Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 709738, 12 pages
http://dx.doi.org/10.1155/2014/709738

Research Article

A Novel Hybrid Self-Adaptive Bat Algorithm

Iztok Fister Jr.,[1] Simon Fong,[2] Janez Brest,[1] and Iztok Fister[1]

[1] *Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia*
[2] *Department of Computer and Information Science, University of Macau, Avenue Padre Tomas Pereira, Taipa, Macau*

Correspondence should be addressed to Iztok Fister Jr.; iztok.fister2@uni-mb.si

Received 17 November 2013; Accepted 9 March 2014; Published 9 April 2014

Academic Editors: J. Shu and F. Yu

Nature-inspired algorithms attract many researchers worldwide for solving the hardest optimization problems. One of the newest members of this extensive family is the bat algorithm. To date, many variants of this algorithm have emerged for solving continuous as well as combinatorial problems. One of the more promising variants, a self-adaptive bat algorithm, has recently been proposed that enables a self-adaptation of its control parameters. In this paper, we have hybridized this algorithm using different DE strategies and applied these as a local search heuristics for improving the current best solution directing the swarm of a solution towards the better regions within a search space. The results of exhaustive experiments were promising and have encouraged us to invest more efforts into developing in this direction.

1. Introduction

Optimization has become more and more important, especially during these times of recession. It has been established throughout practically all spheres of human activities, for example, finances, industries, sport, pharmacy, and so forth. In each of these spheres, a goal of the optimization is to find the optimal input parameters according to known outcomes and a known model. This model represents a problem to be solved and transforms input parameters to output outcomes. Essentially, the optimal input parameters must be properly evaluated in order to determine the quality of a solution. Indeed, an objective function is used that mathematically describes this quality. In the praxis, the value of the objective function can be either minimized or maximized. For example, when buying a car either the minimum cost $\min(f(x))$ or maximum comfort $\max(f(x))$ is interesting objective for a potential buyer, where $f(x)$ denotes the objective function. Using the equation $\min(f(x)) = \max(-f(x))$, the maximization problem can be transformed into a minimization one, and vice versa. The function expressed in this way is also named as fitness function in evolutionary computation community. In place of the objective function, the minimization of the fitness function is assumed in this paper.

Most of the problems arising in practice today are NP-hard. This means that the time complexity of an exhaustive search algorithm running on a digital computer which checks all solutions within a search space increases exponentially by increasing the instance size as determined by the number of input parameters. In some situations, when the number of input parameters increases to a certain limit, it can be expected that a user never obtains the results from the optimization. As a result, algorithms solving the NP-problems approximately have arisen in the past. Although these algorithms do not find the exact optimal solution, in general, their solutions are good enough in practice. For instance, the well-known algorithms for approximately solving the hardest problems today are

(i) artificial bee colony (ABC) [1],

(ii) bat algorithm (BA) [2],

(iii) cuckoo search (CS) [3],

(iv) differential evolution (DE) [4],

(v) firefly algorithm (FA) [5, 6],

(vi) particle swarm optimization (PSO) [7],

(vii) many more [8].

The developers of the above-mentioned algorithms were typically inspired by nature in a development phase. Namely, each nature-inspired algorithm mimics the natural behaviors of specific biological, chemical, or physical systems in order to solve particular problems using a digital computer. Most of the inspiration from nature emanates from biology. The greatest lasting impression on the developers has been left by Darwin's evolution [9]. Darwin observed that nature is not static and therefore the fitter individuals have more chances of surviving within the changing environments. Holland in [10] connected this adaptive behavior of the natural systems to artificial systems that simulate their behavior by solving the particular problems (also optimization problems) on the digital computers.

Optimization algorithms are controlled by algorithm parameters that can be changed deterministically, adaptively, and self-adaptively [11]. Deterministic parameters are altered by using some deterministic rules (e.g., Rechenberg's 1/5 success rule [12]). In contrast, adaptively controlled parameters are subject to feedback from the search process that serves as an input to the mechanism used which determines the direction and magnitude of the change [11]. Finally, the self-adaptively controlled parameters are encoded into a representation of the solution and undergo the actions of variation operators [13].

This paper focuses on the adaptation and hybridization of a swarm intelligence algorithm. Swarm intelligence (SI) belongs to an artificial intelligence discipline (AI) which first became popular over the last decade and still is [14]. It is inspired by the collective behavior of social swarms of ants, termites, bees, and worms, flock of birds, and schools of fish [15]. Although these swarms consist of relatively unsophisticated individuals, they exhibit coordinated behavior that directs the swarms towards their desired goals. This usually results in the self-organizing behavior of the whole system, and collective intelligence or swarm intelligence is in essence the self-organization of such multiagent systems, based on simple interaction rules.

