO) [12], Particle Swarm Optimization (PSO) [29] and Artificial Bee Colony algorithm (ABC) [5,25–27] are well-known examples of the swarm-based algorithms.

In the past two decades, both the single-solution based and population-based metaheuristics have been successfully applied to many real-world optimization problems. These algorithms produce solutions by exploring the search space efficiently while reducing the effective size of the search. Thus, the success of a metaheuristic method on a given optimization problem is defined by its ability to provide a good balance between the exploration and exploitation. The exploration defines the global search ability of the algorithm, whereas the exploitation is the ability to find the optimum around a near-optimal solution, which can also be considered as the local search ability. Because there is no any information provided regarding the search space in the initial steps, more exploration is required. However, as the algorithm converges to a near-optimal solution, more exploitation is required to tune the current solution towards the optimal one. The main differences between the existing metaheuristics concern the particular manner in which they attempt to achieve this balance [8]. Single-solution based metaheuristics are accepted to be more exploitation oriented, whereas population-based metaheuristics are more exploration oriented.

In this study, we propose a new single-solution based metaheuristic, namely, the Vortex Search (VS) algorithm, for the solution of bound-constrained global optimization problems. The proposed algorithm can be studied within the family of the search algorithms that comprises the Random Search and Pattern Search algorithms. The Random Search algorithm (which is also known as the Fixed Step Size Random Search) was proposed by Rastrigin [36], who introduced RS along with a basic mathematical analysis. RS functions by iteratively moving to better positions in the search space that are sampled from a hypersphere surrounding the current position. The PS, which was proposed by Hooke and Jeeves is a type of algorithm similar to the RS [38]. The problem with the above-mentioned algorithms is "the step size", which significantly affects the performance of the algorithms. To overcome this problem, a number of RS variants (e.g., Optimum Step Size Random Search (OSSRS) [38], Adaptive Step Size Random Search (ASSRS) [38], Optimized Relative Step Size Random Search (ORSSRS) [37]) were proposed. However, none of these algorithms could challenge the performance of population-based metaheuristics. Here, the proposed VS algorithm uses a new adaptive step size adjustment scheme that considerably improves the performance of the search process. Because the search behavior of the VS algorithm is inspired from the vortex pattern, we named the newly proposed algorithm the "Vortex Search" algorithm.

The proposed algorithm was tested on the 50 benchmark mathematical optimization functions and the obtained results were compared to the results found by the SA, PS, PSO2011, and ABC algorithms. The proposed VS algorithm was found to outperform the two single-solution based algorithms of SA and PS and a population-based algorithm ABC, while being competitive with another population-based algorithm, PSO2011. Because the proposed VS algorithm is very simple, this leads to a decrease in the computational time of the 50 benchmark functions when compared to the population-based algorithms.

The remaining part of this paper is organized as follows. The following section presents in detail the proposed VS algorithm. Section 3 covers the experimental results and discussion. Finally, Section 4 concludes the work.

2. The proposed vortex search algorithm

Single-solution based metaheuristics iteratively apply the generation and replacement procedures from the current single solution [41]. In the generation phase, a set of candidate solutions $C(s)$ is first created from the current solution s, while in the replacement phase a solution $s' \in C(s)$ is selected from $C(s)$ to replace the current solution s. This process iterates until the termination condition is met. Fig. 1 shows a high-level representation of the single-solution based metaheuristics [41].

Generation of candidate solutions by using some neighborhood structures is of critical importance for the success of the single-solution based metaheuristics. In single-solution based methods one of the main properties searched for a neighborhood is the locality. When small changes are made on the current solution, the neighborhood is said to have a strong locality. In contrast, a weak locality is characterized by a large effect on the solution, which results in the search being a random search in the search space. As mentioned in the previous section, an efficient exploration (a weak locality) is required in the initial steps. Once the algorithm converges to a near-optimal solution, further exploitation (strong locality) is required to tune the current solution towards to the optimal one. In the proposed VS algorithm, this balance is achieved by using a vortex-like search method.

2.1. Methodology

2.1.1. Generating the initial solution

Let us consider a two-dimensional optimization problem. In a two dimensional space a vortex pattern can be modeled by a number of nested circles. Here, the outer (largest) circle of the vortex is first centered on the search space, where the initial center μ_0 can be calculated using Eq. (1)

$$\mu_0 = \frac{upperlimit + lowerlimit}{2} \tag{1}$$

where *upperlimit* and *lowerlimit* are $d \times 1$ vectors that define the bound constraints of the problem in d dimensional space.

2.1.2. Generating the candidate solutions

A number of neighbor solutions $C_t(s)$, (t represents the iteration index and initially $t = 0$) are randomly generated around the initial center μ_0 in the d-dimensional space by using a Gaussian distribution. Here, $C_0(s) = \{s_1, s_2, \ldots, s_k\}$ $k = 1, 2, \ldots, n$ represents the solutions, and n represents the total number of candidate solutions. In Eq. (2), the general form of the multivariate Gaussian distribution is given.

$$p(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\} \tag{2}$$

where d represents the dimension, x is the $d \times 1$ vector of a random variable, μ is the $d \times 1$ vector of sample mean (center) and Σ is the covariance matrix. If the diagonal elements (variances) of the values of Σ are equal and if the off-diagonal elements (covariance) are zero (uncorrelated), then the resulting shape of the distribution will be spherical (which can be considered circular for a two-dimensional problem, as in our case). Thus, the value of Σ can be computed by using equal variances with zero covariance by using Eq. (3).

$$\Sigma = \sigma^2 \cdot [I]_{d \times d} \tag{3}$$

In Eq. (3), σ^2 represents the variance of the distribution and I represents the $d \times d$ identity matrix. The initial standard deviation (σ_0) of the distribution can be calculated by using Eq. (4).

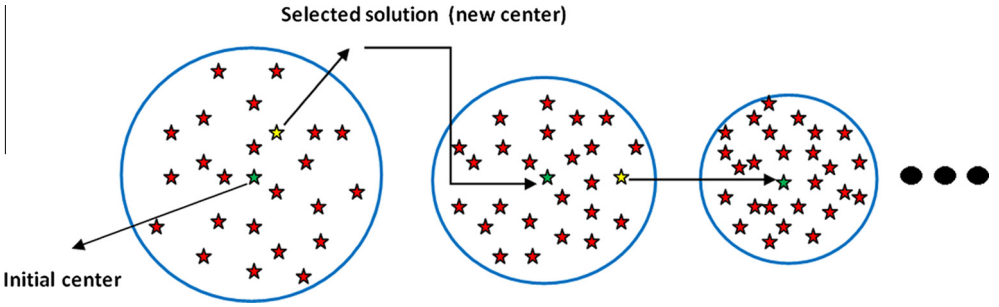

Fig. 2. An illustrative sketch of the search process.

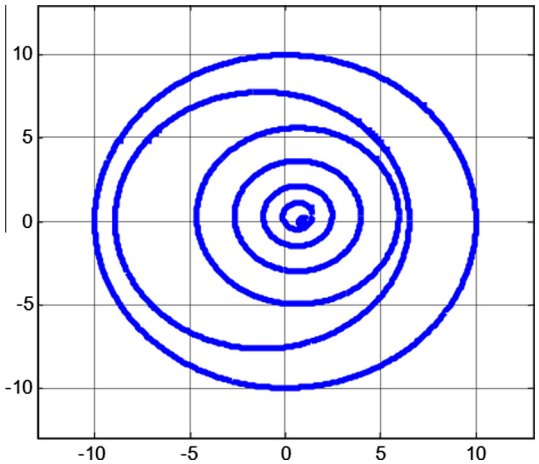

Fig. 3. A representative pattern showing the search boundaries (circles) of the VS algorithm after a search process, which has a vortex-like structure.

Inputs: Initial center μ_0 is calculated using Eq. 1
Initial radius r_0 (or the standard deviation, σ_0) is computed using Eq. 4
Fitness of the best solution found so far $f(s_{best}) = \inf$
$t = 0$;
Repeat
/* Generate candidate solutions by using Gaussian distribution around the center μ_t with a standard deviation (radius) r_t */
Generate($C_t(s)$) ;
If exceeded, then shift the $C_t(s)$ values into the boundaries as in Eq. 5
/* Select the best solution from $C_t(s)$ to replace the current center μ_t */
$s' =$ Select($C_t(s)$) ;
if $f(s') < f(s_{best})$
$s_{best} = s'$
$f(s_{best}) = f(s')$
else
keep the best solution so far s_{best}
end
/* Center is always shifted to the best solution found so far */
$\mu_{t+1} = s_{best}$
/* Decrease the standard deviation (radius) for the next iteration */
$r_{t+1} =$ Decrease(r_t)
$t = t + 1$;
Until the maximum number of iterations is reached
Output: Best solution found so far s_{best}

Fig. 4. A description of the proposed VS algorithm.

$$\sigma_0 = \frac{\max(upperlimit) - \min(lowerlimit)}{2} \tag{4}$$

Here, σ_0 can also be considered as the initial radius (r_0) of the outer circle for a two dimensional optimization problem. Because a weak locality is required in the initial phases, r_0 is chosen to be a large value. Thus, a full coverage of the search space by the outer circle is provided in the initial step. This process provides a bird's-eye view for the problem at hand.

2.1.3. Replacement of the current solution

In the selection phase, a solution (which is the best one) $s' \in C_0(s)$ is selected and memorized from $C_0(s)$ to replace the current circle center μ_0. Prior to the selection phase, the candidate solutions must be ensured to be inside the search boundaries. For this purpose, the solutions that exceed the boundaries are shifted into the boundaries, as in Eq. (5).

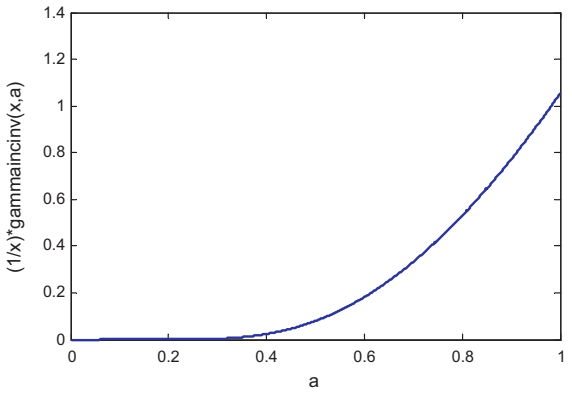

Fig. 5. $(1/x)\cdot gammaincinv(x,a)$ where $x = 0.1$ and $a \in [0,1]$.

$$s_k^i = \begin{cases} rand \cdot (upperlimit^i - lowerlimit^i) + lowerlimit^i, & s_k^i < lowerlimit^i \\ s_k^i, & lowerlimit^i \leqslant s_k^i \leqslant upperlimit^i \\ rand \cdot (upperlimit^i - lowerlimit^i) + lowerlimit^i, & s_k^i > upperlimit^i \end{cases} \tag{5}$$

where $k = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, d$ and *rand* is a uniformly distributed random number. Next, the memorized best solution s' is assigned to be the center of the second circle (the inner one). In the generation phase of the second step, the effective radius (r_1) of this new circle is reduced, and then, a new set of solutions $C_1(s)$ is generated around the new center. Note that in the second step, the locality of the generated neighbors increased with the decreased radius. In the selection phase of the second step, the new set of solutions $C_1(s)$ is evaluated to select a solution $s' \in C_1(s)$. If the selected solution is better than the best solution found so far, then this solution is assigned to be the new best solution and it is memorized. Next, the center of the third circle is assigned to be the memorized best solution found so far. This process iterates until the termination condition is met. An illustrative sketch of the process is given in Fig. 2. In this manner, once the algorithm is terminated, the resulting pattern appears as a vortex-like structure, where the center of the smallest circle is the optimum point found by the algorithm. A representative pattern is sketched in Fig. 3 for a two-dimensional optimization problem for which the upper and lower limits are between the $[-10,10]$ interval. A description of the VS algorithm is also provided in Fig. 4.

