Applied Mathematics & Information Sciences
An International Journal

Hybrid Seeker Optimization Algorithm for Global Optimization

Milan Tuba[1], *Ivona Brajevic*[2], *Raka Jovanovic*[3]

[1] Faculty of Computer Science, Megatrend University, Belgrade, Serbia
[2] Faculty of Mathematics, University of Belgrade, Serbia
[3] Institute of Physics, University of Belgrade, Pregrevica 118, Zemun, Serbia

Received: 26 Nov. 2012, Revised: 30 Nov. 2012, Accepted: 19 Jan. 2013
Published online: 1 May 2013

Abstract: Swarm intelligence algorithms have been succesfully applied to hard optimization problems. Seeker optimization algorithm is one of the latest members of that class of metaheuristics and it has not yet been thorougly researched. Since the early versions of this algorithm were less succesful with multimodal functions, we propose in this paper hybridization of the seeker optimization algorithm with the well known artificial bee colony (ABC) algorithm. At certain stages we modify seeker's position by search formulas from the ABC algorithm and also modify the inter-subpopulation learning phase by using the binomial crossover operator. Our proposed algorithm was tested on the complete set of 23 well-known benchmark functions. Comparisons show that our proposed algorithm outperforms six state-of-the-art algorithms in terms of the quality of the resulting solutions as well as robustenss on most of the test functions.

Keywords: Seeker optimization algorithm, artificial bee colony, swarm intelligence, unconstrained optimization metaheuristics, nature inspired algorithms.

1 Introduction

Many population-based optimization algorithms have been developed and applied to difficult optimization problems. The success of these methods, which are working on a set of candidate solutions and trying to improve them, depends on their ability to maintain proper balance between exploration and exploitation. The exploitation refers to the ability of the algorithm to apply the knowledge of previously discovered good solutions to better guide the search towards the global optimum. The exploration refers to the ability of the algorithm to investigate the unknown and less promising regions in the search space to avoid being trapped in local optima. A poor balance between these two algorithm's abilities may result in a weak optimization method which suffers from premature convergence or infinite random search.

Genetic algorithm (GA) inspired by Darwin's theory of evolution [1], differential evolution (DE) [2] which is iteratively trying to improve a candidate solution with regard to a given measure of quality, ant colony optimization (ACO) [3] based on ant colony foraging behavior, particle swarm optimization (PSO) [4] inspired by the social behavior of birds or fish, artificial bee colony (ABC) algorithm [5], [6] based on honey bees foraging behavior, and cuckoo search (CS) [7] based on cuckoo bird's behavior, are among the most popular metaheuristics which employ a population of individuals trying to solve the problem. Pure versions of these algorithms were later enhanced to improve the performance in general, or for some class of the problems [8], [9], [10], [11], [12], [13]. Sometimes they are combined and successfully used for wide range of problems [14], [15], [16], [17].

Swarm intelligence based algorithms represent an important class of population-based optimization algorithms, where ABC algorithm is one of the most popular. Karaboga has introduced an artificial bee colony (ABC) algorithm for numerical optimization problems [5]. The performance of the ABC algorithm was tested for optimization of multivariable unconstrained functions and the results were compared with GA, PSO and particle

This research was supported by Ministry of education and science of Republic of Serbia, Grant III-44006.

* Corresponding author e-mail: tuba@ieee.org

swarm inspired evolutionary algorithm (PS-EA) [18]. The results showed that ABC outperforms the other algorithms. Since that time, ABC has been modified by many researchers and has been applied to solve several numerical optimization problems [19], [20], [21], [22]. There are object-oriented software implementations [23], as well as parallelized versions [24] for unconstrained optimization problems of the ABC metaheuristic.

Seeker optimization algorithm (SOA), based on simulating the act of human searching, is a novel search algorithm for unconstrained optimization problems [25]. SOA was analyzed with a challenging set of benchmark problems, where its performance was compared to the performance of DE and three modified versions of PSO algorithms [26]. SOA has shown better global search ability and faster convergence speed for most of the chosen benchmark problems, especially for unimodal benchmarks. For multimodal test functions the results were not very satisfactory because it has been noticed that for that type of problems algorithm may easily be trapped at some local optimum. Since its invention, SOA has been successfully applied to different problems. In [27] the application of the SOA to tuning the structures and parameters of artificial neural networks is presented, while in [28] SOA-based evolutionary method is proposed for digital IIR filter design. Also, a new optimized model of proton exchange membrane fuel cell (PEMFC) was proposed by using SOA [29]. Recently, SOA was renamed to Human Group Optimization algorithm or Human Group Optimizer (HGO), because the term "seeker optimization algorithm" could not reflect the essential nature of the novel algorithm of simulating human behaviors [30].

In this paper, we propose a hybrid algorithm named hybrid seeker optimization (HSO), which integrates seeker optimization algorithm (SOA) with artificial bee colony optimization (ABC) algorithm to solve unconstrained optimization problems. The performance of the proposed HSO algorithm has been tested on a set of test functions and compared with SOA, ABC, DE and three modified versions of the PSO algorithms. The simulation results show that HSO algorithm can outperform the other algorithms in most of the experiments.

The organization of the rest of this paper is as follows. In Sections 2 and 3, SOA and ABC algorithms are described. In Section 4, our proposed hybrid seeker optimization algorithm (HSO) is presented. Benchmark functions are described in Section 5. In Section 6 simulation results and comparisons are presented.