The bat algorithm (BA) is one of the younger members of this family which was developed by Yang [16]. It is inspired by the microbats that use an echolocation for orientation and prey seeking. The original bat algorithm employs two strategy parameters: the pulse rate and the loudness. The former regulates an improvement of the best solution, while the latter influences an acceptance of the best solution. Both mentioned parameters are fixed during the execution of the original bat algorithm. In the self-adaptive bat algorithm (SABA) developed by Fister et al. [17] these parameters are self-adaptive. The aim of this self-adaptation is twofold. On the one hand, it is very difficult to guess the valid value of the parameter at the beginning of the algorithm. On the other hand, this value depends on the phase in which the search process is. This means that the parameter setting at the beginning of the search process can be changed when this process reaches maturity. In this paper, a further step forward has been taken.

Hybridization with local search heuristics has now been applied in order to further improve the results of the SABA algorithm. Domain-specific knowledge can be incorporated using the local search. Although the local search is as yet an ingredient of the original bat algorithm it can be replaced by different DE strategies [18]. Indeed, the further improvement of the current best solution is expected which would direct the swarm intelligence search towards the more promising areas of the search space. As a result, the hybrid self-adaptive bat algorithm (HSABA) that was applied to a benchmark suite consisting of ten well-known functions from the literature was developed. The results of HSABA obtained during extensive experiments showed that the HSABA improved the results of both the original BA and the SABA. Moreover, the results were also comparable with the other well-known algorithms, like firefly (FA) [5], differential evolution (DE) [19], and artificial bee colony (ABC) [20].

The structure of this paper is as follows. Section 2 presents an evolution of the bat algorithms from the original BA via the self-adaptive SABA to the hybrid self-adaptive HSABA algorithm. Section 3 describes the experimental work, while Section 4 illustrates the obtained results in detail. The paper concludes by summarizing the performed work and outlines directions for the further development.

2. Evolution of Bat Algorithms

The results of experiments regarding the original bat algorithm showed that this algorithm is efficient especially when optimizing the problems of lower dimensions. In line with this, Eiben and Smith in [11] asserted that 2-dimensional functions are not suitable for solving with the population-based stochastic algorithms (like evolutionary algorithms and swarm intelligence), because these can be solved optimally using traditional methods. On the other hand, these kinds of algorithms could play a role as general problem solvers, because they share the same performances when averaged over all the discrete problems. This fact is the essence of the so-called No-Free Lunch (NFL) theorem [21]. In order for this theorem to prevail, there are almost two typical mechanisms for improving the performance of the population-based algorithms as follows:

(i) self-adaptation of control parameters,

(ii) hybridization.

The former enables the control parameters to be changed during the search process in order to better suit the exploration and exploitation components of this search process [22], while the latter incorporates the problem-specific knowledge within it.

In the rest of this paper, firstly the original BA algorithm is presented in detail, followed by describing the application of a self-adaptive mechanism within the original BA algorithm which leads to the emergence of a self-adaptive BA algorithm SABA. Finally, a hybridization of the SABA algorithm is broadly discussed which obtains a hybridized SABA algorithm named the HSABA.

2.1. The Original Bat Algorithm. The original BA is a population-based algorithm, where each particle within the

Input: Bat population $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD})^T$ for $i = 1, \ldots, Np$, MAX_FE.
Output: The best solution $\mathbf{x}_{\text{best}}$ and its corresponding value $f_{\min} = \min(f(\mathbf{x}))$.

```
(1) init_bat();
(2) eval = evaluate_the_new_population;
(3) f_min = find_best_solution(x_best); {initialization}
(4) while termination_condition_not_meet do
(5)   for i = 1 to Np do
(6)     y = generate_new_solution(x_i);
(7)     if rand(0, 1) > r_i then
(8)       y = improve_the_best_solution(x_best)
(9)     end if {local search}
(10)    if f_new = evaluate_new_solution(y);
(11)    eval = eval + 1;
(12)    if f_new ≤ f_i and N(0, 1) < A_i then
(13)      x_i = y; f_i = f_new;
(14)    end if {simulated annealing}
(15)    f_min = find_the_best_solution(x_best);
(16)  end for
(17) end while
```

Algorithm 1: Pseudocode of the original bat algorithm.

bat population represents the candidate solution. The candidate solutions are represented as vectors $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD})^T$ for $i = 1 \cdots Np$ with real-valued elements x_{ij}, where each element is drawn from interval $x_{ij} \in [x_{lb} \cdots x_{ub}]$. Thus, x_{lb} and x_{ub} denote the corresponding lower and upper bounds, and Np determines a population size.