As indicated by Fig. 4, the proposed VS algorithm is quite simple. Different from the high-level representation of the single-solution based metaheuristics shown in Fig. 1, the proposed VS algorithm uses a poor memory (in which only the best solution is memorized) and an additional step in which the radius is iteratively decreased. The use of a poor memory is not new for the single-solution based metaheuristics. The iterated local search algorithm (ILS) and PS algorithm also use a similar type of memory approach [7]. The radius decrement process can be considered as a type of adaptive step-size adjustment process, which is also used in RS (Random Search) algorithms. However, the method by which this adjustment is performed is of critical importance for the success of the algorithms. This process should be performed in such a way that allows the

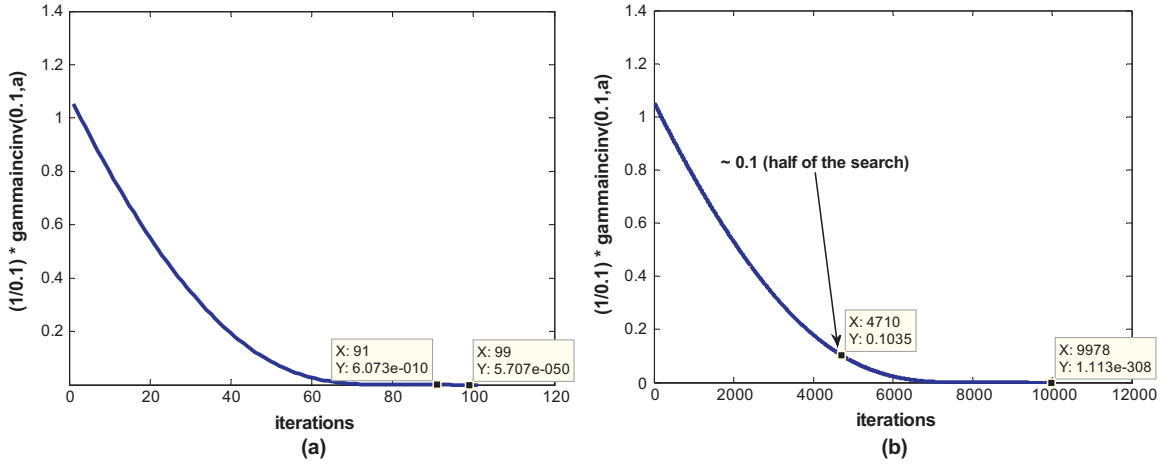

Fig. 6. $(1/0.1)\cdot gammaincinv(0.1,a)$ for (a) *MaxItr* = 100 (b) *MaxItr* = 10,000.

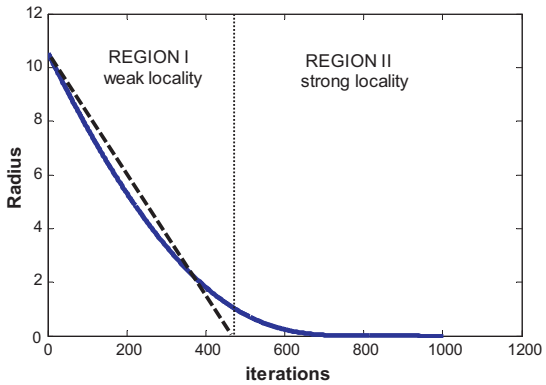

Fig. 7. Change of the radius for a problem defined within the $[-10, 10]$ interval (step size = 0.001).

algorithm to behave in an explorative manner in the initial steps and in an exploitative manner in the latter steps. To achieve this type of process, the value of the radius must be tuned properly during the search process.

2.1.4. The radius decrement process

In the VS algorithm, the inverse incomplete gamma function is used to decrease the value of the radius during each iteration pass.

The incomplete gamma function given in Eq. (6) most commonly arises in probability theory, particularly in those applications involving the chi-square distribution [4].

$$\gamma(x,a) = \int_0^x e^{-t}t^{a-1}dt \quad a > 0 \tag{6}$$

where $a > 0$ is known as the shape parameter and $x \geqslant 0$ is a random variable. In conjunction with the incomplete gamma function, its complementary $\Gamma(x,a)$ is usually also introduced (Eq. (7)).

$$\Gamma(x,a) = \int_x^\infty e^{-t}t^{a-1}dt \quad a > 0 \tag{7}$$

Thus, it follows that,

$$\gamma(x,a) + \Gamma(x,a) = \Gamma(a) \tag{8}$$

where $\Gamma(a)$ is known as the gamma function. There exist many studies in the literature on different proposed methods for the numerical calculation of the incomplete gamma function [16,44,2].

MATLAB® also provides some tools for the calculation of functions including, gamma function (*gamma*), incomplete gamma function (*gammainc*), and inverse incomplete gamma (*gammaincinv*) function. The inverse incomplete gamma function (*gammaincinv*), computes the inverse of the incomplete gamma function with respect to the integration limit x and represented as *gammaincinv(x,a)* in MATLAB®. In Fig. 5 the inverse incomplete gamma function is plotted for $x = 0.1$ and $a \in [0, 1]$. Here, for our case the parameter a of the inverse incomplete gamma function defines the resolution of the search. By equally sampling a values within $[0, 1]$ interval at a certain step size, the resolution of the search can be adjusted. For this purpose, at each iteration, a value of a is computed by using the Eq. (9)

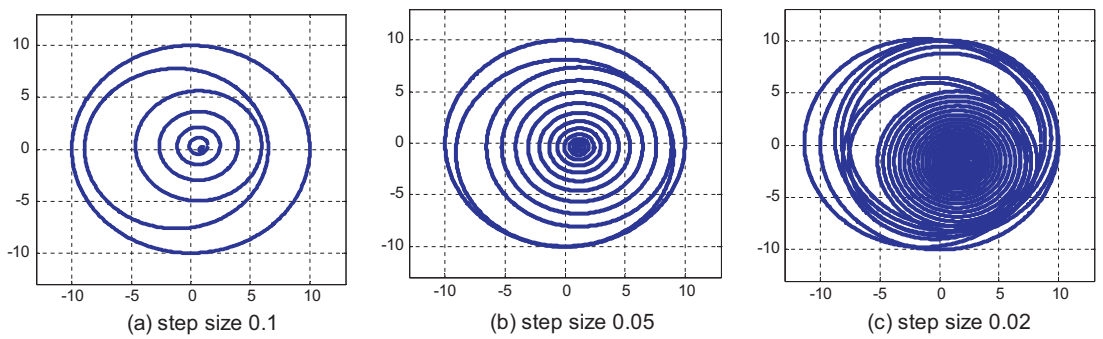

Fig. 8. Resolution of the search increases with a decrease in the step size (increased iteration number).

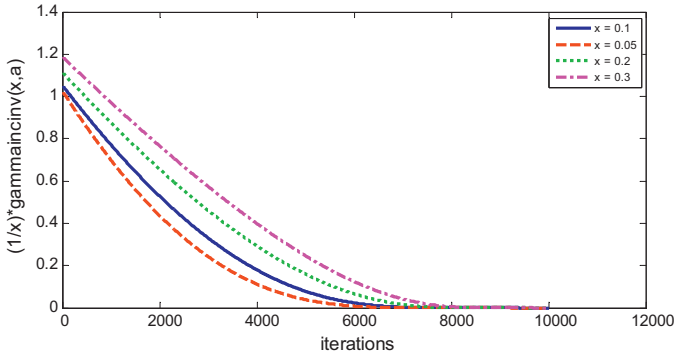

Fig. 9. $(1/x) \cdot gammaincinv(x,a)$ function for different x values (step size = 0.0001).

$$a_t = a_0 - \frac{t}{MaxItr} \tag{9}$$

where a_0 is selected as $a_0 = 1$ to ensure a full coverage of the search space at the first iteration, t is the iteration index, and *MaxItr* represents the maximum number of iterations.

Thus, for different values of *MaxItr*, the sampling rate of the a values are changed, thereby changing the resolution of the search. In Fig. 6(a) and (b) sample plots for the inverse incomplete gamma function are given with respect to the iteration number for a fixed value of $x = 0.1$. In Fig. 6(a), *MaxItr* is selected to be 100, and in Fig. 6(b), *MaxItr* is selected to be 10,000, which results a values ranging from 1 down to 0 with a step size of 0.01 and 0.0001, for Fig. 6(a) and (b), respectively. Note that $(1/x) \cdot gammaincinv(x,a) \approx 1$ for $a = 1$, which means $gammaincinv(x,a) \approx x$ for $a = 1$. Here, the choice for x determines the value of the $(1/x) \cdot gammaincinv(x,a)$ that will be reached at approximately half of the number of iterations (Fig. 6b).

From Fig. 6(a) and (b), it can be clearly shown that as the number of iterations increases (step size decreases) the value of the function decreases significantly. For example, in Fig. 6b, for the iteration number of 9978, the resulting value of the function is 1.113e−308, which is a very small number. However, until half of the number of iterations is reached, the function behaves approximately linearly. This behavior allows us to analyze the function in two separate regions.

Let us consider an optimization problem defined within the $[-10, 10]$ region. The initial radius r_0 can be calculated with Eq. (10). Because $a_0 = 1$, the resulting function value is $(1/x) \cdot gammaincinv(x, a_0) \approx 1$, which means $r_0 \approx \sigma_0$ as indicated before.

$$r_0 = \sigma_0 \cdot (1/x) \cdot gammaincinv(x, a_0) \tag{10}$$

By means of Eq. (4), the initial radius value r_0 can be calculated as $r_0 \approx 10$. In Eq. (11), a general formula is also given to obtain the value of the radius at each iteration pass.

$$r_t = \sigma_0 \cdot (1/x) \cdot gammaincinv(x, a_t) \tag{11}$$

Here, t represents the iteration index. In Fig. 7, the change of the radius with respect to the iteration number is given for this case. From Fig. 7, it can be shown that, for the first half of the number of iterations, the radius changes approximately linearly. Thus, the algorithm has a weak locality in this region. In contrast, in the other half, the value of the function decreases significantly. Thus, the algorithm has a strong locality in this region.