2 Seeker optimization algorithm

Seeker optimization algorithm (SOA) models the behavior of human search population based on their memory, experience, uncertainty reasoning and communication with each other [25]. Therefore the individual of this population is called seeker or searcher. In the SOA, the total population is equally categorized into three subpopulations according to the indexes of the seekers. All the seekers in the same subpopulation constitute a neighborhood which represents the social component for the social sharing of information.

Seeker *i* has the following attributes: the current position $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$, the dimension of the problem *D*, the iteration number *t*, the personal best position $p_{i,best}$ so far, and the neighborhood best position g_{best} so far. The algorithm uses search direction and step length to update the positions of seekers. In the SOA, the search direction is determined by seeker's egoistic behavior, altruistic behavior and pro-activeness behavior, while step length is given by uncertainty reasoning behavior. Search direction α_{ij} and step length d_{ij} are separately computed for each individual *i* on each dimension *j* at each iteration *t*, where $\alpha_{ij} \geq 0$ and $d_{ij} \in \{-1, 0, 1\}$. At each iteration the position of each seeker is updated by:

$$x_{ij}(t+1) = x_{ij}(t) + \alpha_{ij}(t) \cdot d_{ij} \quad (1)$$

where $i = 1, 2, ..., SN, \quad j = 1, 2, ..., D$ (*SN* is the number of seekers). Also, at each iteration, the current positions of the worst two individuals of each subpopulation are exchanged with the best ones in each of the other two subpopulations, which is called inter-subpopulation learning.

The pseudo-code for the SOA is:

$t = 0$;
Generate SN positions uniformly and randomly in the search space;
Evaluate all the seekers and save the historical best position;
repeat
  Compute search direction and step length for each seeker;
  Update each seeker's position using Eq. (1);
  Evaluate all the seekers and save the historical best position;
  Implement the inter-subpopulation learning operation;
  $t = t + 1$;
until $t = T_{max}$

More detailed explanation of calculation of search direction and step length follows.

2.1 Calculation of the search direction

Seeker cooperative behavior types that are modeled are: egoistic, altruistic and pro-active behavior. Seeker's behavior is considered egoistic if he believes that he should go toward his personal best position $p_{i,best}$ through cognitive learning. For reaching the desired goal, by altruistic behavior, seekers want to communicate with

each other and adjust their behaviors in response to other seekers in the same neighborhood region. If a seeker wants to change his search direction and exhibit goal-directed behavior according to his past behavior, then it is considered that his behavior is pro-active. The expression for search direction d_i, which models these types of behavior, for the i^{th} seeker is:

$$d_i = w \cdot sign(p_{i,best} - x_i) + r_1 \cdot (g_{best} - x_i) + \\ + r_2 \cdot (x_i(t1)) - x_i(t2))) \quad (2)$$

where the function *sign()* is a signum function on each dimension of the input vector, w is the inertia weight, $t_1, t_2 \in \{t, t-1, t-2\}$, $x(t1)$ and $x(t2)$ are the best and the worst positions in the set $x(t), x(t-1), x(t-2)$ respectively, and r_1 and r_2 are real numbers chosen uniformly and randomly in the range [0,1]. The balance between global and local exploration and exploitation is provided by reducing the value of inertia weight. Here, inertia weight is linearly decreased from 0.9 to 0.1 during a run.

2.2 Calculation of the step size

Fuzzy reasoning is used to generate the step length because the uncertain reasoning of human searching. The uncertainty rule of intelligent search is described as "If {function value is small}, then {search radius is small}". The linear membership function was used for "short" of "step length". The vector μ_i, which is the grade of membership from cloud model and fuzzy set theory, needs to be calculated in order to calculate the step length. It is inverse proportional to the objective function value of x_i. Hence, the best position so far has the maximum $\mu_{max} = 1.0$, while other positions have a $\mu < 1.0$, and the worst position so far has the minimum μ_{min}. The expression is presented as:

$$\mu_i = \mu_{max} - \frac{S - I_i}{S - 1} \cdot (\mu_{max} - \mu_{min}) \quad (3)$$

where *S* denotes the size of the subpopulation to which the seekers belong, I_i is the sequence number of x_i after sorting the objective function values in ascending order. Besides the vector μ_i , we need to calculate vector δ_i by:

$$\delta_i = w \cdot abs(x_{max} - x_{min}) \quad (4)$$

where the absolute value of the input vector as the corresponding output vector is represented by the symbol *abs()*, x_{max} and x_{min} are the positions of the best and the worst seeker in the subpopulation to which the i^{th} seeker belongs, respectively. In order to introduce the randomness in each variable and to improve the local search capability, the following equation is introduced to convert μ_i into a vector with elements as given by:

$$\mu_{ij} = rand(\mu_i, 1), \quad j = 1, 2, ..., D \quad (5)$$

The equation used for generating the step length α_i for i^{th} seeker is :

$$\alpha_i = \delta_i \cdot \sqrt{-\ln(\mu_i)} \quad (6)$$

3 Artificial bee colony algorithm

In ABC algorithm the colony of artificial bees consists of three groups of bees: employed bees, onlookers and scouts. All bees that are currently exploiting a food source are known as employed bees. The number of the employed bees is equal to the number of food sources and an employed bee is assigned to one of the sources. Each food source is a possible solution for the problem and the nectar amount of a food source represents the quality of the solution represented by the fitness value. Onlookers are those bees that are waiting in the hive for the employed bees to share information about the food sources presently being exploited by them, while scouts are those bees that are searching for new food sources randomly. The number of onlooker and employed bees is the same. Onlookers are allocated to a food source based on probability. Like the employed bees, onlookers calculate a new solution from its food source. After certain number of cycles, if food source cannot be further improved, it is abandoned and replaced by randomly generated food source. This is called exploration process and it is performed by the scout bees. Hence, employed and onlooker bees carry out exploitation process, while scout bees perform exploration. Short pseudocode of the ABC algorithm is given below:

