

MODIFIED VORTEX SEARCH ALGORITHM FOR REAL PARAMETER OPTIMIZATION

Berat Doğan

Department of Biomedical Engineering, Inonu University, Malatya, Turkey
berat.dogan@inonu.edu.tr

ABSTRACT

The Vortex Search (VS) algorithm is one of the recently proposed metaheuristic algorithms which was inspired from the vortical flow of the stirred fluids. Although the VS algorithm is shown to be a good candidate for the solution of certain optimization problems, it also has some drawbacks. In the VS algorithm, candidate solutions are generated around the current best solution by using a Gaussian distribution at each iteration pass. This provides simplicity to the algorithm but it also leads to some problems along. Especially, for the functions those have a number of local minimum points, to select a single point to generate candidate solutions leads the algorithm to being trapped into a local minimum point. Due to the adaptive step-size adjustment scheme used in the VS algorithm, the locality of the created candidate solutions is increased at each iteration pass. Therefore, if the algorithm cannot escape a local point as quickly as possible, it becomes much more difficult for the algorithm to escape from that point in the latter iterations. In this study, a modified Vortex Search algorithm (MVS) is proposed to overcome above mentioned drawback of the existing VS algorithm. In the MVS algorithm, the candidate solutions are generated around a number of points at each iteration pass. Computational results showed that with the help of this modification the global search ability of the existing VS algorithm is improved and the MVS algorithm outperformed the existing VS algorithm, PSO2011 and ABC algorithms for the benchmark numerical function set.

KEYWORDS

Metaheuristics, Numerical Function Optimization, Vortex Search Algorithm, Modified Vortex Search Algorithm.

1. INTRODUCTION

In the past two decades, a number of metaheuristic algorithms have been proposed to solve complex real-world optimization problems. Most of these algorithms are nature inspired methods and therefore mimic natural metaphors such as, evolution of species (GA [1] and DE [2-3]), annealing process (SA [4-5]), ant behaviour (ACO [6]), swarm behaviour (PSO [7] and ABC [8-9]) etc. These algorithms make few or no assumptions for the problem at hand and provide fast and robust solutions. Although, the solutions provided by metaheuristics may not be optimal solutions, they are highly preferred because of their simplicity and flexibility.

Despite the high number of available metaheuristics, developing new metaheuristic algorithms is still an active research area. In [10-15], a number of recently proposed metaheuristics can be Jan Zizka et al. (Eds) : CCSEIT, AIAP, DMDB, MoWiN, CoSIT, CRIS, SIGL, ICBB, CNSA-2016 pp. 113–126, 2016. © CS & IT-CSCP 2016 DOI : 10.5121/csit.2016.60610

found. All of these metaheuristics have certain characteristics and thus each one may be more successful on a certain optimization problem when compared to the others. The Vortex Search (VS) algorithm [16] is one of these recently proposed metaheuristic algorithms which was inspired from the vortical flow of the stirred fluids. The search behaviour of the VS algorithm is modelled as a vortex pattern by using an adaptive step-size adjustment scheme. By this way, it is aimed to have a good balance between the explorative and exploitative behaviour of the search. The proposed VS algorithm was tested over 50 benchmark mathematical functions and the obtained results compared to 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 was performed to measure the pairwise statistical performances of the algorithms, the results of which indicated that the proposed VS algorithm outperforms the SA, PS and ABC algorithms while being competitive with the PSO2011 algorithm. Because of the simplicity of the proposed VS algorithm, a significant decrease in the computational time of the 50 benchmark numerical functions was also achieved when compared to the population-based algorithms. In some other studies [17-20], the VS algorithm has also been successfully used for the solution of some real-world optimization problems.

Although the proposed VS algorithm is a good candidate for the solution of optimization problems, it also has some drawbacks. In the VS algorithm, candidate solutions are generated around the current best solution by using a Gaussian distribution at each iteration pass. This provides simplicity to the algorithm but it also leads to some problems along. Especially, for the functions those have a number of local minimum points, to select a single point to generate candidate solutions leads the algorithm to being trapped into a local minimum point. Due to the adaptive step-size adjustment scheme used in the VS algorithm, the locality of the created candidate solutions is increased at each iteration pass. Therefore, if the algorithm cannot escape a local point as quickly as possible, it becomes much more difficult for the algorithm to escape from that point in the latter iterations.