This algorithm consists of the following main components:

(i) an initialization,

(ii) a variation operation,

(iii) a local search,

(iv) an evaluation of a solution,

(v) a replacement.

In the *initialization*, the algorithm parameters are initialized, then, the initial population is generated randomly, next, this population is evaluated, and finally, the best solution in this initial population is determined. The *variation operator* moves the virtual bats in the search space according to the physical rules of bat echolocation. In the *local search*, the current best solution is improved by the random walk direct exploitation heuristics (RWDE) [23]. The quality of a solution is determined during the *evaluation of solution*. The *replacement* replaces the current solution with the newly generated solution regarding the some probability. This component is similar to the simulated annealing [24], where the new solution is accepted by the acceptance probability function which simulates the physical rules of annealing. The pseudocode of this algorithm is presented in Algorithm 1.

In Algorithm 1 the particular component of the BA algorithm is denoted either by function name when it comprises one line or by a comment between two curly brackets when the components are written within a structured statement and/or it comprises more lines. In line with this, the initialization comprises lines 1–3 in Algorithm 1, the variation operation line 6 (function *generate_new_solution*), the local search lines 7–9, the evaluation of the solution (function *evaluate_new_solution* in line 10), and the replacement lines 12–14. In addition, the current best solution is determined in each generation (function *find_best_solution* in line 15).

The variation operation which is implemented in function *generate_new_solution* moves the virtual bats towards the best current bat's position according to the following equations:

$$\begin{aligned} Q_i^{(t)} &= Q_{\min} + \left(Q_{\max} - Q_{\min}\right) N(0,1), \\ \mathbf{v}_i^{(t+1)} &= \mathbf{v}_i^t + \left(\mathbf{x}_i^t - \mathbf{best}\right) Q_i^{(t)}, \\ \mathbf{x}_i^{(t+1)} &= \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t+1)}, \end{aligned} \tag{1}$$

where $N(0, 1)$ is a random number drawn from a Gaussian distribution with zero mean and a standard deviation of one. A RWDE heuristics [23] implemented in the function *improve_the_best_solution* modifies the current best solution according to the following equation:

$$\mathbf{x}^{(\mathbf{t})} = \mathbf{best} + \epsilon A_i^{(t)} N(0,1), \tag{2}$$

where $N(0, 1)$ denotes the random number drawn from a Gaussian distribution with zero mean and a standard deviation of one, ϵ being the scaling factor and $A_i^{(t)}$ the loudness.

A local search is launched with the probability of pulse rate r_i. As already stated, the probability of accepting the new best solution in the component *save_the_best_solution_conditionaly* depends on loudness A_i. Actually, the original BA algorithm is controlled by two algorithm parameters: the pulse rate r_i and the loudness A_i. Typically, the rate of

pulse emission r_i increases and the loudness A_i decreases when the population draws nearer to the local optimum. Both characteristics imitate natural bats, where the rate of pulse emission increases and the loudness decreases when a bat finds a prey. Mathematically, these characteristics are captured using the following equations:

$$A_i^{(t+1)} = \alpha A_i^{(t)}, \qquad r_i^{(t)} = r_i^{(0)} \left[1 - \exp\left(-\gamma\epsilon\right)\right], \tag{3}$$

where α and γ are constants. Actually, the α parameter controls the convergence rate of the bat algorithm and therefore plays a similar role as the cooling factor in the simulated annealing algorithm.

In summary, the original BA algorithm is based on a PSO algorithm [7] which is hybridized with RWDE and simulated annealing heuristics. The former represents the local search that directs the bat search process towards improving the best solution, while the latter takes care of the population diversity. In other words, the local search can be connected with exploitation, while simulated annealing uses the exploration component of the bat search process. The exploitation is controlled by the parameter r and exploration by the parameter A. As a result, the BA algorithm is able to explicitly control the exploration and exploitation components within its search process.

2.2. The Self-Adaptive Bat Algorithm. Almost two advantages can be expected when the self-adaptation of control parameters is applied to a population-based algorithm [11]:

(i) control parameters need not be set before the algorithm's run,

(ii) the control parameters are adapted during the run to the fitness landscape defined by the positions of the candidate solutions within the search space and their corresponding fitness values [25].

Normally, the self-adaptation is realized by encoding the control parameters into representation of candidate solutions and letting them undergo an operation of the variation operators. In this way, the self-adaptation of the BA control parameters (the loudness and the pulse rate) is considered.