As shown above, the inverse incomplete gamma function meets the objectives expected from a successful search. The main drawback of using such a method to tune the radius is the dependence of the convergence speed on the number of iterations. As the number of iterations increase (step size decreases), the resolution of the search also increases. Because the corresponding search space is thoroughly explored, in some cases, this could be seen as an advantage. However, usually, a small step size (e.g., iteration number > 500,000) lowers the convergence speed of the problem. Fortunately, most of the problems are solved within the 100,000 iterations, which is trivial for the VS algorithm. In Fig. 8(a)–(c), the effect of step size on the search boundaries is given.

The proposed VS algorithm is quite simple and does not require any additional parameters, except the number of iterations, the number of neighbor solutions, the upper and lower limits of the problem and the dimension of the problem, which are common parameters for all of the other metaheuristics. The only parameter that could be is the x value. However, the x value can also be selected as a fixed value. In Fig. 9, $(1/x) \cdot gammaincinv(x,a)$ is plotted for different x values of a certain step size.

As found from Fig. 9, at approximately half of the number of iterations, the value of the $gammaincinv(x,a) \approx x$. Although a detailed analysis is not performed, for a fixed value of $x = 0.1$ the VS algorithm performs well. The detailed results are given in Section 3.

Table 1
Benchmark functions used in experiments D: Dimension, C: Characteristics, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable.

No.	Range	D	C	Function	Formulation
F1	$[-5.12, 5.12]$	5	US	Stepint	$f(x) = 25 + \sum_{i=1}^{5} \lceil x_i \rceil$
F2	$[-100, 100]$	30	US	Step	$f(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$
F3	$[-100, 100]$	30	US	Sphere	$f(x) = \sum_{i=1}^{n} (x_i)^2$
F4	$[-10, 10]$	30	US	SumSquares	$f(x) = \sum_{i=1}^{n} (ix_i)^2$
F5	$[-1.28, 1.28]$	30	US	Quartic	$f(x) = \sum_{i=1}^{n} (ix_i)^4 + \text{random}[0, 1)$
F6	$[-4.5, 4.5]$	5	UN	Beale	$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$
F7	$[-100, 100]$	2	UN	Easom	$f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$
F8	$[-10, 10]$	2	UN	Matyas	$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$
F9	$[-10, 10]$	4	UN	Colville	$f(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$
F10	$[-D^2, D^2]$	6	UN	Trid6	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
F11	$[-D^2, D^2]$	10	UN	Trid10	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
F12	$[-5, 10]$	10	UN	Zakharov	$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4$
F13	$[-4, 5]$	24	UN	Powell	$f(x) = \sum_{i=1}^{n/k} (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4$
F14	$[-10, 10]$	30	UN	Schwefel 2.22	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
F15	$[-10, 10]$	30	UN	Schwefel 1.2	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$
F16	$[-30, 30]$	30	UN	Rosenbrock	$f(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$
F17	$[-10, 10]$	30	UN	Dixon–Price	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$
F18	$[-65.536, 65.536]$	2	MS	Foxholes	$f(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})6}\right]^{-1}$
F19	$[-5, 10] \times [0, 15]$	2	MS	Branin	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$
F20	$[-100, 100]$	2	MS	Bohachevsky1	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$
F21	$[-10, 10]$	2	MS	Booth	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$
F22	$[-5.12, 5.12]$	30	MS	Rastrigin	$f(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$
F23	$[-500, 500]$	30	MS	Schwefel	$f(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$
F24	$[0, \pi]$	2	MS	Michalewicz2	$f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(ix_i^2/\pi))^{2m}$, $m = 10$
F25	$[0, \pi]$	5	MS	Michalewicz5	$f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(ix_i^2/\pi))^{2m}$, $m = 10$
F26	$[0, \pi]$	10	MS	Michalewicz10	$f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(ix_i^2/\pi))^{2m}$, $m = 10$
F27	$[-100, 100]$	2	MN	Schaffer	$f(x) = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}$
F28	$[-5, 5]$	2	MN	Six Hump Camel Back	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
F29	$[-100, 100]$	2	MN	Bohachevsky2	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)(4\pi x_2) + 0.3$
F30	$[-100, 100]$	2	MN	Bohachevsky3	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$
F31	$[-10, 10]$	2	MN	Shubert	$f(x) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$
F32	$[-2, 2]$	2	MN	GoldStein-Price	$f(x) = \left[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \cdot \left[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$
F33	$[-5, 5]$	4	MN	Kowalik	$f(x) = \sum_{i=1}^{11} \left(a_i - \frac{x_1(b_i^2 + b_ix_2)}{b_i^2 + b_ix_3 + x_4}\right)^2$

Table 1 (*continued*)

No.	Range	D	C	Function	Formulation
F34	$[0,10]$	4	MN	Shekel5	$f(x)=-\sum_{i=1}^{5}\sum_{j=1}^{4}\left[(x_j-a_{ij})(x_j-a_{ij})^T+c_i\right]^{-1}$
F35	$[0,10]$	4	MN	Shekel7	$f(x)=-\sum_{i=1}^{7}\sum_{j=1}^{4}\left[(x_j-a_{ij})(x_j-a_{ij})^T+c_i\right]^{-1}$
F36	$[0,10]$	4	MN	Shekel10	$f(x)=-\sum_{i=1}^{10}\sum_{j=1}^{4}\left[(x_j-a_{ij})(x_j-a_{ij})^T+c_i\right]^{-1}$
F37	$[D,D]$	4	MN	Perm	$f(x)=\sum_{k=1}^{n}\left[\sum_{i=1}^{n}(i^k+\beta)((x_i/i)^k-1)\right]^2$
F38	$[0,D]$	4	MN	PowerSum	$f(x)=\sum_{k=1}^{n}\left[(\sum_{i=1}^{n}x_i^k)-b_k\right]^2$
F39	$[0,1]$	3	MN	Hartman3	$f(x)=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right]$
F40	$[0,1]$	6	MN	Hartman6	$f(x)=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2\right]$
F41	$[-600,600]$	30	MN	Griewank	$f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$
F42	$[-32,32]$	30	MN	Ackley	$f(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$
F43	$[-50,50]$	30	MN	Penalized	$f(x)=\frac{\pi}{n}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_n-1)^2\right\}$ $+\sum_{i=1}^{n}u(x_i,10,100,4)y_i=1+\frac{1}{4}(x_i+1)u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\leqslant x_i\leqslant a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$
F44	$[-50,50]$	30	MN	Penalized2	$f(x)=0.1\left\{\sin^2(\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right]+(x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\right\}$ $+\sum_{i=1}^{n}u(x_i,5,100,4)$
F45	$[0,10]$	2	MN	Langerman2	$f(x)=-\sum_{i=1}^{m}c_i\left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\right)$
F46	$[0,10]$	5	MN	Langerman5	$f(x)=-\sum_{i=1}^{m}c_i\left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\right)$
F47	$[0,10]$	10	MN	Langerman10	$f(x)=-\sum_{i=1}^{m}c_i\left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\right)$
F48	$[d]$	2	MN	Fletcher Powell2	$f(x)=\sum_{i=1}^{n}(A_i-B_i)^2$ $A_i=\sum_{j=1}^{n}(a_{ij}\sin\alpha_j+b_{ij}\cos\alpha_j),\quad B_i=\sum_{j=1}^{n}(a_{ij}\sin x_j+b_{ij}\cos x_j)$
F49	$[d\times 1]$	5	MN	Fletcher Powell5	$f(x)=\sum_{i=1}^{n}(A_i-B_i)^2$ $A_i=\sum_{j=1}^{n}(a_{ij}\sin\alpha_j+b_{ij}\cos\alpha_j),\quad B_i=\sum_{j=1}^{n}(a_{ij}\sin x_j+b_{ij}\cos x_j)$
F50	$[d\times 1]$	10	MN	Fletcher Powell10	$f(x)=\sum_{i=1}^{n}(A_i-B_i)^2$ $A_i=\sum_{j=1}^{n}(a_{ij}\sin\alpha_j+b_{ij}\cos\alpha_j),\quad B_i=\sum_{j=1}^{n}(a_{ij}\sin x_j+b_{ij}\cos x_j)$

3. Experimental results

The proposed VS algorithm is tested on 50 benchmark functions, that are obtained from the study performed by Karaboğa and Akay [28]. In their study, Karaboğa and Akay compared the performance of the ABC algorithm to the GA, PSO, and DE algorithms. By using the same functions, in this study, the performance of the proposed VS algorithm is compared to the algorithms SA, PS, PSO2011 and ABC. SA and PS are two well-known single-solution based algorithms, and PSO2011 [35,48] is an extension of the standard PSO algorithm. In the literature, in addition to the newly proposed optimization algorithms [17,19], some extensions of the above-mentioned algorithms are also proposed [13,32,14,15,10,42,1,43,34,23]. However, using well-known standard algorithms in the comparisons enables the interpretation of the results over a larger group.

To evaluate the performances of the algorithms, two different set of experiments are performed. In the first set of experiments, the overall performances of the algorithms are studied for a constant number of iterations. After a certain number of iterations, the algorithms are evaluated according to the mean and best fitness values found for each benchmark function. In the second set of experiments, the convergence behavior of the algorithms is studied. For this purpose, a different number of iterations is selected and the algorithms are run to evaluate the mean fitness value found for each case. Thus, an iteration based convergence behavior of the algorithms is obtained. In literature, a time-based comparison of the optimization algorithms is also used to evaluate the convergence behavior [13]. However, in our case, this comparison was not possible. Because, the proposed VS algorithm is an iteration-dependent algorithm, to perform the VS algorithm, one must set a certain number of iterations. However, the proposed VS algorithm behaves quite different for different numbers of iterations. Because the resolution of the search differs for different number of iterations, the VS algorithm gives quite different results in a time-based comparison.

Table 2
Statistical results of 30 runs obtained by SA, PS, PSO2011, ABC and VS algorithms (values < 10^{-16} are considered as 0).