In this study, a modified Vortex Search algorithm (MVS) is proposed to overcome above mentioned drawback of the existing VS algorithm. In the MVS algorithm, the candidate solutions are generated around different points at each iteration pass. These points are iteratively updated during the search process, details of which are given in the following section. The MVS algorithm is tested with 7 benchmark functions that was used earlier in [16]. These 7 functions are selected from the benchmark set of 50 functions for which the VS algorithm trapped into the local minimum points. Because the SA and PS algorithms showed poor performances in [16], in this study these two algorithms are excluded and the results are compared to the results those obtained by the VS algorithm, PSO2011 and ABC algorithms. It is shown that, the MVS algorithm outperforms all of these algorithms and can successfully escape from the local minimum points of the functions that the VS algorithm was being trapped earlier.

The remaining part of this paper is organized as follows. In the following section, first a brief description of the VS algorithm is given. Then, the modification performed on the VS algorithm is detailed and the MVS algorithm is introduced. Section 3 covers the experimental results and discussion. Finally, Section 4 concludes the work.

2. METHODOLOGY

2.1. A Brief Description of the Vortex Search Algorithm

Let us consider a two-dimensional optimization problem. In a two dimensional space a vortex pattern can be modelled 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 can be calculated using Eq. 1.

$$\mu_0 = \frac{\text{upperlimit} + \text{lowerlimit}}{2} \quad (1)$$

In Eq.1, *upperlimit* and *lowerlimit* are $d \times 1$ vectors that define the bound constraints of the problem in d dimensional space. Then, 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, $C_0(s) = \{s_1, s_2, \dots, s_k\}$ $k=1, 2, \dots, n$ 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.

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

In Eq.2, 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.

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

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.

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

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.

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.

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

In Eq.5, $k = 1, 2, \dots, n$ and $i = 1, 2, \dots, d$ and $rand$ is a uniformly distributed random number. Next, the memorized best solution s^i 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 $s^i \in C_1(s)$. 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 Figure 1. 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 Figure 2 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 Figure 3.

The radius decrement process given in Figure 3 can be considered as a type of adaptive step-size adjustment process which has critical importance on the performance of the VS algorithm. This process should be performed in such a way that allows the 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. In the VS algorithm, the inverse incomplete gamma function is used to decrease the value of the radius during each iteration pass.

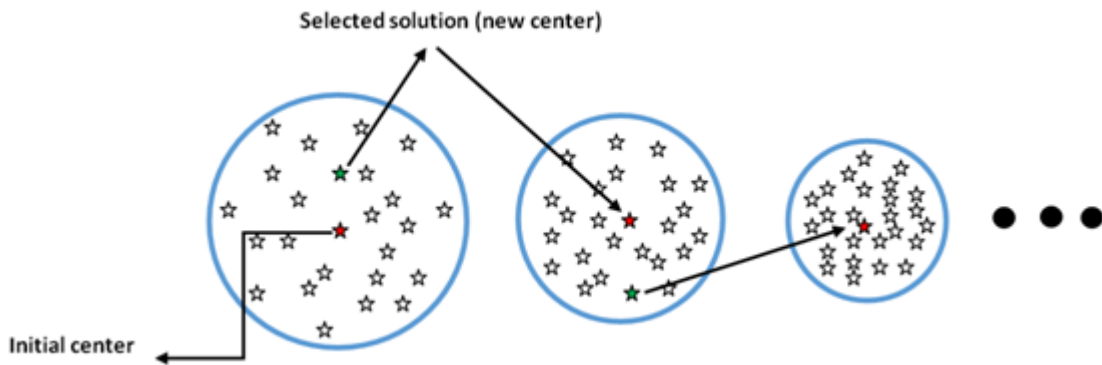


Figure 1. An illustrative sketch of the search process

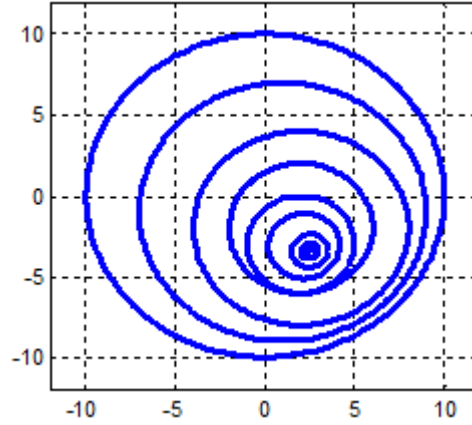


Figure 2. A representative pattern showing the search boundaries (circles) of the VS algorithm after a search process, which has a vortex-like structure.