This means that the existing representation of the candidate solutions consisting of problem variables $\mathbf{x}_i^{(t)} = (x_{i1}^{(t)}, \ldots, x_{iD}^{(t)})$ is widened by the control parameters $A^{(t+1)}$ and $r^{(t+1)}$ to

$$\mathbf{x}_i^{(t)} = \left(x_{i1}^{(t)}, \ldots, x_{iD}^{(t)}, A^{(t)}, r^{(t)}\right)^T, \quad \text{for } i = 1 \cdots Np, \tag{4}$$

where Np denotes the population size. The control parameters are modified according to the following equations:

$$A^{(t+1)} = \begin{cases} A_{lb}^{(t)} + \text{rand}_0\left(A_{ub}^{(t)} - A_{lb}^{(t)}\right) & \text{if } \text{rand}_1 < \tau_1, \\ A^{(t)} & \text{otherwise,} \end{cases} \tag{5}$$

$$r^{(t+1)} = \begin{cases} r_{lb}^{(t)} + \text{rand}_2\left(r_{ub}^{(t)} - r_{lb}^{(t)}\right) & \text{if } \text{rand}_3 < \tau_2, \\ r^{(t)} & \text{otherwise.} \end{cases} \tag{6}$$

Note that the parameters τ_0 and τ_1 denote the learning rates that were set, as $\tau_0 = \tau_1 = 0.1$, while rand_i for $i = 1 \cdots 4$ designate the randomly generated value from interval $[0, 1]$.

The self-adapting part of the SABA algorithm is performed by the *generate_the_new_solution* function (line 6 in Algorithm 1). Note that the control parameters are modified in this function according to the learning rates τ_0 and τ_1. In the case $\tau_0 = \tau_1 = 0.1$, each 10th candidate solution is modified on average. The modified parameters influence the application of the local search and the probability of the replacement. The replacement function preserves the problem variables as well as the control parameters.

This self-adapting function was inspired by Brest et al. [4] who proposed the self-adaptive version of DE, better known as jDE. This self-adaptive algorithm improves the results of the original DE significantly by continuous optimization.

2.3. The Hybrid Self-Adaptive Bat Algorithm. The population-based algorithms, like evolutionary algorithms and swarm intelligence, can be seen as some kind of general problem solvers, because they are applicable for all classes of optimization problems. Thus, their results confirm the so-called No-Free Lunch (NFL) theorem by Wolpert and Macready [21]. According to this theorem any two optimization algorithms are equivalent when their performances are averaged across all possible problems.

In order to circumvent the NFL theorem by solving a specific problem, domain-specific knowledge must be incorporated within the algorithm for solving it. The domain-specific knowledge can be incorporated by the bat algorithm within each of its components, that is, an initialization, a variation operator, a local search, an evaluation function, and a replacement.

Although the SABA algorithm significantly outperformed the results of the original BA algorithm, it suffers from a lack of incorporated domain-specific knowledge of the problem to be solved. This paper focuses on hybridizing the SABA using a novel local search heuristics that better exploits the self-adaptation mechanism of this algorithm. The standard "rand/1/bin" DE strategy and three other DE strategies focusing on the improvement of the current best solution were used for this purpose, where the modification of the local search in a hybrid self-adaptive BA algorithm (HSABA) is illustrated in Algorithm 2.

The local search is launched according to a threshold determined by the self-adapted pulse rate r_i (line 1). This parameter is modified by each 10th virtual bat, on average. The local search is an implementation of the operators crossover and mutation borrowed from DE [19]. Firstly, the four virtual bats are selected randomly from the bat population (line 2) and the random position is chosen within the virtual bat (line 3). Then, the appropriate DE strategy modifies the trial solution (lines 4–9) as follows:

$$y_n = \begin{cases} \text{DE\_Strategy}, & \text{if } \text{rand}\,(0,1) \le \text{CR} \vee n = D, \\ x_{\text{in}}^{(t)} & \text{otherwise,} \end{cases} \tag{7}$$

where the DE_Strategy is launched according to the probability of crossover CR. The term $n = D$ ensures that almost

Input: Population $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD}, A, r)^T$ for $i = 1, \ldots, Np$.
Output: Trial solution $\mathbf{y} = (y_1, \ldots, y_D, A, r)^T$.