No.	Min.		SA	PS	PSO2011	ABC	VS
F1	0	Mean	1.866666667	0	0	0	0
		StdDev	1.136641554	0	0	0	0
		Best	0	0	0	0	0
F2	0	Mean	0	0	0.066666667	0	0.2
		StdDev	0	0	0.253708132	0	0.406838102
		Best	0	0	0	0	0
F3	0	Mean	0	0	0	2.78624E−16	0
		StdDev	0	0	0	0	0
		Best	0	0	0	2.23487E−16	0
F4	0	Mean	0	0	0	2.75098E−16	0
		StdDev	0	0	0	0	0
		Best	0	0	0	1.85594E−16	0
F5	0	Mean	0.4028326	0.049370406	1.64098E−05	0.013732963	0.000145026
		StdDev	0.301544881	0.046578461	5.56581E−06	0.002379448	7.30549E−05
		Best	0.001414536	1.61333E−05	7.13993E−06	0.008413424	5.54996E−05
F6	0	Mean	0.000430475	0	0	6.37598E−16	0
		StdDev	0.000943865	0	0	3.58687E−16	0
		Best	2.51078E−08	0	0	0	0
F7	−1	Mean	−0.028827505	−8.11022E−05	−1	−1	−1
		StdDev	0.157894721	0	0	0	0
		Best	−0.864825008	−8.11022E−05	−1	−1	−1
F8	0	Mean	0	0	0	0	0
		StdDev	0	0	0	0	0
		Best	0	0	0	0	0
F9	0	Mean	1.83377047	0.002199995	0	0.00576453	0
		StdDev	2.351638954	0	0	0.003966867	0
		Best	0.000971812	0.002199995	0	0.000383073	0
F10	−50	Mean	−49.84789091	−50	−50	−50	−50
		StdDev	0.150775917	0	3.61345E−14	4.94748E−14	2.96215E−14
		Best	−49.98701123	−50	−50	−50	−50
F11	−210	Mean	−209.5023223	−209.9954224	−210	−210	−210
		StdDev	0.230476381	0	2.30778E−13	9.62204E−12	6.19774E−13
		Best	−209.8801988	−209.9954224	−210	−210	−210
F12	0	Mean	0	0	0	7.56674E−14	0
		StdDev	0	0	0	3.76382E−14	0
		Best	0	0	0	2.31887E−14	0
F13	0	Mean	0	0	2.04664E−07	9.09913E−05	1.43967E−05
		StdDev	0	0	1.21051E−08	1.42475E−05	2.27742E−06
		Best	0	0	1.72679E−07	5.23427E−05	5.71959E−06
F14	0	Mean	0	0	1.094284383	8.51365E−16	0
		StdDev	0	0	0.870781136	0	0
		Best	0	0	0.107097937	6.93597E−16	0
F15	0	Mean	0	0	0	0.000760232	0
		StdDev	0	0	0	0.000440926	0
		Best	0	0	0	0.00027179	0
F16	0	Mean	0.224618742	9.84185348	0.930212233	0.003535257	0.367860114
		StdDev	0.097171414	0	1.714978077	0.003314818	1.130879848
		Best	0.082077849	9.84185348	0	7.08757E−05	9.42587E−05
F17	0	Mean	0.990721802	0.666666667	0.666666667	1.91607E−15	0.666666667
		StdDev	0.029412712	0	4.38309E−16	2.55403E−16	7.68909E−16
		Best	0.871516993	0.666666667	0.666666667	1.1447E−15	0.666666667
F18	0.998	Mean	5.5682975	0.998003838	34.26621987	0.998003933	0.998003838
		StdDev	4.367922182	4.51681E−16	126.6004794	4.33771E−07	0
		Best	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838
F19	0.398	Mean	0.398269177	0.397887358	0.397887358	0.397887358	0.397887358
		StdDev	0.001624387	0	0	0	0
		Best	0.397887361	0.397887358	0.397887358	0.397887358	0.397887358
F20	0	Mean	0	0	0	0	0
		StdDev	0	0	0	0	0
		Best	0	0	0	0	0

Table 2 (*continued*)

No.	Min.		SA	PS	PSO2011	ABC	VS
F21	0	Mean	5.28496E−05	0	0	0	0
		StdDev	7.35674E−05	0	0	0	0
		Best	4.08428E−08	0	0	0	0
F22	0	Mean	0	0	26.11016129	0	57.60799224
		StdDev	0	0	5.686650032	0	13.94980276
		Best	0	0	16.91429893	0	33.82857771
F23	−12569.5	Mean	−1891.275468	−3686.285205	−8316.185447	−12569.48662	−11283.05416
		StdDev	137.3913021	2.77513E−12	463.9606712	1.85009E−12	352.1869262
		Best	−2188.304761	−3686.285205	−9466.201047	−12569.48662	−11799.62928
F24	−1.8013	Mean	−1.792778285	−1.80130341	−1.80130341	−1.80130341	−1.80130341
		StdDev	0.043874926	1.35504E−15	9.03362E−16	9.03362E−16	9.03362E−16
		Best	−1.801296643	−1.80130341	−1.80130341	−1.80130341	−1.80130341
F25	−4.6877	Mean	−3.670604734	−4.495893207	−4.67700874	−4.687658179	−4.670953055
		StdDev	0.496257736	2.71009E−15	0.036487971	2.60778E−15	0.020809276
		Best	−4.684023442	−4.495893207	−4.687658179	−4.687658179	−4.687658179
F26	−9.6602	Mean	−6.060491565	−8.461507306	−9.204154798	−9.660151716	−8.793361668
		StdDev	0.504024688	5.42017E−15	0.298287637	0	0.382153549
		Best	−6.880235805	−8.461507306	−9.660151716	−9.660151716	−9.410563187
F27	0	Mean	0	0	0	0	0
		StdDev	0	0	0	0	0
		Best	0	0	0	0	0
F28	−1.03163	Mean	−1.031621639	−1.031628453	−1.031628453	−1.031628453	−1.031628453
		StdDev	2.1595E−05	4.51681E−16	6.71219E−16	6.77522E−16	6.77522E−16
		Best	−1.031628448	−1.031628453	−1.031628453	−1.031628453	−1.031628453
F29	0	Mean	0	0	0	0	0
		StdDev	0	0	0	0	0
		Best	0	0	0	0	0
F30	0	Mean	0	0	0	0	0
		StdDev	0	0	0	0	0
		Best	0	0	0	0	0
F31	−186.73	Mean	−186.7309087	−123.5767709	−186.7309088	−186.7309088	−186.7309088
		StdDev	5.76173E−07	0	4.49449E−13	1.18015E−14	3.76909E−14
		Best	−186.7309088	−123.5767709	−186.7309088	−186.7309088	−186.7309088
F32	3	Mean	3.000000254	30	3	3	3
		StdDev	4.36073E−07	1.08403E−14	1.22871E−15	1.7916E−15	1.44961E−15
		Best	3	30	3	3	3
F33	0.00031	Mean	0.002635099	0.00031966	0.000307486	0.000319345	0.000307486
		StdDev	0.001644496	0	0	5.4385E−06	0
		Best	0.000780214	0.00031966	0.000307486	0.00030894	0.000307486
F34	−10.15	Mean	−9.002836723	−5.055197729	−9.363375596	−10.15319968	−10.15319968
		StdDev	2.071338221	9.03362E−16	2.081063878	7.2269E−15	7.2269E−15
		Best	−10.15247689	−5.055197729	−10.15319968	−10.15319968	−10.15319968
F35	−10.4	Mean	−8.979515615	−5.087671825	−10.40294057	−10.40294057	−10.40294057
		StdDev	2.744123131	3.61345E−15	1.80672E−15	1.04311E−15	1.61598E−15
		Best	−10.40272816	−5.087671825	−10.40294057	−10.40294057	−10.40294057
F36	−10.53	Mean	−8.498294797	−5.128480787	−10.53640982	−10.53640982	−10.53640982
		StdDev	2.645882377	3.61345E−15	0	2.13774E−15	1.47518E−15
		Best	−10.5362255	−5.128480787	−10.53640982	−10.53640982	−10.53640982
F37	0	Mean	0.844338683	0.295334941	0.002854996	0.003526435	0.002815467
		StdDev	1.292248967	1.1292E−16	0.007218334	0.001604834	0.002374325
		Best	0.001422252	0.295334941	1.30581E−08	0.00097117	0
38	0	Mean	0.021367747	0	3.14986E−05	0.000288005	1.78046E−06
		StdDev	0.032095897	0	6.43525E−05	0.00013892	1.28089E−06
		Best	3.02411E−05	0	1.50435E−11	5.82234E−05	4.82E−09
F39	−3.86	Mean	−3.861918832	−3.862782148	−3.862782148	−3.862782148	−3.862782148
		StdDev	0.001277024	2.25841E−15	2.71009E−15	2.71009E−15	2.69625E−15
		Best	−3.86274843	−3.862782148	−3.862782148	−3.862782148	−3.862782148
F40	−3.32	Mean	−3.281839942	−3.203161918	−3.318394475	−3.322368011	−3.322368011
		StdDev	0.03736791	1.80672E−15	0.021763955	6.54548E−16	5.14996E−16
		Best	−3.318182614	−3.203161918	−3.322368011	−3.322368011	−3.322368011

(*continued on next page*)

Table 2 (*continued*)

No.	Min.		SA	PS	PSO2011	ABC	VS
F41	0	Mean	0	0	0.004761038	0	0.032798017
		StdDev	0	0	0.008047673	0	0.018570459
		Best	0	0	0	0	0.00739604
F42	0	Mean	8.88178E−16	8.88178E−16	0.660186991	2.44545E−14	1.15463E−14
		StdDev	0	0	0.711496752	3.02083E−15	3.61345E−15
		Best	8.88178E−16	8.88178E−16	7.99361E−15	2.22045E−14	7.99361E−15
F43	0	Mean	1.668971097	0	0.024187276	2.63417E−16	0.114662313
		StdDev	1.1292E−15	0	0.080213839	0	0.532276418
		Best	1.668971097	0	0	1.29727E−16	0
F44	0	Mean	3	0	0	2.7797E−16	0
		StdDev	0	0	0	0	0
		Best	3	0	0	2.22214E−16	0
F45	−1.08	Mean	−1.072219306	−1.080938442	−1.080938442	−1.080938442	−1.080938442
		StdDev	0.017963587	9.03362E−16	4.51681E−16	4.96507E−16	4.51681E−16
		Best	−1.080936396	−1.080938442	−1.080938442	−1.080938442	−1.080938442
F46	−1.5	Mean	−0.491126582	−0.303738989	−1.499999223	−1.499999223	−1.499999223
		StdDev	0.154632766	0	6.77522E−16	1.05365E−15	6.77522E−16
		Best	−0.797704495	−0.303738989	−1.499999223	−1.499999223	−1.499999223
F47	NA	Mean	−0.017734088	−0.422042106	−1.069011938	−1.482016588	−1.271399999
		StdDev	0.018288461	1.6938E−16	0.422205043	0.097662612	0.313658787
		Best	−0.075917322	−0.422042106	−1.5	−1.499998488	−1.5
F48	0	Mean	1.12203E−07	0	0	0	0
		StdDev	4.54332E−07	0	0	0	0
		Best	4.99041E−12	0	0	0	0
F49	0	Mean	765.3235916	0	3.083487114	1.48707E−12	0
		StdDev	1025.49687	0	4.389694328	8.11041E−12	0
		Best	3.65426E−05	0	0	3.1715E−16	0
F50	0	Mean	7148.686851	18.79802113	580.0839029	1.111095363	0
		StdDev	14013.62432	3.61345E−15	1280.698395	0.598962098	0
		Best	0.016208788	18.79802113	0	0.182153237	0

3.1. Benchmark functions

The functions used in the test are listed in Table 1, and include many different types of problems, such as unimodal, multimodal, regular, irregular, separable, non-separable, and multidimensional. For a list of constant parameters used in some functions, please refer to Appendix A.