The incomplete gamma function given in Eq. 6 most commonly arises in probability theory, particularly in those applications involving the chi-square distribution [21].

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

In Eq.6, $a > 0$ is known as the shape parameter and $x \geq 0$ is a random variable. In conjunction with the incomplete gamma function, its complementary $\Gamma(x, a)$ is usually also introduced (Eq. 7).

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

Thus, it follows that,

$$\gamma(x, a) + \Gamma(x, a) = \Gamma(a) \quad (8)$$

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 [22-24]. MATLAB® also provides some tools for the calculation of the 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 Figure 4, 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

$$a_t = a_0 - \frac{t}{\text{MaxItr}} \quad (9)$$

```

Inputs: Initial center  $\mu_0$  is calculated by using Eq. 1
           Initial radius  $r_0$  (or the standard deviation,  $\sigma_0$ ) is computed by using Eq. 10
           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  $\mu_t$ 
  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  $\mu_t$  */
   $s' = \text{Select}(C_t(s))$ ;
  if  $f(s') < f(s_{best})$ 
     $s_{best} = s'$ 
     $f(s_{best}) = f(s')$ 
  else
    keep the best solution found 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 */
   $r_{t+1} = \text{Decrease}(r_t)$ 
   $t = t + 1$ ;
Until the maximum number of iterations is reached
Output: Best solution found so far  $s_{best}$ 

```

Figure 3. A description of the VS algorithm

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.

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 $(1/x) \cdot \text{gammaincinv}(x, a_0) \approx 1$, which means $r_0 \approx \sigma_0$ as indicated before.

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

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.

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

Here, t represents the iteration index.

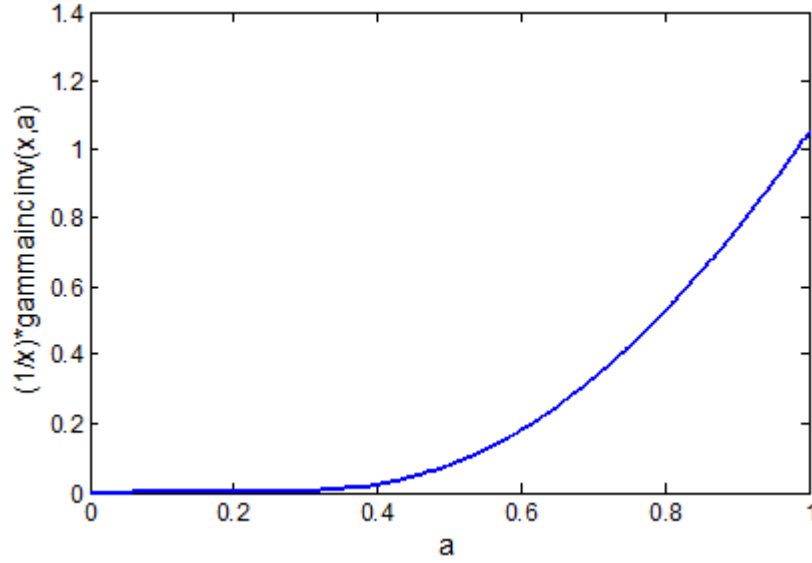


Figure 4. $(1/x) \cdot \text{gammaincinv}(x, a)$ where $x = 0.1$ and $a \in [0, 1]$

2.2. The Modified Vortex Search Algorithm

The VS algorithm creates candidate solutions around a single point at each iteration pass. At the first iteration, this point is the initial center μ_0 which is determined with the upper and lower limits of the problem at hand while in the latter iterations the center is shifted to the current best position found so far. As mentioned before, this mechanism leads the VS algorithm to being trapped into local minimum points for a number of functions.