Initialize the population of solutions
Evaluate the population
$t = 0$;
repeat
  Employed bee phase
  Calculate probabilities for onlookers
  Onlooker bee phase
  Scout bee phase
  Memorize the best solution achieved so far
  $t = t + 1$;
until $t = T_{max}$

In employed bee phase an update process is performed for each solution in order to produce a new solution:

$$v_{ij} = x_{ij} + rand \cdot (x_{ij} - x_{kj}) \quad (7)$$

where $k = 1, 2, ..., SN$, $j = 1, 2, ..., D$ are randomly chosen indexes, $k \neq i$, and $rand$ is a random number between [-1,1] (*SN* is the number of solutions, D is the dimension of the problem). Then, a greedy selection is done between x_i and v_i, which completes the update process. The main distinction between the employed bee phase and the onlooker bee phase is that every solution in the employed bee phase involves the update process, while only the selected solutions have the opportunity to update in the onlooker bee phase. An inactive solution refers to a solution that does not change over a certain number of generations. In scout bee phase one of the most inactive solutions is selected and replaced by a new randomly generated solution.

4 Our Proposed Approach: HSO

Hybrid seeker optimization (HSO) algorithm combines two different solution search equations of the ABC algorithm and solution search equation of the SOA in order to improve the performance of SOA and ABC algorithms. Also, HSO algorithm implements the modified inter-subpopulation learning using the binomial crossover operator. Therefore, HSO algorithm has changed the phase of updating seeker's positions and inter-subpopulation learning phase. The initialization phase remained the same as in SOA. Except common control parameters (solution number and maximum number of iterations), the HSO algorithm keeps control parameter *SubpopN* (subpopulation number) from SOA, while it does not include any other control parameter from the ABC algorithm. The introduced modifications are described as follows.

4.1 Modification of updating seeker's positions

In the first 55% (determined empiricaly) of the maximum number of iterations the HSO algorithm is searching for candidate solutions using search formula of ABC which is given by Eq. (7). After each candidate solution is produced and then evaluated, its performance is compared with the old solution and a greedy selection mechanism is employed as the selection operation between the old and the new candidate. If the new solution has better function value than the old candidate solution, it replaces the old one in the memory. In the remaining iterations, HSO chooses between search equation Eq. (1) which is used in SOA and the variant of ABC search equation which can be described as:

$$v_{ij} = \begin{cases} x_{ij} + rand_i \cdot (x_{ij} - x_{kj}) & , \text{ if } R_j < 0.5 \\ x_{ij} & , \text{ otherwise} \end{cases} \quad (8)$$

where R_j is a random number within [0, 1), *k* is randomly chosen index from the whole population and has to be different from *i*, and $rand_i$ is a random number between [-1, 1) and $j = 1, 2, ..., D$. The similar search equation is used in the ABC algorithm extended for constrained optimization problems, but the main difference is that in the Eq. (8) the value $rand_i$ is kept fixed for every $j = 1, 2, ..., D$. Also in [20], this modification is used in order to improve the ABC algorithm for the engineering optimization problems.

The distinction between the Eq. (7) and the Eq. (8) is in the number of the optimization parameters which will be changed. In the basic ABC, while producing a new solution, v_i , changing only one parameter of the parent solution x_i results in a slow convergence rate. In order to overcome this disadvantage, we set the probability of changing the optimization parameter to 0.5. Also, in these iterations, the greedy selection mechanism is not used between the old and the new candidate solution. Hence, the diversity in the population is increased.

In the SOA search equation which is used in HSO, the Eq. (4) for calculating vector δ_i is changed. In [26] it has been concluded that the vector δ is a sensitive parameter and that proposed calculation of its values was not suitable for optimization of multimodal functions. In order to overcome this obstacle, δ_i is calculated by:

$$\delta = w \cdot abs(x_{max} - x_{rand}) \quad (9)$$

where x_{rand} are the positions of the seekers in the same subpopulation where the solution i belongs. Also, in order to further increase the diversity of the solutions, and hence of the population, in the HSO algorithm the inertia weight parameter *w* is linearly decreased from 0.9 to 0.7 during a run.

The HSO included a new control parameter which is called behavior rate (*BR*) in order to select the search equation in the following way: if a random number between [0,1) is less then *BR* the SOA search equation is used, otherwise the Eq. (8) is performed.

4.2 Modification of inter-subpopulation learning

In the modified inter-subpopulation learning the positions of seekers with the lowest objective function values of each subpopulation *l* are combined with the positions of seekers with the highest objective function values of *(l+t) mod SubpopN* subpopulations respectively, where $t = 1, 2, 3, ..., NSC$. *NSC* denotes the number of the worst seekers of each population which are combined with the best seekers. The appropriate seekers are combined using the following binomial crossover operator as expressed in:

$$x_{l_{nj_{worst}}} = \begin{cases} x_{i_j,best} & , \text{ if } R_j < 0.5 \\ x_{l_{nj},worst} & , \text{ otherwise} \end{cases} \quad (10)$$

In Eq. (10) R_j is random number within [0,1), $x_{l_{nj_{worst}}}$ is denoted as the j^{th} variable of the n^{th} worst position in the l^{th} subpopulation, $x_{i_j,best}$ is the j^{th} variable of the best position in the i^{th} subpopulation.