(1) **if** rand$(0, 1) > r_i$ **then**
(2) $r_{i=1,\ldots,4} = \lfloor \text{rand}(0, 1) * Np + 1 \rfloor \wedge r_1 \neq r_2 \neq r_3 \neq r_4$;
(3) $n = \text{rand}(1, D)$;
(4) **for** $i = 1$ **to** D **do**
(5) **if** ((rand $(0, 1) <$ CR) || $(n == D)$) **then**
(6) $y_n = \text{DE\_Strategy}(n, i, r_1, r_2, r_3, r_4)$;
(7) **end if**
(8) $n = (n + 1)\%(D + 1)$;
(9) **end for** {DE Strategies}
(10) **end if**

Algorithm 2: Modification in hybrid self-adaptive BA algorithm (HSABA).

Table 1: Used DE strategies in the HSABA algorithm.

DE/rand/1/bin	$y_j = x_{r1,j} + F \cdot (x_{r2,j} - x_{r3,j})$
DE/randToBest/1/bin	$y_j = x_{i,j} + F \cdot (\text{best}_j - x_{i,j}) - F \cdot (x_{r1,j} - x_{r2,j})$
DE/best/2/bin	$y_j = \text{best}_j + F \cdot (x_{r1,j} + x_{r2,j} - x_{r3,j} - x_{r4,j})$
DE/best/1/bin	$y_j = \text{best}_j + F \cdot (x_{r1,j} - x_{r2,j})$

one modification is performed on the trial solution. The DE_Strategy function is presented in Algorithm 3.

Although more different DE strategies exist today, we have focused mainly on those that include the current best solution in the modification of the trial solution. These strategies are presented in Table 1. Note that the suitable DE strategy is selected using the global parameter strategy.

The strategies illustrated in the table that use the best solution in the modification operations, that is, "randToBest/1/bin," "best/2/bin," and "best/1/bin," typically direct the virtual bats towards the current best solution. Thus, it is expected that the new best solution is found when the virtual bats move across the search space directed by the best solution. The "rand/1/bin" represents one of the most popular DE strategies today that introduces an additional randomness into a search process. Obviously, it was used also in this study.

3. Experiments

The goal of our experimental work was to show that the HSABA outperforms the results of the original BA as well as the self-adaptive SABA algorithms, on the one hand, and that these results were comparable with the results of other well-known algorithms, like firefly (FA) [5], differential evolution (DE) [19], and artificial bee colony (ABC) [20]. All the mentioned algorithms were applied to a function optimization that belongs to a class of combinatorial optimization problems.

Function optimization is formally defined as follows. Let us assume a fitness function $f(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_D)$ denotes a vector of D design variables from a decision space $x_j \in S$. The values of the design variables are drawn from the interval $x_j \in [lb_j, ub_j]$, where $lb_j \in \mathbb{R}$ and $ub_j \in \mathbb{R}$ are their corresponding lower and upper bounds, respectively. Then, the task of function optimization is to find the minimum of this objective function.

In the remainder of this paper, the benchmark function suite is presented, then the experimental setup is described, and finally, the configuration of the personal computer (PC) on which the tests were conducted is clarified in detail.

3.1. Benchmark Suite. The benchmark suite was composed of ten well-known functions selected from various publications. The reader is invited to check deeper details about test functions in the state-of-the-art reviews [26–28]. The definitions of the benchmark functions are summarized in Table 2 which consists of three columns denoting the function tag f, the *function name*, and the function *definition*.

Each function from the table is tagged with its sequence number from f_1 to f_{10}. Typically, the problem becomes heavier to solve when the dimensionality of the benchmark functions is increased. Therefore, benchmark functions of more dimensions needed to be optimized during the experimental work.

The properties of the benchmark functions can be seen in Table 3 which consists of five columns: the function tag f, the value of the optimal solution $f^* = f(x^*)$, the optimal solution x^*, the function *characteristics*, and *domain*. One of the more important characteristics of the functions is the number of local and global optima. According to this characteristic the functions are divided into either unimodal or multimodal. The former type of functions has only one global optimum, while the latter is able to have more local and global optima throughout the whole search space. Parameter *domain* limits the values of parameters to the interval between their lower and upper bounds. As a matter of fact, these bounds determine the size of the search space. In order to make the problems heavier to solve, the parameter domains were more widely selected in this paper than those prescribed in the standard publications.

3.2. Experimental Setup. The characteristics of the HSABA were observed during this experimental study. Then, the best parameter setting was determined. Finally, the results of the HSABA were compared with the results of the original BA and self-adaptive SABA algorithms. The results of the other well-known algorithms, like FA, DE, and ABC, were also added to this comparative study. Note that the best parameter settings for each particular algorithm were used in the tests, except BA algorithms, where the same setup was employed in order to make the comparison as fair as possible. All the parameter settings were found after extensive testing.