To overcome above mentioned drawback, in this study a modified VS algorithm (MVS) is proposed. In the MVS algorithm, candidate solutions are generated around multiple centers at each iterations pass. The search behavior of the MVS algorithm can be thought as a number of parallel vortices that have different centers at each iteration pass. Initially, the centers of these multiple vortices are selected as in the VS algorithm. Let us consider, the total number of centers (or vortices) to be represented by m . Let us say, $M_t(\mu)$ represents the matrix that stores the values of these m centers at each iteration pass and t represents the iteration index. Thus, initially $M_0(\mu) = \{\mu_0^1, \mu_0^2, \dots, \mu_0^l\}$, $l = 1, 2, \dots, m$ and initial positions of these centers are computed as in Eq. 12.

$$\mu_0^1 = \mu_0^2 = \dots = \mu_0^l = \frac{\text{upperlimit} + \text{lowerlimit}}{2}, \quad l = 1, 2, \dots, m \quad (12)$$

Next, a number of candidate solutions are generated with a Gaussian distribution around these initial centers by using the initial radius value r_0 . In this case the total number of candidate solutions is again selected to be n . But note that, these n solutions are generated around m centers. Thus, one should select n/m solutions around each center.

Let us say, $CS_t^l(s) = \{s_1, s_2, \dots, s_k\}$ $k = 1, 2, \dots, n/m$ represents the subset of solutions generated around the center $l = 1, 2, \dots, m$ for the iteration t . Then, the total solution set generated for the

iteration $t=0$ can be represented by $C_0(s) = \{CS_0^1, CS_0^2, \dots, CS_0^l\}$, $l=1,2,\dots,m$. In the selection phase, for each subset of solutions, a solution (which is the best one) $s'_l \in CS_0^l(s)$ is selected. Prior to the selection phase it must be ensured that the candidate subsets of solutions are inside the search boundaries. For this purpose, the solutions that exceed the boundaries are shifted into the boundaries, as in Eq. 5. Let us say, the best solution of each subset is stored in a matrix $PBest_t(s')$ at each iteration pass. Thus, for $t=0$, $PBest_0(s') = \{s'_1, s'_2, \dots, s'_l\}$, $l=1,2,\dots,m$. Note that, the best solution of this matrix ($PBest_0(s')$) is also the best solution of the total candidate solution set $C_0(s)$ for the current iteration, which is represented as Itr_{best} .

In the VS algorithm, at each iterations pass, the center is always shifted to the best solution found so far, s_{best} . However, in the MVS algorithm, there exist m centers which positions need to be updated for the next iteration. The most important difference between the VS and MVS algorithm arises from here. In the MVS algorithm, one of these centers is again shifted to the best solution found so far, s_{best} . But, the remaining $m-1$ centers are shifted to a new position determined by the best positions generated around the each center at the iteration t and the best position found so far, s_{best} as shown in Eq. 13.

$$\mu_t^l = s'_l + rand \cdot (s'_l + s_{best}) \quad (13)$$

In Eq. 13, $rand$ is a uniformly distributed random number, $l=1,2,\dots,m-1$ and $s'_l \in PBest_{t-1}(s')$. Thus, for $t=1$, $M_1(\mu) = \{\mu_1^1, \mu_1^2, \dots, \mu_1^l\}$, $l=1,2,\dots,m-1$ is determined by using the $s'_l \in PBest_0(s')$ positions and the best position found so far, s_{best} . In Figure-5, an illustrative sketch of the center update process is given for a two-dimensional problem. In Figure 5, only one center is considered.

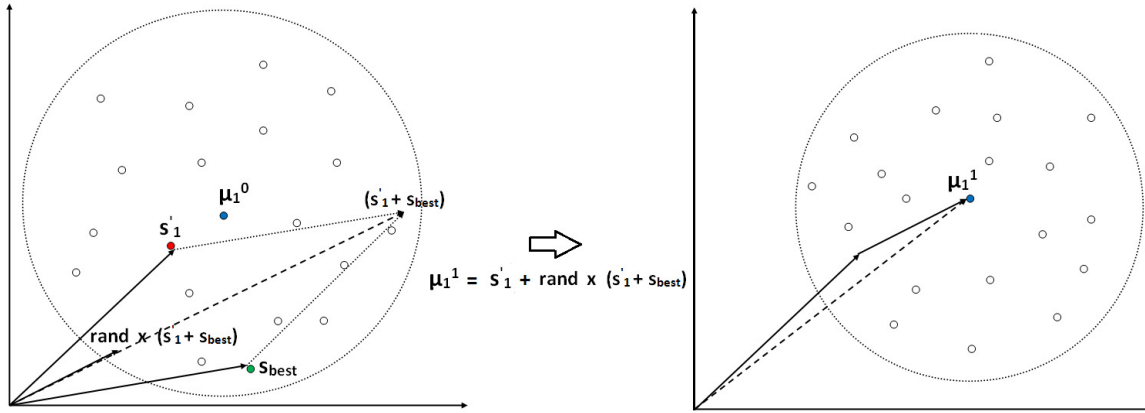


Figure 5. An illustrative sketch of the center updating process for the MVS algorithm (only one center is considered)