A function is called unimodal if it has one global optimum point with no or a single local optimum point, whereas a multimodal function is a function with many local optimum points. For multimodal functions, an optimization algorithm is tested for its ability to avoid local optimum points. If the algorithm has a poor level of exploration ability, it cannot thoroughly explore the search space and is thus likely to select a local optimum point. Functions that have a flat search space (Stepint, Matyas, PowerSum) are also difficult for the algorithms, as in the multimodal functions because, a flat search space does not provide any gradient information to direct the algorithm towards the optimum point [28]. Another group of functions includes separable and non-separable functions. A p-variable separable function can be expressed as the sum of p functions of one variable. Non-separable functions cannot be written in this form because they have an interrelation among their variables [28]. Therefore, optimization of the non-separable functions is more difficult than the separable ones.

In a high-dimensional space, algorithms usually face another problem, which is known as the curse of dimensionality. As the dimensionality of the problem increases, the volume of the space increases so rapidly that the data points become sparse. A search algorithm that is effective in small dimensions might exhibit poor performances in such high dimensional spaces. Therefore, it is common to test algorithms to evaluate their ability of finding global optima in high dimensional spaces. In some functions, the global minima is very small when compared to the entire search space, such as the functions of Easom, Michalewicz ($m = 10$), and Powell. For problems such as Perm, Kowalik, and Schaffer, the global minimum is located very close to the local minima. If the algorithm cannot adapt to the direction changes in the functions having a narrow curving valley (e.g., Beale, Colville), it will fail when applied to these types of problems [45].

Another problem that algorithms suffer is the scaling problem, with a difference of many magnitude orders between the domain and the frequency of the hypersurface (GoldStein-Price, Trid) [24]. Some functions, such as Fletcher–Powell and Langerman, are non-symmetrical and their local optima are randomly distributed. Because the objective functions have no implicit symmetry, optimization of these functions is difficult for certain algorithms. A quartic function is padded with random noise (either Gaussian or uniform), which ensures the algorithm to never produce the same value on the same point.

Table 3
Pair-wise statistical comparison of the algorithms by Wilcoxon Signed-Rank Test ($\alpha = 0.05$).

Function	VS vs. SA				VS vs. PS				VS vs. PSO2011				VS vs. ABC			
	p-value	T+	T−	Winner	*p*-value	T+	T−	Winner	*p*-value	T+	T−	Winner	*p*-value	T+	T−	Winner
F1	2.871E−06	0	406	+	1	0	0	=	1	0	0	=	1	0	0	=
F2	0.0143059	21	0	−	0.014306	21	0	−	0.0455	10	0	−	0.014306	21	0	−
F3	1	0	0	=	1	0	0	=	1	0	0	=	1.73E−06	0	465	+
F4	1	0	0	=	1	0	0	=	1	0	0	=	1.73E−06	0	465	+
F5	1.734E−06	0	465	+	1.92E−06	1	464	+	1.73E−06	465	0	−	1.73E−06	0	465	+
F6	1.734E−06	0	465	+	1	0	0	=	1	0	0	=	3.79E−06	0	406	+
F7	1.014E−07	0	465	+	4.32E−08	0	465	+	1	0	0	=	1	0	0	=
F8	1	0	0	=	1	0	0	=	1	0	0	=	0.067889	0	10	=
F9	1.734E−06	0	465	+	4.32E−08	0	465	+	1	0	0	=	1.73E−06	0	465	+
F10	1.734E−06	0	465	+	1.96E−07	0	465	+	0.0455	10	0	−	1.21E−06	0	465	+
F11	1.734E−06	0	465	+	9.57E−07	0	465	+	0.001986	99	6	−	1.69E−06	0	465	+
F12	1	0	0	=	1	0	0	=	1	0	0	=	1.73E−06	0	465	+
F13	1.734E−06	465	0	−	1.73E−06	465	0	−	1.73E−06	465	0	−	1.73E−06	0	465	+
F14	1	0	0	=	1	0	0	=	1.73E−06	0	465	+	1.73E−06	0	465	+
F15	1	0	0	=	1	0	0	=	1	0	0	=	1.73E−06	0	465	+
F16	0.0027653	87	378	+	1.73E−06	0	465	+	0.130592	306	159	=	0.002957	88	377	+
F17	1.717E−06	0	465	+	1.71E−06	465	0	−	0.00085	63.5	371.5	+	1.72E−06	465	0	−
F18	1.724E−06	0	465	+	4.32E−08	0	465	+	0.10247	0	6	=	1.73E−06	0	465	+
F19	1.734E−06	0	465	+	1	0	0	=	1	0	0	=	1	0	0	=
F20	1	0	0	=	1	0	0	=	1	0	0	=	1	0	0	=
F21	1.734E−06	0	465	+	1	0	0	=	1	0	0	=	1	0	0	=
F22	1.733E−06	465	0	−	1.73E−06	465	0	−	1.92E−06	464	1	−	1.73E−06	465	0	−
F23	1.734E−06	0	465	+	1.73E−06	0	465	+	1.73E−06	0	465	+	1.73E−06	465	0	−
F24	1.734E−06	0	465	+	4.32E−08	0	465	+	1	0	0	=	1	0	0	=
F25	1.734E−06	0	465	+	1.42E−06	0	465	+	0.018985	182.5	48.5	−	2.12E−06	435	0	−
F26	1.734E−06	0	465	+	0.000332	58	407	+	0.000306	408	57	−	1.73E−06	465	0	−
F27	1	0	0	=	1	0	0	=	1	0	0	=	1	0	0	=
F28	1.734E−06	0	465	+	4.32E−08	0	465	+	0.317311	0	1	=	1	0	0	=
F29	1	0	0	=	1	0	0	=	1	0	0	=	1	0	0	=
F30	1	0	0	=	1	0	0	=	1	0	0	=	0.001766	0	66	+
F31	1.734E−06	0	465	+	1.2E−06	0	465	+	0.000492	22.5	230.5	+	0.000141	342	36	−
F32	1.734E−06	0	465	+	7.45E−07	0	465	+	0.067045	196.5	79.5	=	0.002375	33.5	219.5	+
F33	1.734E−06	0	465	+	1.59E−06	0	465	+	0.00016	281.5	18.5	−	1.73E−06	0	465	+
F34	1.734E−06	0	465	+	4.32E−08	0	465	+	0.0656	0	10	=	1	0	0	=
F35	1.734E−06	0	465	+	3.26E−07	0	465	+	0.014306	21	0	−	0.000967	19	152	+
F36	1.734E−06	0	465	+	6.25E−07	0	465	+	7.74E−06	210	0	−	0.001054	0	78	+
F37	4.286E−06	9	456	+	1.73E−06	0	465	+	0.082206	317	148	=	0.184622	168	297	=
F38	1.734E−06	0	465	+	1.73E−06	465	0	−	0.001593	79	386	+	1.73E−06	0	465	+
F39	1.734E−06	0	465	+	7.24E−08	0	435	+	0.317311	1	0	=	0.317311	1	0	=
F40	1.734E−06	0	465	+	1.44E−07	0	465	+	0.705457	6	4	=	0.032509	22.5	82.5	+
F41	1.733E−06	465	0	−	1.73E−06	465	0	−	5.22E−06	454	11	−	1.73E−06	465	0	−
F42	8.207E−07	465	0	−	8.21E−07	465	0	−	0.00499	66	285	+	1.38E−06	0	465	+
F43	2.933E−07	1	464	+	0.042168	15	0	−	0.632281	26.5	18.5	=	0.057096	140	325	=
F44	4.32E−08	0	465	+	1	0	0	=	1	0	0	=	1.73E−06	0	465	+
F45	1.734E−06	0	465	+	4.32E−08	0	465	+	1	0	0	=	0.025347	0	15	+
F46	1.734E−06	0	465	+	4.32E−08	0	465	+	1	0	0	=	8.83E−07	0	435	+
F47	1.734E−06	0	465	+	8.01E−07	0	465	+	0.061202	76.5	199.5	=	0.416534	272	193	=
F48	1.734E−06	0	465	+	1	0	0	=	1	0	0	=	1	0	0	=
F49	1.734E−06	0	465	+	1	0	0	=	0.003346	0	66	+	1.73E−06	0	465	+
F50	1.734E−06	0	465	+	4.32E−08	0	465	+	8.84E−05	0	210	+	1.73E−06	0	465	+
+/=/−	35/10/5				25/17/8				8/30/12				26/16/8			

Table 4
Problem-based comparison of the proposed VS algorithm.

Problem type	VS vs. SA	VS vs. PS	VS vs. PSO2011	VS vs. ABC
US	2/2/1	1/3/1	0/3/2	3/1/1
UN	7/4/1	5/5/2	2/7/3	9/2/1
MS	7/1/1	5/3/1	1/5/3	1/4/4
MN	19/3/2	14/6/4	5/15/4	13/9/2
Total (+/=/−)	35/10/5	25/17/8	8/30/12	26/16/8

This function allows us to test algorithms for their performance on the noisy data. The algorithms that do not perform well on this function will also fail on noisy data.

3.2. Algorithm settings

Population based metaheuristics (ABC, PSO2011) are selected to have a population size of 50, which is also the number of neighborhood solutions of the proposed VS algorithm. The SA algorithm always performs with a single solution, and the PS algorithm creates its own neighbor vectors (pattern). The acceleration coefficients (c_1 and c_2) of the PSO2011 algorithm are both set to 1.8, and the inertia coefficient is set to 0.6, as in [28]. The *limit* value for the ABC algorithm is determined as $limit = SN * D$, where SN represents the number of food sources and D represents the dimension. As mentioned before in the previous section, the proposed VS algorithm is a parameter-free algorithm. There are no additional parameters for the VS algorithm.

For the first set of experiments, the maximum number of iterations is selected as 500,000 to evaluate the overall performances of the algorithms. For the second set of experiments, the number of iterations is selected as 100, 1000 and 10,000 to evaluate the convergence behavior of the algorithms.