During the tests, the BA parameters were set as follows: the loudness $A_0 = 0.5$, the pulse rate $r_0 = 0.5$, minimum frequency $Q_{\min} = 0.0$, and maximum frequency $Q_{\max} = 2.0$. The same initial values for r_0 and A_0 were also applied by SABA and HSABA, while the frequency was captured from the same interval $Q \in [0.0, 2.0]$ as by the original bat algorithm. Thus, the values for loudness are drawn from the interval $A^{(t)} \in [0.9, 1.0]$ and the pulse rate from the interval $r^{(t)} \in [0.001, 0.1]$. The additional parameters

Input: nth position in the trial solution, indexes of virtual bats i, r_1, r_2, r_3, r_4.
Global: Population $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD}, A, r)^T$ for $i = 1, \ldots, Np$, STRATEGY.
Output: Modified value of the trial solution z.

(1) **switch** (STRATEGY)
(2) **case** DE/rand/1/bin**:**
(3) $z = x_{r1,n} + F * (x_{r2,n} - x_{r3,n})$;
(4) **case** DE/randToBest/1/bin**:**
(5) $z = x_{i,n} + F * (\text{best}_n - x_{i,n}) - F * (x_{r1,n} - x_{r2,n})$;
(6) **case** DE/best/2/bin**:**
(7) $z = \text{best}_n + F * (x_{r1,n} + x_{r2,n} - x_{r3,n} - x_{r4,n})$;
(8) **case** DE/best/1/bin**:**
(9) $z = \text{best}_n + F * (x_{r1,n} - x_{r2,n})$;
(10) **end switch**
(11) **return** z;

Algorithm 3: DE_Strategy function.

Table 2: Definitions of benchmark functions.

f	Function name	Definition
f_1	Griewangk's function	$f(\mathbf{x}) = -\prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + \sum_{i=1}^{n} \frac{x_i^2}{4000} + 1$
f_2	Rastrigin's function	$f(\mathbf{x}) = n * 10 + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$
f_3	Rosenbrock's function	$f(\mathbf{x}) = \sum_{i=1}^{n-1} 100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2$
f_4	Ackley's function	$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left(20 + e^{-20} e^{-0.2\sqrt{0.5(x_{i+1}^2 + x_i^2)}} - e^{0.5(\cos(2\pi x_{i+1}) + \cos(2\pi x_i))}\right)$
f_5	Schwefel's function	$f(\mathbf{x}) = 418.9829 * D - \sum_{i=1}^{D} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$
f_6	De Jong's sphere function	$f(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$
f_7	Easom's function	$f(\mathbf{x}) = -(-1)^D \left(\prod_{i=1}^{D} \cos^2(x_i)\right) \exp\left[-\sum_{i=1}^{D} (x_i - \pi)^2\right]$
f_8	Michalewicz's function	$f(\mathbf{x}) = -\sum_{i=1}^{D} \sin(x_i)\left[\sin\left(\frac{ix_i^2}{\pi}\right)\right]^{2 \cdot 10}$
f_9	Xin-She Yang's function	$f(\mathbf{x}) = \left(\sum_{i=1}^{D} \lvert x_i \rvert\right) \exp\left[-\sum_{i=1}^{D} \sin\left(x_i^2\right)\right]$
f_{10}	Zakharov's function	$f(\mathbf{x}) = \sum_{i=1}^{D} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{D} ix_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{D} ix_i\right)^4$

Table 3: Properties of benchmark functions.

f	f^*	x^*	Characteristics	Domain
f_1	0.0000	$(0, 0, \ldots, 0)$	Highly multimodal	$[-600, 600]$
f_2	0.0000	$(0, 0, \ldots, 0)$	Highly multimodal	$[-15, 15]$
f_3	0.0000	$(1, 1, \ldots, 1)$	Several local optima	$[-15, 15]$
f_4	0.0000	$(0, 0, \ldots, 0)$	Highly multimodal	$[-32.768, 32.768]$
f_5	0.0000	$(0, 0, \ldots, 0)$	Highly multimodal	$[-500, 500]$
f_6	0.0000	$(0, 0, \ldots, 0)$	Unimodal, convex	$[-600, 600]$
f_7	–1.0000	$(\pi, \pi, \ldots, \pi)$	Several local optima	$[-2\pi, 2\pi]$
f_8	–1.8013[1]	$(2.20319, 1.57049)$[1]	Several local optima	$[0, \pi]$
f_9	0.0000	$(0, 0, \ldots, 0)$	Several local optima	$[-2\pi, 2\pi]$
f_{10}	0.0000	$(0, 0, \ldots, 0)$	Unimodal	$[-5, 10]$

[1]Valid for 2-dimensional parameter space.

controlling behavior of DE strategies were set as $F = 0.01$ and CR = 1.0. Both parameters had a great influence on the results of the HSABA algorithm, while this setting initiated the best performance by the self-adaptive SABA framework hybridized with DE strategies.