In the MVS algorithm, the radius decrement process is held totally in the same way as it is done in the VS algorithm. At each iteration pass, the radius is decreased by utilizing the inverse

incomplete gamma function and thus, the locality of the generated solutions is increased. In Figure 6, a description of the MVS algorithm is provided.

```

Inputs: Initial centers  $M_0(\mu) = \{\mu_0^1, \mu_0^2, \dots, \mu_0^l\}$ ,  $l = 1, 2, \dots, m$  is computed by using Eq. 12
          Initial radius  $r_0$  (or the standard deviation,  $\sigma_0$ ) is computed by using Eq. 10
          Fitness of the best solution found so far  $f(s_{best}) = \inf$ 

 $t = 0$ ;
Repeat
  /* Generate candidate solution sets by using Gaussian distribution around the centers
  with a standard deviation (radius)  $r_t$  */
  Generate( $CS_t^l(s)$ );
  If exceeded, then shift the  $CS_t^l(s)$  values into the boundaries as in Eq.5
  /* Select the best solution from each subset  $CS_t^l(s)$  to update the corresponding centers
   $\mu_t^l$  */
   $s_t^l = \text{Select}(CS_t^l(s))$ ;
  /* Store the best solution of each subset  $CS_t^l(s)$  into the matrix  $PBest_t(s')$  */
   $PBest_t(s') = \text{Store}(s_t^l)$ 
  /* Select the best solution  $Itr_{best}$  from the  $PBest_t(s')$  */
   $Itr_{best} = \text{Select}(PBest_t(s'))$ 
  if  $f(Itr_{best}) < f(s_{best})$ 
     $s_{best} = Itr_{best}$ 
     $f(s_{best}) = f(Itr_{best})$ 
  else
    keep the best solution found so far  $s_{best}$ 
  end

  Shift  $m-1$  centers to their new positions as in Eq. 13
  Shift one of the centers to the best solution found so far,  $s_{best}$ 

  /* Decrease the standard deviation (radius) for the next iteration */
   $r_{t+1} = \text{Decrease}(r_t)$ 
   $t = t + 1$ ;
Until the maximum number of iterations is reached
Output: Best solution found so far  $s_{best}$ 

```

Figure 6. A description of the MVS algorithm

3. RESULTS

The proposed MVS algorithm is tested on 7 benchmark functions for which the VS algorithm was being trapped into a local minimum point. By using these functions, in this study, the

performance of the MVS algorithm is compared to the VS, PSO2011 and ABC algorithms. PSO2011 [25-26] is an extension of the standard PSO algorithm and the ABC algorithm is a well-known optimization algorithm which was inspired from the collective behaviours of honey bees.

The functions used in the experiments are listed in Table 1. For the formulations of the functions listed in Table 1, please refer to the reference [16].

3.1. Algorithm Settings

The ABC and PSO2011 algorithms are selected to have a population size of 50, which is also the number of neighborhood solutions of the proposed VS algorithm. 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 [27]. 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. VS algorithm does not have any additional parameters. Different from the VS algorithm, the MVS algorithm has the parameter m , which represents the total number of centers.

3.2. Experimental Results

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. For the MATLAB® codes of the PSO2011, ABC, VS and MVS algorithms please refer to [25], [28], [29] and [30]. 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 given in Table 2

Table 1. Benchmark function set that is used in the experiments

No	Function	Characteristics	Range	Dim.	Min.
F1	Powell	Unimodal Non-Separable	[-4,5]	24	0
F2	Rosenbrock	Unimodal Non-Separable	[-30, 30]	30	0
F3	Dixon-Price	Unimodal Non-Separable	[-10, 10]	30	0
F4	Rastrigin	Multimodal Separable	[-5.12, 5.12]	30	0
F5	Schwefel	Multimodal Separable	[-500, 500]	30	-12569.5
F6	Griewank	Multimodal Non-Separable	[-600, 600]	30	0
F7	Penalized	Multimodal Non-Separable	[-50, 50]	30	0