3.3. Results

3.3.1. Overall performances of the algorithms

In the first set of experiments, the proposed VS algorithm is compared to the SA, PS, PSO2011 and ABC algorithms by using the 50 benchmark functions given in Table 1. For each algorithm, 30 different runs are performed, and the mean and the best values are recorded. The maximum number of iterations is selected to be 500,000, as mentioned previously. For the SA and PS algorithms, the MATLAB® Global Optimization Toolbox is used, and the other algorithms are also coded in MATLAB®. For the PSO2011, ABC and VS algorithms please refer to [35,46,47]. For each algorithm, all of the functions are run in parallel using a 32 core Intel® CPU 32 GB RAM workstation. For the first set of experiments, results are presented in Table 2.

Although the statistical results presented in Table 2 provide a first insight into the performance of the algorithms, a pairwise statistical test is typically used for a better comparison. For this purpose, by using the results obtained from 30 runs of each algorithm, a Wilcoxon Signed-Rank Test is performed with a statistical significance value $\alpha = 0.05$. The null hypothesis H_0 for this test is: "There is no difference between the median of the solutions produced by algorithm A and the median of the solutions produced by algorithm B for the same benchmark problem", i.e., median (A) = median (B). To determine whether algorithm A reached a statistically better solution than algorithm B, or if not, whether the alternative hypothesis is valid, the sizes of the ranks provided by the Wilcoxon Signed-Rank Test (i.e., T+ and T−, as defined in [11]) are examined.

Most of the modern software development tools use an arithmetic precision of 10^{-16} in the double-precision mode. An arithmetic precision value that is higher than necessary makes it difficult to compare the local search abilities of the algorithms [11]. For this purpose, during the statistical pair-wise comparison, resulting values below 10^{-16} are considered as 0.

In Table 3, the statistical pair-wise results of the VS algorithm compared to those of other algorithms are given. In this table, '+' indicates cases in which the null hypothesis is rejected and the VS algorithm exhibited a statistically superior performance in the pair-wise Wilcoxon Signed-Rank Test at the 95% significance level ($\alpha = 0.05$); '−' indicates cases in which the null hypothesis is rejected and the VS algorithm displayed an inferior performance; and '=' indicates cases in which there is no statistical different between two algorithms. The last row of the Table 3 shows the total count of (+/=/−) the three statistical significance cases in the pair-wise comparison. From this table, it can be shown that the VS algorithm outperforms the SA, PS and ABC algorithms and compete with the PSO2011 algorithm. The SA algorithm performs a pure random search over the search space for which obtained results become meaningful. The PS algorithm is also a single-solution based algorithm that performs poorly compared to the VS algorithm. The ABC algorithm is a powerful swarm-based algorithm that is used successfully for the solution of many types of optimization problems. From Table 2, it can be shown that the difference between the VS and ABC algorithms is mainly due to the local search ability of the VS algorithm. For a number of functions,

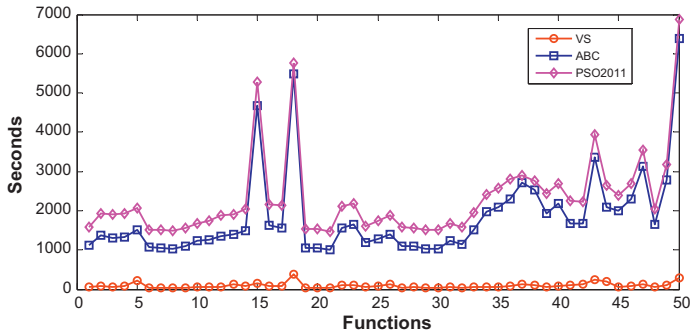

Fig. 10. Average computational time of 30 runs for 50 benchmark functions (500,000 iterations).

Table 5
Average results of 30 runs to study the convergence behavior of the algorithms. Exp-1 = 100, Exp-2 = 1000 and Exp-3 = 10,000 iterations (values <10^{-16} are considered as 0).

No.	Min.		SA	PS	PSO2011	ABC	VS
F1	0	Exp-1	8.266666667	1	0	0	1.7
		Exp-2	3.366666667	0	0	0	0
		Exp-3	4	0	0	0	0
F2	0	Exp-1	0	0	21.76666667	1407.7	246.2333333
		Exp-2	0	0	0.133333333	0	5.266666667
		Exp-3	0	0	0.1	0	2
F3	0	Exp-1	0	0	17.83386799	1295.311832	207.816882
		Exp-2	0	0	0	1.64239E-11	2.51339E−08
		Exp-3	0	0	0	4.64604E−16	0
F4	0	Exp-1	0	0	11.84694367	152.3545786	69.58526034
		Exp-2	0	0	0.000333638	2.08038E−12	0.009149095
		Exp-3	0	0	0	4.37095E−16	1.41264E−15
F5	0	Exp-1	0.47953561	0.283638723	0.066563724	1.472308609	0.436218853
		Exp-2	0.427474235	0.064990433	0.006226974	0.111736883	0.032896182
		Exp-3	0.364413261	0.067813379	0.000748077	0.02993958	0.005195975
F6	0	Exp-1	0.754518564	0.061084986	6.2751E−05	0.009077853	2.07862E−09
		Exp-2	0.022823365	3.31263E−07	1.24268E−10	3.40978E−06	0
		Exp-3	0.037855479	0	2.02464E−15	3.86825E−12	0
F7	−1	Exp-1	−4.33472E−09	−8.11021E−05	−1	−0.666175773	−1
		Exp-2	−2.51646E−07	−8.11022E−05	−1	−0.99872753	−1
		Exp-3	−1.98058E−06	−8.11022E−05	−1	−0.999999336	−1
F8	0	Exp-1	0	0	6.20157E−14	0.001179646	1.0176E−14
		Exp-2	0	0	0	1.56512E−08	0
		Exp-3	0	0	0	7.43831E−15	0
F9	0	Exp-1	19.12830629	20.259375	0.361837754	1.407757675	1.788528801
		Exp-2	3.661805918	0.00330069	5.51134E−06	0.301430841	0.00102707
		Exp-3	2.488655875	0.002199995	0	0.075522598	3.53655E−16
F10	−50	Exp-1	−25.75587151	−5	−49.99994001	−49.56419749	−49.99920138
		Exp-2	−48.8472231	−49.75	−50	−49.99998522	−50
		Exp-3	−48.85964687	−50	−50	−50	−50
F11	−210	Exp-1	−23.47203906	10	−204.178138	−154.3120633	−187.758909
		Exp-2	−126.8987485	−94	−210	−209.135915	−209.9999894
		Exp-3	−126.2521879	−209.9589844	−210	−209.9999999	−210
F12	0	Exp-1	0	0	0.243903567	42.85390927	0.011121307
		Exp-2	0	0	0	1.828101769	7.82812E−15
		Exp-3	0	0	0	0.000221146	0
F13	0	Exp-1	0	0	2.285803279	27.91221909	20.67093817
		Exp-2	0	0	0.047460485	0.122474319	0.284153185
		Exp-3	0	0	0.000456088	0.003219558	0.00716883
F14	0	Exp-1	0	0	3.759761969	2.92866529	36.51488013
		Exp-2	0	0	1.401158988	7.53555E−07	0.034092402
		Exp-3	0	0	1.210521235	1.15906E−15	8.26573E−08
F15	0	Exp-1	0	0	2559.631822	35203.6207	9251.760935
		Exp-2	0	0	1.10605399	11612.21356	15.93412113
		Exp-3	0	0	0	1294.286411	1.14751E−11
F16	0	Exp-1	29	29	1034.076117	293570.9219	18932.75325
		Exp-2	28.83571779	27.81680679	46.66422891	1.613886475	252.1545704
		Exp-3	28.82692118	25.14455549	16.31958178	0.392342246	75.95278982
F17	0	Exp-1	1	1	17.3424916	2044.257203	187.724627
		Exp-2	0.999958145	0.666992188	0.680708375	0.02370942	1.090003416
		Exp-3	1	0.666666667	0.666666667	5.81096E−15	0.669410059
F18	0.998	Exp-1	11.08157418	0.998003839	51.31249642	442.0799609	41.02927716
		Exp-2	6.467075194	0.998003838	84.2031109	39.81361114	1.19667749
		Exp-3	6.648011115	0.998003838	34.26881766	1.31818964	0.998003838
F19	0.398	Exp-1	0.745748422	0.399103668	0.39788741	0.397887358	0.397887358
		Exp-2	0.398421335	0.397887358	0.397887358	0.397887358	0.397887358
		Exp-3	0.398272764	0.397887358	0.397887358	0.397887358	0.397887358

(*continued on next page*)

Table 5 (*continued*)