FA ran with the following set of parameters: $\alpha = 0.1$, $\beta = 0.2$, and $\gamma = 0.9$, while DE was configured as follows: the amplification factor of the difference vector $F = 0.5$ and the crossover control parameter CR = 0.9. In the ABC algorithm, the onlooker bees represented 50% of the whole colony, while the other 50% of the colony was reserved for the employed bees. On the other hand, the scout bee was generated when its value had not improved in 100 generations. In other words, the parameter *limits* was set to the value 100.

Each algorithm in the tests was run with a population size of 100. Each algorithm was launched 25 times. The obtained results of these algorithms were aggregated according to their best, the worst, the mean, the standard deviation, and the median values reached during 25 runs.

3.3. PC Configuration. All runs were made on HP Compaq with the following configurations.

(1) Processor: Intel Core i7-2600 3.4 (3.8) GHz.

(2) RAM: 4 GB DDR3.

(3) Operating system: Linux Mint 12.

Additionally, all the tested algorithms were implemented within the Eclipse Indigo CDT framework.

4. The Results

Our experimental work consisted of conducting the four experiments, in which the following characteristics of the HSABA were tested:

(i) an influence of using the different DE strategies,

(ii) an influence of the number of evaluations,

(iii) an influence of the dimensionality of problems,

(iv) a comparative study.

In the first test, searching was performed for the most appropriate DE strategy incorporated within the HSABA algorithm. This strategy was then used in all the other experiments that followed. The aim of the second experiment was to show how the results of the HSABA converge according to the number of fitness function evaluations. Dependence of the results on the dimensionality of the functions to be optimized is presented in the third experiment. Finally, the results of the HSABA were compared with the original BA and self-adaptive SABA algorithms as well as the other well-known algorithms, like FA, DE, and ABC.

In the remainder of this paper, the results of the experiments are presented in detail.

4.1. Influences of Using the Different DE Strategies. In this experiment, the qualities of four different DE strategies that were implemented within the HSABA algorithm were observed. The outcome of this experiment had an impact on the further tests, because all the tests that followed were performed with the most promising DE strategy found during the experiment. In line with this, each of the DE strategies, that is, "rand/1/bin," "randToBest/1/bin," "best/2/bin," and "best/1/bin," was tested by optimizing the benchmark function suite. The results of function optimization were averaged after 25 runs and are illustrated in Figure 1.

This figure is divided into six diagrams, two for each observed dimension of the function. Each diagram represents the progress of the optimization in percent on the x-axis, where 100% denotes the corresponding number of fitness function evaluations (e.g., 10,000 for 10-dimensional functions), while the value of the fitness function is placed alongside the y-axis.