As shown in Table 2, for the MVS algorithm two different cases are considered. In the first case, the total number of candidate solutions is selected to be 50, which means 10 candidate solutions are generated around each center for $m=5$. In this case, the MVS algorithm can avoid from the local minimum points of the functions which is not the case for the VS algorithm. However, poor sampling of the search space for this case (10 points around each center) may lead the MVS algorithm to show a correspondingly poor performance on the improvement of the found near optimal solutions (exploitation). Therefore, another case in which the total number of candidate solutions is selected to be 250 is considered for the MVS algorithm. In this case, 50 candidate solutions are generated around each center for $m=5$. As can be shown in Table 2, the MVS algorithm with 250 candidate solutions performs better than the MVS algorithm with 50 candidate solutions.

In [31], authors stated that after a sufficient value for colony size, any increment in the value does not improve the performance of the ABC algorithm significantly. For the test problems carried out in [31] colony sizes of 10, 50 and 100 are used for the ABC algorithm. It is shown that although from 10 to 50 the performance of the ABC algorithm significantly increased, there is not any significant difference between the performances achieved by 50 and 100 colony sizes. Similarly, for the PSO algorithm it is reported that, PSO with different population sizes has almost the similar performance which means the performance of PSO is not sensitive to the population size [32]. Based on the above considerations, in this study a comparison of the MVS algorithm to the ABC and PSO2011 algorithms with a different population size is not performed. For the VS algorithm it is expected to achieve better exploitation ability with an increased number of candidate solutions. But the problem with the VS algorithm is with its global search ability rather than the local search ability for some of the functions listed above. Therefore, a comparison of the MVS ($m = 5, n = 50$) to VS algorithm with 50 candidate solutions is thought to be enough to show the improvement achieved by the modification performed on the VS algorithm.

In Figure 7, average computational time of 30 runs for 500,000 iterations is also provided for the MVS ($m = 5, n = 50$), MVS ($m = 5, n = 250$), VS, PSO2011 and ABC algorithms. As shown in this figure, the required computational time to perform 500,000 iterations with the MVS algorithm is slightly increased when compared to the VS algorithm. However, even for the MVS ($m = 5, n = 250$) algorithm the required computational time to perform 500,000 iterations is still lower than the PSO2011 and ABC algorithms.

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

No	Min.		MVS ($m = 5, n = 50$)	MVS ($m = 5, n = 250$)	VS	PSO2011	ABC
F1	0	Mean	7.59934E-09	3.88377E-10	1.43967E-05	2.04664E-07	9.09913E-05
		StdDev	4.14437E-08	1.27749E-09	2.27742E-06	1.21051E-08	1.42475E-05
		Best	1.1432E-16	0	5.71959E-06	1.72679E-07	5.23427E-05
F2	0	Mean	1.20813E-07	3.51659E-08	0.367860114	0.930212233	0.003535257
		StdDev	2.94163E-07	5.41004E-08	1.130879848	1.714978077	0.003314818
		Best	1.14463E-12	1.85577E-13	9.42587E-05	0	7.08757E-05
F3	0	Mean	0	0	0.666666667	0.666666667	1.91607E-15
		StdDev	0	0	7.68909E-16	4.38309E-16	2.55403E-16
		Best	0	0	0.666666667	0.666666667	1.1447E-15
F4	0	Mean	4.14483E-16	0	57.60799224	26.11016129	0
		StdDev	8.95296E-16	3.24317E-16	13.94980276	5.686650032	0
		Best	0	0	33.82857771	16.91429893	0
F5	0	Mean	-12569.48662	-12569.48662	-11283.05416	-8316.185447	-12569.48662
		StdDev	3.63798E-12	3.02118E-12	352.1869262	463.9606712	1.85009E-12
		Best	-12569.48662	-12569.48662	-11799.62928	-9466.201047	-12569.48662
F6	0	Mean	0	0	0.032798017	0.004761038	0
		StdDev	0	0	0.018570459	0.008047673	0
		Best	0	0	0.00739604	0	0
F7	-1	Mean	0	0	0.114662313	0.024187276	2.63417E-16
		StdDev	0	0	0.532276418	0.080213839	0
		Best	0	0	0	0	1.29727E-16