No.	Min.		SA	PS	PSO2011	ABC	VS
F20	0	Exp-1	0	0	5.43941E−12	0	2.42354E−12
		Exp-2	0	0	0	0	0
		Exp-3	0	0	0	0	0
F21	0	Exp-1	0.314734743	0	5.01241E−14	1.17502E−05	3.65947E−12
		Exp-2	0.000156561	0	0	0	0
		Exp-3	0.000175671	0	0	0	0
F22	0	Exp-1	0	0	166.8509083	91.85093274	134.1911749
		Exp-2	0	0	39.20933653	0.157157274	90.01036413
		Exp-3	0	0	28.28451301	0	71.43784888
F23	−12569.5	Exp-1	−516.8821802	−221.1097472	−4688.812483	−8574.098182	−7942.151101
		Exp-2	−837.1458474	−1695.174729	−6541.101632	−12184.34762	−9866.229322
		Exp-3	−836.2994779	−3686.207854	−8170.468922	−12569.48662	−10459.83993
F24	−1.8013	Exp-1	−1.159905575	−1.801293149	−1.80130341	−1.80130341	−1.80130341
		Exp-2	−1.79290997	−1.80130341	−1.80130341	−1.80130341	−1.80130341
		Exp-3	−1.794159626	−1.80130341	−1.80130341	−1.80130341	−1.80130341
F25	−4.6877	Exp-1	−2.153552487	−2.402497914	−4.568771105	−4.686708108	−4.145175998
		Exp-2	−3.199309786	−4.495893207	−4.634488914	−4.687658179	−4.449685947
		Exp-3	−3.20317597	−4.495893207	−4.64120811	−4.687658179	−4.567268598
F26	−9.6602	Exp-1	−3.145901017	−1.296343016	−6.721497307	−9.145677633	−6.991477404
		Exp-2	−4.85399641	−8.208602594	−8.603123358	−9.658759622	−7.913983601
		Exp-3	−4.962493257	−8.461507306	−8.965074725	−9.660151716	−8.686465036
F27	0	Exp-1	0	0	0.001632485	0.005802235	0.008744319
		Exp-2	0	0	0	3.48268E−05	0.001943182
		Exp-3	0	0	0	5.18835E−09	0
F28	−1.03163	Exp-1	−0.779030894	−1.031551548	−1.031628453	−1.031628453	−1.031628453
		Exp-2	−1.031473907	−1.031628453	−1.031628453	−1.031628453	−1.031628453
		Exp-3	−1.031453693	−1.031628453	−1.031628453	−1.031628453	−1.031628453
F29	0	Exp-1	0	0	2.48318E−12	1.28927E−09	1.95223E−12
		Exp-2	0	0	0	0	0
		Exp-3	0	0	0	0	0
F30	0	Exp-1	0	0	5.13608E−11	0.000235255	1.073E−09
		Exp-2	0	0	0	3.34607E−09	0
		Exp-3	0	0	0	1.05101E−15	0
F31	−186.73	Exp-1	−103.533756	−123.445392	−186.7273973	−186.7308492	−186.7309088
		Exp-2	−186.7309085	−123.5767709	−186.7309086	−186.7309088	−186.7309088
		Exp-3	−176.9865641	−123.5767709	−186.7309088	−186.7309088	−186.7309088
F32	3	Exp-1	6.822499574	30.0170927	3	3.003161304	3
		Exp-2	3.000001293	30	3	3.000304997	3
		Exp-3	3.000000669	30	3	3.000000076	3
F33	0.00031	Exp-1	0.033032546	0.00216132	0.001090008	0.001414731	0.000931416
		Exp-2	0.010334262	0.000437024	0.000670146	0.00058389	0.000771504
		Exp-3	0.009121193	0.00031966	0.000374177	0.000406602	0.000307486
F34	−10.15	Exp-1	−0.732013706	−5.055195641	−9.896441311	−10.14567432	−7.975216386
		Exp-2	−5.225705315	−5.055197729	−10.10667432	−10.15319968	−10.15319968
		Exp-3	−5.326989674	−5.055197729	−10.15319968	−10.15319968	−10.15319968
F35	−10.4	Exp-1	−1.220313721	−5.087666505	−10.4029402	−10.39761973	−8.917279506
		Exp-2	−4.755328631	−5.087671825	−10.40294057	−10.40293573	−10.05133272
		Exp-3	−4.886948793	−5.087671825	−10.40294057	−10.40294057	−10.40294057
F36	−10.53	Exp-1	−1.43669402	−5.12847104	−10.31752746	−10.30514872	−9.39555491
		Exp-2	−4.979281676	−5.128480787	−10.53640982	−10.53542879	−10.53640982
		Exp-3	−4.787259047	−5.128480787	−10.53640982	−10.53640982	−10.53640982
F37	0	Exp-1	98.83521851	30.09775949	0.570092614	0.705770381	1.326897344
		Exp-2	15.86484077	3.923713434	0.04660114	0.169743099	0.026398136
		Exp-3	8.293007379	0.295334941	0.002107886	0.043522003	0.002401675
F38	0	Exp-1	18.06593955	0	0.0341867	0.070928502	0.027452781
		Exp-2	1.161031853	0	0.000974964	0.022145972	0.000242164
		Exp-3	0.196493545	0	0.000144605	0.003404818	0.00012477
F39	−3.86	Exp-1	−3.525357459	−3.812405944	−3.862782105	−3.862782147	−3.862781719
		Exp-2	−3.859529942	−3.862782148	−3.862782148	−3.862782148	−3.862782148
		Exp-3	−3.860088635	−3.862782148	−3.862782148	−3.862782148	−3.862782148

Table 5 (*continued*)

No.	Min.		SA	PS	PSO2011	ABC	VS
F40	−3.32	Exp-1	−2.025767543	−2.275890604	−3.306796203	−3.322329207	−3.253836322
		Exp-2	−3.21392372	−3.203161892	−3.312304129	−3.322368011	−3.254815564
		Exp-3	−3.196408153	−3.203161918	−3.318394475	−3.322368011	−3.262764965
F41	0	Exp-1	0	0	1.174905607	13.76813973	2.586458043
		Exp-2	0	0	0.0064782	0.002035399	0.008888174
		Exp-3	0	0	0.00476161	0	0.020667864
F42	0	Exp-1	8.88178E−16	8.88178E−16	2.822882317	13.57612974	5.741707238
		Exp-2	8.88178E−16	8.88178E−16	0.594499875	1.13074E−05	0.661994795
		Exp-3	8.88178E−16	8.88178E−16	0.471201803	3.25073E−14	2.63493E−14
F43	0	Exp-1	1.668971097	1.668971097	3.055864107	5.641956224	67.26147149
		Exp-2	1.668971097	0.831213056	0.031100039	3.59957E−12	29.80917701
		Exp-3	1.668971097	0	0.041467608	4.17011E−16	7.698703865
F44	0	Exp-1	3	3	2.777568301	10.60775637	93.32996747
		Exp-2	3	1.9	0.003383422	1.18784E−11	3.77628E−09
		Exp-3	3	0	0	4.39646E−16	0
F45	−1.08	Exp-1	−0.773794902	−1.079704294	−1.080938442	−1.079661136	−1.080938442
		Exp-2	−1.04129998	−1.080938442	−1.080938442	−1.080937366	−1.080938442
		Exp-3	−1.056610147	−1.080938442	−1.080938442	−1.080938442	−1.080938442
F46	−1.5	Exp-1	−0.096506449	−0.29625687	−1.403550728	−0.946054327	−1.120088196
		Exp-2	−0.277626581	−0.303738989	−1.481074867	−1.460442332	−1.499999223
		Exp-3	−0.302885888	−0.303738989	−1.478771147	−1.499999069	−1.480265916
F47	NA	Exp-1	−4.99714E−05	−3.43953E−06	−0.593200476	−0.337366551	−0.492070745
		Exp-2	−0.007342278	−0.422042106	−0.934146078	−0.669342735	−0.457840889
		Exp-3	−0.00129138	−0.422042106	−1.047012352	−0.850070407	−0.799618675
F48	0	Exp-1	121.3553309	0.000387005	2.79024E−07	6.72588E−09	2.05801E−12
		Exp-2	0.000341217	0	0	0	0
		Exp-3	47.0197343	0	0	0	0
F49	0	Exp-1	5214.41377	2299.12496	33.75848613	9.826333934	229.1060221
		Exp-2	1390.299019	0.046676938	16.5584008	0.741561448	0.170804975
		Exp-3	798.3016669	0	9.822291507	0.043261128	0.002021546
F50	0	Exp-1	69386.43767	87961.08336	3322.708051	1386.230197	800.1367562
		Exp-2	14395.78759	1460.867978	1266.44514	53.81347745	60.58560925
		Exp-3	11286.43357	109.8065763	1057.962044	6.196070433	4.746764164

the ABC algorithm fails to exceed the 10^{-16} limit that is used during the pair-wise statistical comparison. Once the algorithm converges to a near optimal point, the inverse incomplete gamma function provides excellent local search ability to the VS algorithm, which helps the algorithm to further improve the solution. The PSO2011 algorithm exhibits similar performance to the proposed VS algorithm. Thus, for 30 functions, the null hypothesis is accepted in the pair-wise comparison of the two algorithms. Due to the results obtained from the pair-wise statistical test, PSO2011 performs better, but the VS algorithm is highly competitive.

In Table 4, a problem-based (US, UN, MS and MN) comparison of the algorithms is also provided. Each cell in the Table 4 shows the total count of the three statistical significance cases (+/=/−) in the pair-wise comparison obtained from Table 3. From Table 4, it can be shown that, for MN (multimodal non-separable) functions, the proposed VS algorithm performs better than the others. For UN (unimodal non-separable) functions the proposed VS algorithm also performs better than the SA, PS, and ABC algorithms. Thus, from these results it can be inferred that the VS algorithm performs better for non-separable functions. For MS (multimodal separable) functions the proposed VS algorithm outperforms the single-solution based algorithms, but population-based algorithms are superior to the VS algorithm for this case. There is no strict difference between the results obtained for the US (unimodal separable) functions. In fact, the provided number of functions for US problems is not sufficient to make a statistically convincing inference.

In Fig. 10, the average computational time of 30 runs for 500,000 iterations is provided for the VS, PSO2011 and ABC algorithms. Because the proposed VS algorithm is quite simple, it is computationally inexpensive compared to the population-based methods. A comparison of the SA and PS algorithms could not be provided due to the limitations of MATLAB® Global Optimization toolbox; however, because these algorithms are also single-solution based metaheuristics, the computational time for these algorithms is expected to be similar to that of the VS algorithm.

3.3.2. Convergence behavior of the algorithms

In the second set of experiments, convergence behaviors of the algorithms are studied. For this purpose, an iteration based comparison of the algorithms (for 100, 1000 and 10,000 iterations) is performed. It is shown that, for most of the functions, algorithms generally tend to converge to their optimum for 10,000 iterations. Therefore, to capture the

convergence behavior of the algorithms, iteration numbers smaller than 10,000 are used. In Table 5, the average results of 30 runs for 100, 1000, and 10,000 iterations are given for the convergence analysis of the algorithms.

From Table 5, it can be shown that the proposed VS algorithm is quite competitive and performs better than the SA, PS and ABC algorithms, while again being competitive with the PSO2011 algorithm. For a number of functions, the PS algorithm surprisingly converged to the global minimum earlier than the other algorithms; however, for most of the functions, it became trapped in the local minima. The ABC algorithm usually requires an additional number of iterations to converge to the optimum point. The reason for the choice of 500,000 iterations in [28] is now more meaningful. The SA algorithm failed to converge to the optimum point for most of the functions as expected because, as the number of iterations increased, the probability of finding a good point also increased for the SA algorithm.

As mentioned previously, a time-based comparison was not possible to perform. However, the time-based statistics obtained in the previous experiments (Fig. 10) implicitly provides information for a time-based comparison after performing an iteration-based convergence analysis. Because the proposed VS algorithm is quite rapid compared to the population-based algorithms, the VS algorithm can perform much more iterations than the population-based ones during a certain period of time.

4. Conclusion

This paper introduced the VS algorithm, which is a single-solution based metaheuristic proposed for the bound-constraint numerical function optimization problems. The VS algorithm utilizes an adaptive step-size adjustment scheme that helps to balance the explorative and exploitative behavior of the search. The algorithm is quite simple and does not require any additional parameters.

The VS algorithm is tested over a large set of 50 benchmark functions that comprises unimodal, multimodal, separable and non-separable problems of different dimensions. The results are compared to the both single-solution based metaheuristics (SA and PS) and population-based metaheuristics (PSO2011 and ABC); the results revealed that besides its simplicity, the proposed VS algorithm is also highly competitive when compared to the performance of the other algorithms. Because the proposed algorithm is quite simple, it is also computationally efficient when compared to population-based metaheuristics. These advantages of the proposed VS algorithm make it a good candidate for the solution of real-life optimization problems.

In the future studies, the proposed VS algorithm will be improved to handle constraint optimization problems. The VS algorithm will also be applied to some real-life optimization problems including neural network optimization, analog and digital circuit optimization, and optimum data partitioning by using hard and fuzzy clustering algorithms.

Appendix A

See Tables A.1–A.8.