It can be concluded that in the HSO algorithm we have two new control parameters in comparison with the original SOA: the behavior rate *(BR)* and the number of seekers of each subpopulation for combination *(NSC)*. Behavior rate parameter controls which of the search equations for producing new population will be used. In the inter-subpopulation learning of SOA it has been noticed that it may not always bring the benefits for multimodal functions since it may attract all agents towards a local optimal solution. Hence, in order to provide better balance between exploitation and exploration abilities of the algorithm, the described modifications are introduced.

Table 1 Benchmark functions used in experiments

F	Name	C	D	S	fmin
f1	Sphere	US	30	$[-5.12, 5.12]^D$	0
f2	Schwefel 2.22	UN	30	$[-10, 10]^D$	0
f3	Schwefel 1.2	UN	30	$[-100, 100]^D$	0
f4	Schwefel 2.21	UN	30	$[-100, 100]^D$	0
f5	Generalized Rosenbrock	UN	30	$[-30, 30]^D$	0
f6	Step	US	30	$[-100, 100]^D$	0
f7	Quartic Function with Noise	US	30	$[-1.28, 1.28]^D$	0
f8	Generalized Schwefel 2.26	MS	30	$[-500, 500]^D$	-12569.5
f9	Generalized Rastrigin	MS	30	$[-5.12, 5.12]^D$	0
f10	Ackley	MN	30	$[-32, 32]^D$	0
f11	Generalized Griewank	MN	30	$[-600, 600]^D$	0
f12	Penalized	MN	30	$[-50, 50]^D$	0
f13	Penalized2	MN	30	$[-50, 50]^D$	0
f14	Shekel Foxholes	MS	2	$[-65.54, 65.54]^D$	0.998
f15	Kowalik	MN	4	$[-5, 5]^D$	3.075e-4
f16	Six Hump Camel Back	MN	2	$[-5, 5]^D$	-1.0316
f17	Branin	MN	2	$[-5, 15]^D$	0.398
f18	Goldstein-Price	MN	2	$[-2, 2]^D$	3
f19	Hertman3	MN	3	$[0, 1]^D$	-3.86
f20	Hertman6	MN	6	$[0, 1]^D$	-3.32
f21	Shekel5	MN	4	$[0, 10]^D$	-10.1532
f22	Shekel7	MN	4	$[0, 10]^D$	-10.4029
f23	Shekel10	MN	4	$[0, 10]^D$	-10.5364

5 Benchmark functions

To test the performance of the proposed HSO algorithm a test set of well known twenty three benchmark functions is used. Using the large test set is usual when the test involves function optimization, in order to assure a fair comparison between different optimization algorithms. When a test set is too small there is a chance that the algorithm is optimized toward the chosen set of problems. In that case making a general conclusion could be difficult, since such bias might not be useful for other problems of interest. The test suite used in our experiments is introduced in [31], and it is large enough to include many different types of problems such as unimodal, multimodal, regular, irregular, separable, non-separable and multidimensional.

The function is called multimodal if it has more than one local optimum. Multimodal functions are used to test the ability of algorithms to escape from local optimum and locate a good near-global optimum. Therefore in this case the final results are much more important than the convergence rates. Another group of test problems is separable/non-separable functions. A p-variable separable function can be expressed as the sum of p functions of one variable, while non-separable functions cannot be written in this form. Non-separable functions are more difficult than the separable functions, since these functions have interrelation among their variables.

The benchmark functions are given in Table 1. Characteristics of each function are given under the column titled C. If the function is multimodal, abbreviation M is used to indicate this specification, while U means that the function is unimodal. Also, in this column, letter S refers that the function is separable, while letter N that indicates function is non-separable. Further, in Table 1, D denotes the dimensionality of the test problem, S denotes the ranges of the variables, and fmin is a function value of the global optimum. From Table 1 it can be seen that in this test set 16 functions are multimodal, 7 functions are unimodal, 4 functions are separable and 19 functions are non-separable.

6 Experimental Results

The proposed HSO algorithm has been implemented in Java programming language. We used JDK (Java Development Kit) version 6 and Eclipse platform SDK 3.4.2 to develop the application which includes user-friendly graphical user interface (GUI). ABC algorithm is also implemented and tested on these test functions in order to compare the results. Tests were done on an Intel(R) Core(TM) 2 Duo CPU E8500@4-GHz computer with 4 GB RAM memory.

The performance of our proposed hybridized algorithm is compared with the performance of SOA and ABC algorithma. In [26] the SOA results were compared

with results achieved by differential evolution (DE) algorithm [2], particle swarm optimization with inertia weight (PSO-w) [32], PSO with constriction factor (PSO-cf) [33], comprehensive learning particle swarm optimizer (CLPSO) [34]. We include results for these algorithms for comparison with the proposed HSO.

In all experiments for each algorithms the same size of population of 100 is used and the algorithms were repeated 50 runs. The maximum number of generations (G) for each tested problem is the same for each algorithm and their values are listed in Tables 2, 3, 4 and 5. In ABC algorithm the value of control parameter limit is taken *SN*D*0.5*. In our proposed approach the number of subpopulations (*SubpopNum*) is set to 10 and the values of new control parameters are: the behavior rate (*BR*) is 0.5, the number of seekers of each subpopulation for combination (*NSC*) is *0.3*SP/SubpopNum*.