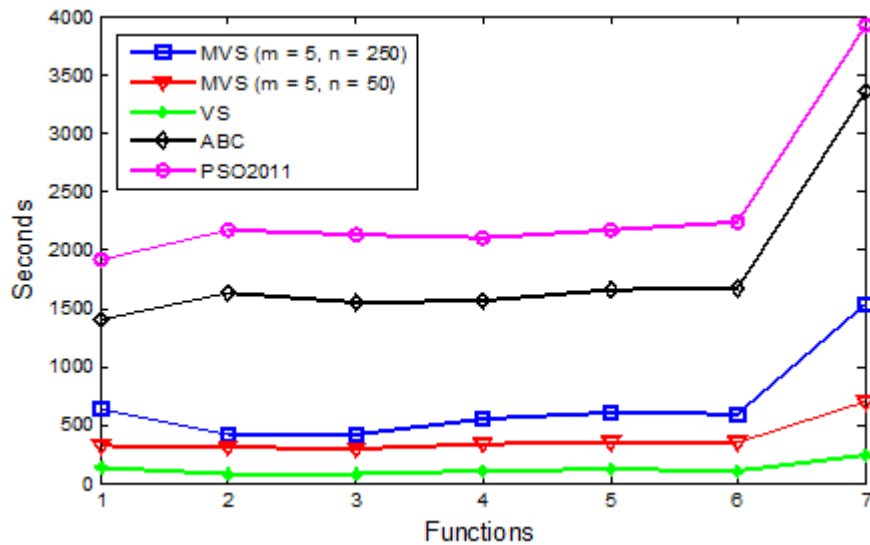


Figure 7. Average computational time of 30 runs for 50 benchmark functions (500,000 iterations)

4. CONCLUSIONS

This paper presents a modified VS algorithm in which the global search ability of the existing VS algorithm is improved. This is achieved by using multiple centers during the candidate solution generation phase of the algorithm at each iteration pass. In the VS algorithm, only one center is used for this purpose and this usually leads the algorithm to being trapped into a local minimum point for some of the benchmark functions. Computational experiments performed on the benchmark functions showed that, the MVS algorithm outperforms the VS, PSO211 and ABC algorithms and can successfully escapes from the local minimum points of the functions that the VS algorithm was being trapped earlier. Although the complexity of the existing VS algorithm is a bit increased with the performed modification, there is not any significant difference between the computational time of the modified VS algorithm and the existing VS algorithm.

In the future studies, the proposed MVS algorithm will be used for the solution of some real world optimization problems such as neural network optimization, optimum data partitioning, and analog circuit parameters optimization.

REFERENCES

- [1] Holland J.H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975
- [2] Storn R., Price K., *Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces*, Technical report, International Computer Science Institute, Berkley, 1995
- [3] Storn R., Price K., *Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces*, *Journal of Global Optimization* 11 (1997) 341–359.
- [4] Kirkpatrick S., Gelatt Jr C.D., Vecchi M.P., *Optimization by Simulated Annealing*, *Science* 220 (4598): 671–680, (1983).