Table A.1
A parameter of the Fletcher–Powell function.

i	A_{ij}, $j = 1, \ldots, 10$									
1	−79	56	−62	−9	92	48	−22	−34	−39	−40
2	91	−9	−18	−59	99	−45	88	−14	−29	26
3	−38	8	−12	−73	40	26	−64	29	−82	−32
4	−78	−18	−49	65	66	−40	88	−95	−57	10
5	−1	−43	93	−18	−76	−68	−42	22	46	−14
6	34	−96	26	−56	−36	−85	−62	13	93	78
7	52	−46	−69	99	−47	−72	−11	55	−55	91
8	81	47	35	55	67	−13	33	14	83	−42
9	5	−43	−45	46	56	−94	−62	52	66	55
10	−50	66	−47	−75	89	−16	82	6	−85	−62

Table A.2
B parameter of the Fletcher–Powell function.

i	B_{ij}, $j = 1, \ldots, 10$									
1	−65	−11	76	78	30	93	−86	−99	−37	52
2	59	67	49	−45	52	−33	−34	29	−39	−80
3	21	−23	−80	86	86	−30	39	−73	−91	5
4	−91	−75	20	−64	−15	17	−89	36	−49	−2
5	−79	99	−31	−8	−67	−72	−43	−55	76	−57
6	−89	−35	−55	75	15	−6	−53	−56	−96	87
7	−76	45	74	12	−12	−69	2	71	75	−60
8	−50	−88	93	68	10	−13	84	−21	65	14
9	−23	−95	99	62	−37	96	27	69	−64	−92
10	−5	−57	−30	−6	−96	75	25	−6	96	77

Table A.3
α parameter of the Fletcher–Powell function.

α_j, $j = 1, \ldots, 10$
−2.7910
2.5623
−1.0429
0.5097
−2.8096
1.1883
2.0771
−2.9926
0.0715
0.4142

Table A.4
A parameter of the FoxHoles function.

j	a_{ij}, $i = 1, 2$	
1	−32	−32
2	−16	−32
3	0	−32
4	16	−32
5	32	−32
6	−32	−16
7	−16	−16
8	0	−16
9	16	−16
10	32	−16
11	−32	0
12	−16	0
13	0	0
14	16	0
15	32	0
16	−32	16
17	−16	16
18	0	16
19	16	16
20	32	16
21	−32	32
22	−16	32
23	0	32
24	16	32
25	32	32

Table A.5
a and b parameters of the Kowalik function.

i	a_i	b_i^{-1}
1	0.1957	0.25
2	0.1947	0.5
3	0.1735	1
4	0.1600	2
5	0.0844	4
6	0.0627	6
7	0.0456	8
8	0.0342	10
9	0.0323	12
10	0.0235	14
11	0.0246	16

Table A.6
a and c parameters of the Shekel functions.

i	a_{ij}, $j = 1, \ldots, 4$				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

Table A.7
a, c and p parameters of the 3-parameter Hartman function.

i	a_{ij}, $j = 1, 2, 3$			c_i	p_{ij}, $j = 1, 2, 3$		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.03815	0.5743	0.8828

Table A.8
a, c and p parameters of the 6-parameter Hartman function.

i	a_{ij}, $j = 1, \ldots, 6$						c_i	p_{ij}, $j = 1, \ldots, 6$					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	.0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1415	0.3522	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

References

[1] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Inf. Sci. 192 (2012) 120–142.
[2] G. Allasia, R. Besenghi, Numerical calculation of incomplete gamma function by the trapezoidal rule, Numer. Math. (Numerische Mathematik) 50 (4) (1987) 419–428.
[3] A. Alsheddy, Empowerment scheduling: a multi-objective optimization approach using guided local search, Ph.D. Thesis, School of Computer Science and Electronic Engineering, University of Essex, 2011.
[4] L.C. Andrews, Special Functions of Mathematics for Engineers, SPIE Press, 1992.
[5] B. Basturk, D. Karaboga, An artificial bee colony (abc) algorithm for numeric function optimization, in: IEEE Swarm Intelligence Symposium 2006, Indianapolis, Indiana, USA, May 2006.
[6] Z. Beheshti, S.M.Hj. Shamsuddin, A review of population-based meta-heuristic algorithms, Int. J. Advance. Soft. Comput. Appl. 5 (1) (2013).
[7] M. Birattari, L. Paquete, T. Stützle, K. Varrentrap, Classification of Metaheuristics and Design of Experiments for the Analysis of Components, Technical Report AIDA-2001-05. Intellektik, Technische Universität Darmstadt, Germany, 2001.
[8] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inf. Sci. 237 (2013) 82–117. ISSN 0020-0255.
[9] V. Černý, Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm, J. Optim. Theory Appl. 45 (1985) 41–51.
[10] S.M. Chen, A. Sarosh, Y.F. Dong, Simulated annealing based artificial bee colony algorithm for global numerical optimization, Appl. Math. Comput. 219 (8) (2012) 3575–3589. ISSN 0096-3003.
[11] P. Civicioglu, Backtracking search optimization algorithm for numerical optimization problems, Appl. Math. Comput. 219 (15) (2013) 8121–8144. ISSN 0096-3003.
[12] M. Dorigo, Optimization, Learning and Natural Algorithms, Ph.D. thesis, Politecnico di Milano, Italie, 1992.
[13] H.E. Espitia, J.I. Sofrony, Vortex particle swarm optimization, in: 2013 IEEE Congress on Evolutionary Computation (CEC), 20–23 June 2013, pp. 1992–1998, doi:http://dx.doi.org/10.1109/CEC.2013.6557803.
[14] K. Ganapathy, V. Vaidehi, B. Kannan, H. Murugan, Hierarchical particle swarm optimization with ortho-cyclic circles, Expert Syst. Appl. 41 (7) (2014) 3460–3476. ISSN 0957-4174.
[15] W. Gao, S. Liu, L. Huang, Enhancing artificial bee colony algorithm using more information-based search equations, Inf. Sci. 270 (2014) 112–133. ISSN 0020-0255.
[16] W. Gautschi, A note on the recursive calculation of incomplete gamma functions, ACM Trans. Math. Software 25 (1) (1999) 101–107.
[17] S. Ghosh, S. Das, S. Roy, S.K. Minhazul Islam, P.N. Suganthan, A differential covariance matrix adaptation evolutionary algorithm for real parameter optimization, Inf. Sci. 182 (1) (2012) 199–219.
[18] F. Glover, Future paths for integer programming and links to artificial intelligence, Comp. Operat. Res. 13 (5) (1986) 533–549.
[19] M. Han, C. Liu, J. Xing, An evolutionary membrane algorithm for global numerical optimization problems, Inf. Sci. 276 (2014) 219–241.
[20] P. Hansen, N. Mladenovic, J.A.M. Perez, Variable neighbourhood search: methods and applications, Ann. Oper. Res. 175 (2010) 367–407.
[21] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
[22] R. Hooke, T.A. Jeeves, "Direct search" solution of numerical and statistical problems, J. Assoc. Comput. Mach. (ACM) 8 (2) (1961) 212–229.

[23] H. Huang, H. Qin, Z. Hao, Andrew Lim, Example-based learning particle swarm optimization for continuous optimization, Inf. Sci. 182 (2012) 125–138.
[24] A.D. Junior, R.S. Silva, K.C. Mundim, L.E. Dardenne, Performance and parameterization of the algorithm simplified generalized simulated annealing, Genet. Mol. Biol. 27 (4) (2004) 616–622.
[25] D. Karaboga, An Idea Based on Honeybee Swarm for Numerical Optimization, Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
[26] D. Karaboga, B. Basturk, A. powerful, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm, J. Global Optim. 39 (3) (2007) 459–471.
[27] D. Karaboga, B. Basturk, On the performance of artificial bee colony (abc) algorithm, Appl. Soft Comput. 8 (1) (2008) 687–697.
[28] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Appl. Math. Comput. 214 (1) (2009) 108–132. ISSN 0096-3003, http://dx.doi.org/10.1016/j.amc.2009.03.090.
[29] J. Kennedy, R.C. Eberhart, in: Particle swarm optimization, in: 1995 IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.
[30] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[31] P. Larrañaga, J.A. Lozano (Eds.), Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer Academic Publishers, Boston, 2002.
[32] A. LaTorre, S. Muelas, J. Pena, A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test, Soft. Comput. 15 (2011) 2187–2199.
[33] H.R. Lourenço, O. Martin, T. Stützle, Iterated Local Search: Framework and Applications, Handbook of Metaheuristics, second ed., vol. 146, Kluwer Academic Publishers, 2010, pp. 363–397. International Series in Operations Research & Management Science.
[34] F. Neri, E. Mininno, G. Iacca, Compact particle swarm optimization, Inf. Sci. 239 (2013) 96–121.
[35] M.G.H. Omran, M. Clerc, 2011 <http://www.particleswarm.info/> (accessed 03.12.13).
[36] L.A. Rastrigin, The convergence of the random search method in the extremal control of a many parameter system, Autom. Rem. Control 24 (10) (1963) 1337–1342.
[37] G. Schrack, M. Choit, Optimized relative step size random searches, Math. Program. 10 (1) (1976) 230–244.
[38] M.A. Schumer, K. Steiglitz, Adaptive step size random search, IEEE Trans. Autom. Control 13 (3) (1968) 270–276.
[39] R. Storn, K. Price, Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces, Technical report, International Computer Science Institute, Berkley, 1995.
[40] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[41] E.-G. Talbi, Metaheuristics: From Design to Implementation, Wiley & Sons, Hoboken, New Jersey, 2009.
[42] A.I.F. Vaz, L.N. Vicente, PSwarm: a hybrid solver for linearly constrained global derivative-free optimization, Optim. Meth. Softw. 24 (4-5) (2009) 669–685.
[43] L. Wang, B. Yang, Y. Chen, Improving particle swarm optimization using multi-layer searching strategy, Inf. Sci. 274 (2014) 70–94.
[44] S. Winitzki, Computing the incomplete gamma function to arbitrary precision computational science and its applications – ICCSA 2003, LNCS, vol. 2667, Springer-Verlag, Berlin, 2003, pp. 790–798.
[45] X.-S. Yang, Z. Cui, R. Xiao, A.H. Gandomi, M. Karamanoglu, Swarm Intelligence and Bio-Inspired Computation, Elsevier, Waltham, MA, 2013.
[46] ABC Algorithm <http://mf.erciyes.edu.tr/abc/> (accessed 03.12.13).
[47] Vortex Search Algorithm <http://web.itu.edu.tr/~bdogan/VortexSearch/VS.htm> (accessed 03.12.13).
[48] Standard Particle Swarm Optimization, Particle Swarm Central, Tech. Rep., 2012 <http://clerc.maurice.free.fr/pso/SPSO_descriptions.pdf> (accessed 17.04.14).