Table 2 shows the best, mean values and standard deviations for 50 independent runs for functions $f_1 - f_7$. From Tables 1 and 2 it can be seen that SOA achieved better results for four out of seven unimodal functions (f_1, f_2, f_4 and f_7) in terms of search precision and robustness. For the test functions f_3 and f_5 HSO algorithm has better results, while for the function f_6 both algorithms perform equally. Although HSO does not reach better results than SOA for the majority of tested unimodal function, its performance is significantly better for the problem f_5. If we compare the performance of HSO algorithm with the performance of ABC algorithm, we can see that HSO algorithm reached better or equal results for all tested unimodal functions. Compared with DE and three variants of PSO, we found that our proposed approach shows better performance for the majority of tested unimodal functions.

In [26] it was concluded that for multimodal test functions the results of SOA were not very satisfactory because it was noticed that for this type of problems algorithm may be stuck at a local minimum. Table 3 show the comparative statistical results for these algorithms, obtained for multimodal functions with many local minima (problems $f_8 - f_{13}$). From Table 3 it can be seen that HSO algorithm enhances the performance of SOA for functions f_8, f_{12} and f_{13}, but also keeps its good performance for problems f_{10} and f_{11}. For the problems f_{10} and f_{11} SOA performs better than HSO at only fine-grained search. For the problem f_9, SOA and HSO perform the same. When comparing our approach with respect to ABC, we can conclude that HSO algorithm achieved better or equal results for all tested multimodal problems with many local minima. From Table 3, it is clear that HSO algorithm performs significantly better than DE, PSO-w and PSO-cf consistently for these functions.

Table 4 and Table 5 show the comparative statistical results of these algorithms, obtained for multimodal functions with only a few local minima (problems $f_{14} - f_{23}$). From Table 4 and Table 5 it can be seen that SOA is outperformed by HSO algorithm for functions f_{14} and $f_{19} - f_{23}$. For the rest of the tested functions with only a few local minima, HSO achieved the same mean values as SOA, which are equal to the optimal values, but

Table 2 Statistical results for 50 runs obtained by DE, PSO-w, PSO-cf, CLPSO, ABC, SOA and HSO algorithms for functions f1-f7

Func.	Algor.	Best	Mean	Std. Dev.
f1	DE	5.20e14	3.74e13	3.94e13
(*G*=1500)	PSO-w	1.79e-15	1.66e-13	4.59e-13
	PSO-cf	4.50e-45	2.28e-41	4.54e-41
	CLPSO	3.22e-13	2.73e-12	1.68e-12
	SOA	1.94e-83	1.02e-76	6.51e-76
	ABC	3.04e-16	5.09e-16	4.82e-17
	HSO	7.64e-26	1.42e-24	3.33e-24
f2	DE	6.17e-10	3.74e-09	2.20e-09
(*G*=2000)	PSO-w	5.36e-12	6.67e-11	7.98e-11
	PSO-cf	3.29e-29	1.60e-00	4.22e-00
	CLPSO	1.63e-09	3.82e-09	1.73e-09
	SOA	3.40e-65	4.22e-63	8.25e-63
	ABC	1.85e-15	2.57e-15	5.01e-16
	HSO	4.84e-18	1.39e-17	8.52e-18
f3	DE	1.10e-11	1.85e-10	1.49e-10
(*G*=5000)	PSO-w	2.00e-02	2.40e-01	2.23e-01
	PSO-cf	3.01e-19	3.33e+02	1.78e+03
	CLPSO	3.37e-02	4.20e-01	3.62e-01
	SOA	5.85e-35	4.26e-25	2.15e-24
	ABC	2.65e-16	4.21e-16	8.04e-17
	HSO	5.12e-86	1.21e-83	5.19E-83
f4	DE	6.83e-13	3.10e-02	8.70e-02
(*G*=5000)	PSO-w	1.18e-02	7.02e-02	4.66e-02
	PSO-cf	1.48e-16	7.13e-13	2.19e-12
	CLPSO	6.88e-04	2.05e-03	1.25e-03
	SOA	1.55e-53	1.02e-48	2.46e-48
	ABC	6.56e-02	1.51e-01	2.12e-01
	HSO	1.04e-04	1.77e-02	3.65e-02
f5	DE	0	3.47e-31	2.45e-30
(*G*=20000)	PSO-w	1.05e-02	1.82e+03	1.27e+03
	PSO-cf	1.87e-12	7.32e+03	2.46e+03
	CLPSO	1.68e-01	3.63e+01	3.12e+01
	SOA	2.42e+01	2.54e+01	7.87e-01
	ABC	8.96e-05	3.06e-02	4.68e-02
	HSO	1.90e-06	5.70e-04	1.77e-03
f6	DE	0	0	0
(*G*=1500)	PSO-w	0	0	0
	PSO-cf	0	0	0
	CLPSO	0	0	0
	SOA	0	0	0
	ABC	0	0	0
	HSO	0	0	0
f7	DE	1.97e-03	4.66e-03	1.30e-03
(*G*=3000)	PSO-w	2.99e-03	6.28e-03	2.17e-03
	PSO-cf	9.86e-04	2.45e-03	1.38e-03
	CLPSO	1.03e-03	2.98e-03	9.72e-04
	SOA	2.66e-05	1.08e-04	6.44e-05
	ABC	1.52e-02	2.95e-02	7.35e-03
	HSO	3.29e-04	8.87e-04	4.46e-04

Table 3 Statistical results for 50 runs obtained by DE, PSO-w, PSO-cf, CLPSO, ABC, SOA and HSO algorithms for functions f8-f13