- [5] Černý V., Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm, *Journal of Optimization Theory and Applications*, 45: 41–51, (1985).
- [6] Dorigo M., *Optimization, Learning and Natural Algorithms*, PhD thesis, Politecnico di Milano, Italy, 1992
- [7] Kennedy J., Eberhart R.C., in: *Particle Swarm Optimization*, 1995 IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948
- [8] Karaboga D., *An idea based on honeybee swarm for numerical optimization*, Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
- [9] Karaboga D., Basturk B., A powerful, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm, *Journal of Global Optimization* 39 (3) (2007) 459–471.
- [10] Civicioglu P., Backtracking Search Optimization Algorithm for numerical optimization problems, *Applied Mathematics and Computation*, Volume 219, Issue 15, 1 April 2013, Pages 8121-8144, ISSN 0096-3003
- [11] Kashan A.H., A new metaheuristic for optimization: Optics inspired optimization (OIO), *Computers & Operations Research*, Volume 55, March 2015, Pages 99-125, ISSN 0305-0548
- [12] Yang X.S., *Flower pollination algorithm for global optimization*, *Unconventional computation and natural computation*. Springer Berlin Heidelberg, 2012. 240-249
- [13] Hajipour H., Khormuji H.B., and Rostami H., ODMA: a novel swarm-evolutionary metaheuristic optimizer inspired by open source development model and communities. *Soft Computing* (2014): 1-21.
- [14] Yong L., Peng T., A multi-start central force optimization for global optimization, *Applied Soft Computing*, Volume 27, February 2015, Pages 92-98, ISSN 1568-4946
- [15] Yu-Jun Z., *Water wave optimization: A new nature-inspired metaheuristic*, *Computers & Operations Research*, Volume 55, March 2015, Pages 1-11, ISSN 0305-0548
- [16] Doğan B., Ölmez T., A new metaheuristic for numerical function optimization: Vortex Search algorithm, *Information Sciences*, Volume 293, 1 February 2015, Pages 125-145, ISSN 0020-0255
- [17] Doğan B., Ölmez T., Vortex search algorithm for the analog active filter component selection problem, *AEU - International Journal of Electronics and Communications*, Volume 69, Issue 9, September 2015, Pages 1243-1253, ISSN 1434-8411
- [18] Doğan, B., Yuksel, A., Analog filter group delay optimization using the Vortex Search algorithm, *Signal Processing and Communications Applications Conference (SIU)*, 2015 23th , vol., no., pp.288,291, 16-19 May 2015
- [19] Doğan B., Ölmez T., Modified Off-lattice AB Model for Protein Folding Problem Using the Vortex Search Algorithm, *International Journal of Machine Learning and Computing* vol. 5, no. 4, pp. 329-333, 2015.

- [20] Doğan B., Ölmez T., Fuzzy clustering of ECG beats using a new metaheuristic approach, 2nd International Work-Conference on Bioinformatics and Biomedical Engineering (IWBBIO), 7-9 April 2014, Granada, Spain.
- [21] Andrews L.C., Special Functions of Mathematics for Engineers, SPIE Press, 1992
- [22] Gautschi W., A note on the recursive calculation of incomplete gamma functions ACM Trans. Math. Software, 25 (1) (1999), pp. 101–107
- [23] Winitzki S., Computing the incomplete Gamma function to arbitrary precision Computational Science and Its Applications – ICCSA 2003, of LNCS, Vol. 2667 Springer-Verlag, Berlin (2003), pp. 790–798
- [24] Allasia, G., Besenghi R., Numerical calculation of incomplete gamma function by the trapezoidal rule, Numer. Math. (Numerische Mathematik) 50 (4):419{428, 1987
- [25] Omran M.G.H., Clerc M., 2011, <<http://www.particleswarm.info/>>, accessed 25 February 2016
- [26] Clerc M., "Standard Particle Swarm Optimization," Particle Swarm Central, Tech. Rep., 2012, http://clerc.maurice.free.fr/ps0/SPSO_descriptions.pdf, accessed 25 February 2016
- [27] Karaboga, D., Akay, B., A comparative study of Artificial Bee Colony algorithm, Applied Mathematics and Computation, Volume 214, Issue 1, 1 August 2009, Pages 108-132, ISSN 0096-3003.
- [28] ABC algorithm, <http://mf.erciyes.edu.tr/abc/>, accessed 25 February 2016
- [29] VS algorithm, <http://web.itu.edu.tr/~bdogan/VortexSearch/VS.htm>, accessed 25 February 2016
- [30] MVS algorithm, <http://web.itu.edu.tr/~bdogan/ModifiedVortexSearch/MVS.htm>, accessed 25 February 2016
- [31] Karaboga D., Basturk B., On the performance of artificial bee colony (abc) algorithm, Applied Soft Computing 8 (1) (2008) 687–697.
- [32] Shi, Y., Eberhart R.C., Empirical study of particle swarm optimization, Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on. Vol. 3. IEEE, 1999.

AUTHORS

Dr. Berat Doğan received his BSc. degree in Electronics Engineering from Erciyes University, Turkey, 2006. He received his MSc. degree in Biomedical Engineering from Istanbul Technical University, Turkey, 2009. He received his PhD. in Electronics Engineering at Istanbul Technical University, Turkey, 2015. Between 2008-2009 he worked as a software engineer at Nortel Networks Netas Telecommunication Inc. Then, from 2009 to July 2015 he worked as a Research Assistant at Istanbul Technical University. Now he is working as an Assistant Professor at Inonu University, Malatya, Turkey. His research interests include optimization algorithms, pattern recognition, biomedical signal and image processing, and bioinformatics.