Func.	Algor.	Best	Mean	Std. Dev.
f8	DE	-12096	-11234	455.5
(*G*=9000)	PSO-w	-10495	-9363.3	445.3
	PSO-cf	-10398	-9026.1	656.9
	CLPSO	-12569	-12271	177.8
	SOA	-11760	-10126	669.5
	ABC	-12569	-12569	4.82e-17
	HSO	-12569	-12569	3.33e-24
f9	DE	9.95e-00	8.10e+01	3.23e+01
(*G*=5000)	PSO-w	7.96e-00	2.10e+01	8.01e-00
	PSO-cf	2.69e+01	6.17e+01	1.84e+01
	CLPSO	2.85e-10	1.34e-09	8.57e-10
	SOA	0	0	0
	ABC	0	0	0
	HSO	0	0	0
f10	DE	5.79e-08	1.71e-07	7.66e-08
(*G*=1500)	PSO-w	1.39e-07	1.66e-06	2.66e-06
	PSO-cf	2.67e-15	5.59e-01	7.30e-01
	CLPSO	3.31e-06	6.81e-06	1.94e-06
	SOA	-4.44e-15	-4.44e-15	0
	ABC	7.44e-11	6.43e-10	4.73e-10
	HSO	8.36e-11	2.13e-11	4.45e-11
f11	DE	0	4.44e-04	1.77e-03
(*G*=2000)	PSO-w	0	1.59e-01	2.19e-02
	PSO-cf	0	1.11e-02	1.25e-02
	CLPSO	1.64e-14	2.96e-04	1.46e-03
	SOA	0	0	0
	ABC	0	8.22e-17	8.55e-17
	HSO	0	6.66e-18	3.45e-17
f12	DE	3.40e-15	3.67e-14	4.07e-14
(*G*=1500)	PSO-w	8.85e-15	2.21e-00	5.52e-00
	PSO-cf	1.57e-32	1.66e+01	1.81e+01
	CLPSO	8.80e-12	4.80e-11	3.96e-11
	SOA	3.04e-04	1.28e-02	7.67e-03
	ABC	3.31e-16	5.15e-16	6.44e-17
	HSO	6.67e-25	2.11e-23	2.77e-23
f13	DE	4.13e-14	2.91e-13	2.88e-13
(*G*=1500)	PSO-w	8.23e-07	5.72e+02	3.57e+02
	PSO-cf	1.35e-32	2.40e+02	2.40e+02
	CLPSO	1.18e-10	6.42e-10	4.46e-10
	SOA	2.77e-03	1.89e-01	1.30e-01
	ABC	4.48e-16	5.60e-13	2.09e-11
	HSO	1.47e-22	2.05e-20	2.74e-20

with better standard deviations. If we compare the results of HSO and ABC algorithms, we can conclude that both algorithms perform very similar for problems $f_{14} - f_{23}$. HSO reached slightly better results for problems f_{15} and f_{23}. Compared with DE and three variants of PSO, we found that our proposed approach shows better or equal performance for the majority of tested multimodal functions with only a few local minima.

Table 4 Statistical results for 50 runs obtained by DE, PSO-w, PSO-cf, CLPSO, ABC, SOA and HSO algorithms for functions f14-f18

Func.	Algor.	Best	Mean	Std.Dev.
f14	DE	0.998	0.998	2.88e-16
(*G*=100)	PSO-w	0.998	1.026	1.52e-01
	PSO-cf	0.998	0.998	8.69e-13
	CLPSO	0.998	0.998	5.63e-10
	SOA	0.998	1.199	5.30e-01
	ABC	0.998	0.998	1.05e-15
	HSO	0.998	0.998	2.68e-14
f15	DE	3.0749e-04	4.7231e-02	3.55e-04
(*G*=4000)	PSO-w	3.0749e-04	2.0218e-03	5.47e-03
	PSO-cf	3.0749e-04	2.0225e-03	5.47e-03
	CLPSO	3.2847e-04	5.3715e-04	6.99e-05
	SOA	3.0749e-04	3.0749e-04	1.58e-09
	ABC	3.0824e-04	3.9940e-04	5.15e-05
	HSO	3.0749e-04	3.0749e-04	5.94e-19
f16	DE	-1.0316	-1.0316	6.77e-13
(*G*=100)	PSO-w	-1.0316	-1.0316	8.80e-12
	PSO-cf	-1.0316	-1.0316	5.92e-12
	CLPSO	-1.0316	-1.0316	8.50e-14
	SOA	-1.0316	-1.0316	6.73e-06
	ABC	-1.0316	-1.0316	9.47e-15
	HSO	-1.0316	-1.0316	1.01e-07
f17	DE	0.39789	0.39789	1.14e-08
(*G*=100)	PSO-w	0.39789	0.39789	2.33e-12
	PSO-cf	0.39789	0.39789	5.25e-12
	CLPSO	0.39789	0.39789	1.08e-13
	SOA	0.39789	0.39838	5.14e-04
	ABC	0.39789	0.39789	8.42e-08
	HSO	0.39789	0.39789	4.30e-06
f18	DE	3	3	3.31e-15
(*G*=100)	PSO-w	3	3	2.50e-11
	PSO-cf	3	3	2.05e-11
	CLPSO	3	3	5.54e-13
	SOA	3	3.0001	1.17e-04
	ABC	3	3	6.77e-07
	HSO	3	3	3.38e-11

7 Conclusion

In this paper we have presented hybridization of the seeker optimization algorithm with the well known ABC algorithm. At earlier stages we use search formulas from the ABC and also modify the inter-subpopulation learning phase by using the binomial crossover operator. Our proposed HSO algorithm has been implemented and tested on complete set of well-known 23 benchmark functions taken from literature. The results show that our algorithm performs better to SOA, ABC, standard DE and three modified particle swarm optimization (PSO) algorithms considering the quality of the solutions found and robustness for the most benchmark problems. Our hybridization successfully overcome SOA algorithm's tendency to prematurely converge to local optima for

Table 5 Statistical results for 50 runs obtained by DE, PSO-w, PSO-cf, CLPSO, ABC, SOA and HSO algorithms for functions f19-f23

Func.	Algor.	Best	Mean	Std. Dev.
f19	DE	-3.8628	-3.8628	1.97e-15
(*G*=100)	PSO-w	-3.8628	-3.8628	2.66e-11
	PSO-cf	-3.8628	-3.8628	2.92e-12
	CLPSO	-3.8628	-3.8628	6.07e-12
	SOA	-3.8628	-3.8621	6.69e-04
	ABC	-3.8628	-3.8628	5.00e-11
	HSO	-3.8628	-3.8628	5.31e-11
f20	DE	-3.322	-3.215	0.036
(*G*=200)	PSO-w	-3.322	-3.256	0.066
	PSO-cf	-3.322	-3.277	0.058
	CLPSO	-3.322	-3.274	0.059
	SOA	-3.321	-3.298	0.045
	ABC	-3.322	-3.322	2.24e-11
	HSO	-3.322	-3.322	1.45e-11
f21	DE	-10.15	-10.15	4.67e-06
(*G*=100)	PSO-w	-6.57	-2.01	1.10e-00
	PSO-cf	-10.15	-6.23	3.25e-00
	CLPSO	-10.14	-9.57	4.28e-01
	SOA	-10.15	-9.67	4.96e-01
	ABC	-10.15	-10.15	6.92e-04
	HSO	-10.15	-10.15	1.02e-04
f22	DE	-10.40	-10.40	2.07e-07
(*G*=100)	PSO-w	-4.61	-2.14	8.34e-01
	PSO-cf	-10.40	-6.47	3.56e-00
	CLPSO	-10.34	-9.40	1.12e-00
	SOA	-10.40	-9.79	4.48e-01
	ABC	-10.40	-10.40	6.92e-04
	HSO	-10.40	-10.40	2.07e-05
f23	DE	-10.54	-10.54	3.21e-06
(*G*=100)	PSO-w	-6.63	-2.20	1.01e-00
	PSO-cf	-10.54	-8.11	3.47e-01
	CLPSO	-10.46	-9.47	1.25e-00
	SOA	-10.54	-9.72	4.72e-01
	ABC	-10.54	-10.53	3.14e-03
	HSO	-10.54	-10.54	7.26e-04

multimodal functions. Further tuning and adjustments can be included in the future research.

References

[1] J. Holland, Adaptation in Natural and Artificial Systems (MIT Press, Cambridge, MA, 1992).

[2] R. Storn, K. Price: Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization, **11(4)**, 341-359 (1997)

[3] M. Dorigo, L.M. Gambardella: Ant colonies for the traveling salesman problem, BioSystems, **43(2)**, 73-81 (1997)

[4] J. Kennedy, R.C. Eberhart: Particle swarm optimization, Proc. of the 1995 IEEE International Conference on Neural Networks, 4, 1942-1948 (1995).

[5] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Technical report-tr06, Erciyes University, Engineering FacultyMA, Computer Engineering Department, 2005

[6] D. Karaboga, B. Akay: A comparative study of Artificial Bee Colony algorithm, Applied Mathematics and Computation, **214(1)**, 108-132 (2009)

[7] X. S. Yang, S. Deb: Cuckoo search via Lvy flights, Proc. of World Congress on Nature & Biologically Inspired Computing, 210 -214 (2009)

[8] R. Jovanovic, M. Tuba: An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem, Applied Soft Computing, **11(8)**, 5360-5366 (2011)

[9] R. Jovanovic, M. Tuba: Ant Colony Optimization Algorithm with Pheromone Correction Strategy for Minimum Connected Dominating Set Problem, Computer Science and Information Systems (ComSIS), **9(4)**, DOI:10.2298/CSIS110927038J (2012)

[10] M. Tuba, R. Jovanovic: An Analysis of Different Variations of Ant Colony Optimization to the Minimum Weight Vertex Cover Problem, WSEAS Transactions on Information Science and Applications, **6(6)**, 936-945 (2009)

[11] R. Cheng, M. Yao : Particle Swarm Optimizer with Time-Varying Parameters based on a Novel Operator, Applied Mathematics & Information Sciences, **5(2)**, 33-38 (2011)

[12] M. Tuba, M. Subotic, N. Stanarevic: Performance of a modified cuckoo search algorithm for unconstrained optimization problems, WSEAS Transactions on Systems, **11(2)**, 62-74 (2012)

[13] M. Tuba, N. Bacanin and N. Stanarevic: Adjusted artificial bee colony (ABC) algorithm for engineering problems, WSEAS Transaction on Computers, **11(4)**, 111-120 (2012)

[14] A. Martinez-Estudillo, C. Hervas-Martinez, F. Martinez-Estudillo, N.Garcia-Pedrajas: Hybridization of Evolutionary Algorithms and Local Search by Means of a Clustering Method, IEEE transactions on systems, man and cybernetics part B: Cybernetics, **36(3)**, 534-545 (2005)

[15] YM. Wang, DM. Wang, XX. Zhong, GQ. Shi: Modified Particle Swarm Algorithm for Radiation Properties of Semi-transparent Rectangular Material, Applied Mathematics & Information Sciences, **5(2)**, 227-233 (2011)

[16] HC. Li, YP. Wang: An Evolutionary Algorithm with Local Search for Convex Quadratic Bilevel Programming Problems, Applied Mathematics & Information Sciences, **5(2)**, 139-146 (2011)

[17] JH. Gao, R. Shan: A New Method for Modification Consistency of the Judgment Matrix Based on Genetic Ant Algorithm, Applied Mathematics & Information Sciences, **6(1)**, 35-39 (2012)

[18] D. Karaboga, B. Basturk: A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm, Journal of Global Optimization, **39(3)**, 459-471 (2007)

[19] N. Bacanin, M. Tuba: Artificial Bee Colony (ABC) Algorithm for Constrained Optimization Improved with Genetic Operators, Studies in Informatics and Control, **21(2)**, 137-146 (2012)

[20] I.Brajevic, M. Tuba: An upgraded artificial bee colony algorithm (ABC) for constrained optimization problems, Journal of Intelligent Manufacturing, published Online First, DOI: 10.1007/s10845-011-0621-6, (2012)
[21] I. Brajevic, M. Tuba, M. Subotic: Performance of the improved artificial bee colony algorithm on standard engineering constrained problems, International Journal of Mathematics and Computers in Simulation, **5(2)**, 135-143 (2011)
[22] N. Stanarevic, M. Tuba, N. Bacanin: Modified artificial bee colony algorithm for constrained problems optimization, International Journal of Mathematical Models and Methods in Applied Sciences, **5(3)**, 644-651 (2011)
[23] N. Bacanin, M. Tuba, I. Brajevic: Performance of object-oriented software system for improved artificial bee colony optimization, International Journal of Mathematics and Computers in Simulation, **5(2)**, 154-162 (2011)
[24] M. Subotic, M. Tuba, N. Stanarevic: Different approaches in parallelization of the artificial bee colony algorithm, International Journal of Mathematical Models and Methods in Applied Sciences, **5(4)**, 755-762 (2011)
[25] C. Dai, Y. Zhu, W. Chen: Seeker Optimization Algorithm, Lecture Notes in Computer Science, 4456, 167-176 (2007)
[26] C. Dai, W. Chen, Y. Song, Y. Zhu: Seeker optimization algorithm: a novel stochastic search algorithm for global numerical optimization, Journal of Systems Engineering and Electronics, **21(2)**, 300-311 (2010)
[27] C. Dai, W. Chen, Y. Zhu, Z. Jiang, Z. You: Seeker optimization algorithm for tuning the structure and parameters of neural networks, Neurocomputing, **74(6)**, 876-883 (2011)
[28] C. Dai, W. Chen, Y. Zhu: Seeker optimization algorithm for digital IIR filter design, IEEE Transactions on Industrial Electronics, **57(5)**, 1710-1718 (2010)
[29] C. Dai, Z. Cheng, Q. Li, Z. Jiang, J. Jia: Seeker optimization algorithm for global optimization: A case study on optimal modelling of proton exchange membrane fuel cell (PEMFC), International Journal of Electrical Power and Energy Systems, **33(3)**, 369-376 (2011)
[30] C. Dai, W. Chen, L. Ran, Y. Zhang, Y. Du: Human Group Optimizer with Local Search, Lecture Notes in Computer Science, **6728**, 310-320 (2011)
[31] J. Brest, S. Greiner, B. Boskovic: Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. on Evolutionary Computation, **10(6)**, 646-657 (2006)
[32] Y. Shi, R. Eberhart: Empirical study of particle swarm optimization, Proc. of the Congress on Evolutionary Computation, **3**, 1945-1950 (1999)
[33] M. Clerc, J. Kennedy: The particle swarm - explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. on Evolutionary Computation, **6(1)**, 58-73 (2002)
[34] J. J. Liang, A. K. Qin, P. N. Suganthan: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. on Evolutionary Computation, **10(3)**, 67-82 (2006)

Milan Tuba is Professor of Computer Science and Provost for Mathematical, Natural and Technical sciences at Megatrend University of Belgrade. He received B. S. in Math., M. S. in Math., M. S., M. Ph., Ph. D. in Computer Science from University of Belgrade and New York University. From 1983 to 1994 he was in the U.S.A. at Vanderbilt University in Nashville, Courant Institute of Mathematical Sciences, New York University and Cooper Union Graduate School of Engineering, New York. From 1994 he was Professor of Computer Science and Director of Computer Center at University of Belgrade, and from 2004 also a Dean of the College of Computer Science, Megatrend University Belgrade. His research interest includes mathematical, queuing theory and heuristic optimizations applied to computer networks, image processing and combinatorial problems. He has been an invited speaker of number of conferences and member of the editorial board or scientific committee of number of scientific journals and conferences and has published more than 100 scientific papers.

Ivona Brajevic received B.S. in mathematics in 2006 and M.S. in mathematics in 2008 from University of Belgrade, Faculty of Mathematics. She is currently Ph.D. student at Faculty of Mathematics, Computer science department, University of Belgrade and works as teaching assistant at College of Business and Economy in Belgrade. She is the coauthor of seven scientific papers. Her current research interest includes nature inspired metaheuristics.

Raka Jovanovic received Ph. D. from University of Belgrade, Faculty of Mathematics in 2012. Worked as a research associate at the Institute of Physics, University of Belgrade and at Texas AM University at Qatar 2007-2011. Has published more then 25 articles in international journals and conference proceedings. Research interests: Optimization Problems, Data Compression, Image Processing, Numeric Simulation and Fractal Imaging.