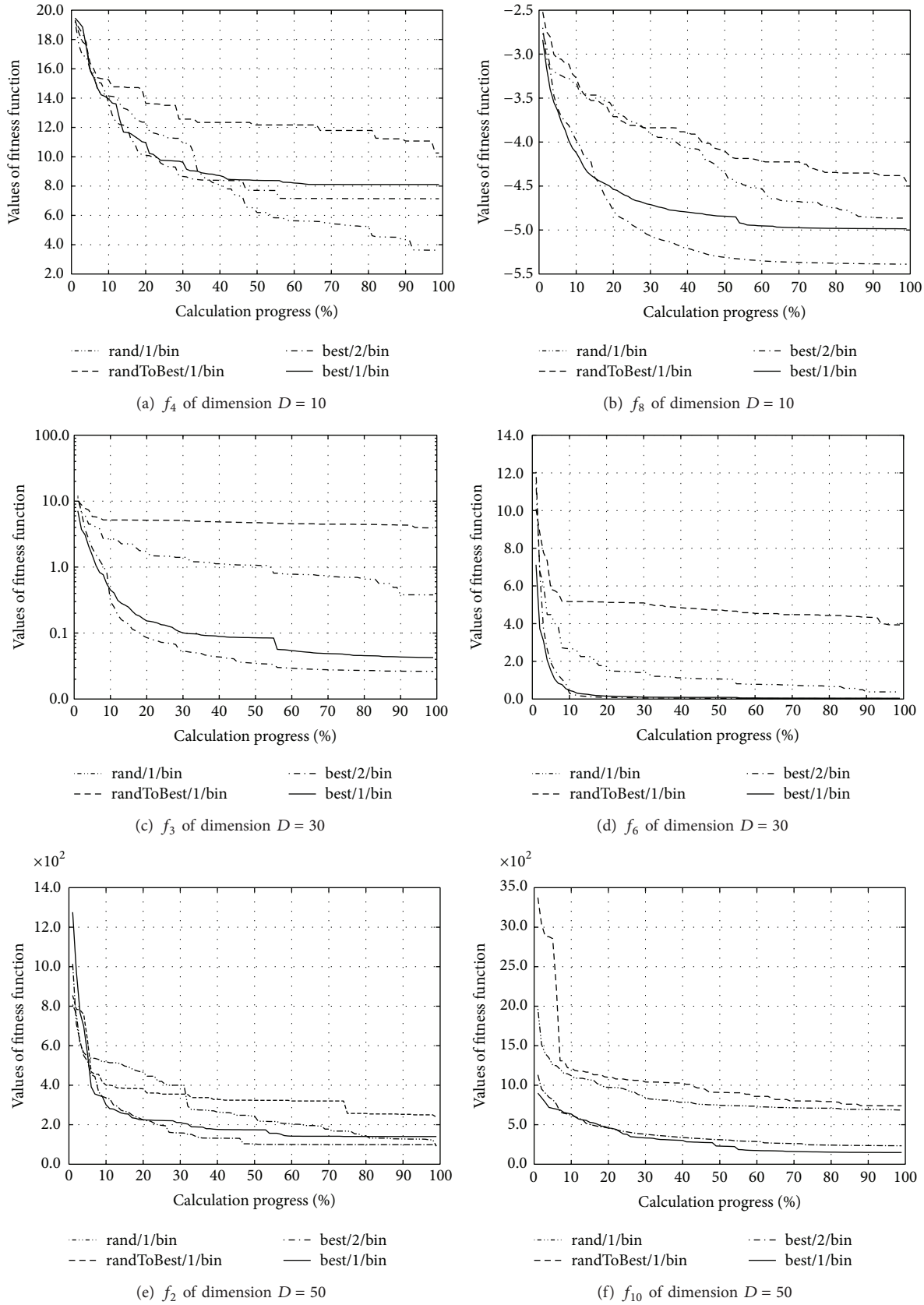
In summary, the "best/2/bin" strategy achieved the best result when compared with the other three strategies, on average. Interestingly, the "rand/1/bin" strategy was the most successful when optimizing the function f_4 of dimension $D = 10$. This behavior was in accordance with the characteristics of this function, because highly multimodal functions demand more randomness during the exploration of the search space.

4.2. Influence of the Number of Evaluations. A long-term lack of improvement regarding the best result during the run was one of the most reliable indicators of the stagnation. If the fitness value did not improve over a number of generations, this probably means that the search process got stuck within a local optimum. In order to detect this undesirable situation during the run of HSABA, the fitness values were tracked at three different phases of the search process, that is, at 1/25, at 1/5, and at the final fitness evaluation. Thus, HSABA runs with the "best/2/bin" strategy and parameters as presented in Section 3.2. The results of this test are collated in Table 4.

It can be seen from Table 4 that HSABA successfully progressed towards the global optimum according to all benchmark functions, except f_7 which fell into a local optimum very quickly and did not find any way of further improving the result.

4.3. Influence of the Dimensionality of the Problem. The aim of this experiment is to discover how the quality of the results depends on the dimension of the problem, in other words, the dimensionality of the functions to be optimized. In line with this, three different dimensions of the benchmark functions $D = 10$, $D = 30$, and $D = 50$ were taken into account. The results of the tests according to five measures are presented in Table 5.

In this experiment, it was expected that the functions of the higher dimensions would be harder to optimize and therefore the obtained results would be worse. The results in Table 5 show that our assumption held for the average fitness values in general, except for function f_9, where optimizing this function of dimension $D = 10$ returned the worst fitness value as by functions of dimensions $D = 30$ and $D = 50$.



(a) f_4 of dimension $D = 10$

(b) f_8 of dimension $D = 10$

(c) f_3 of dimension $D = 30$

(d) f_6 of dimension $D = 30$

(e) f_2 of dimension $D = 50$

(f) f_{10} of dimension $D = 50$

Figure 1: Impact of DE strategies on the results of optimization.

Table 4: Detailed results ($D = 10$).

Evals.	Meas.	f_1	f_2	f_3	f_4	f_5
	Best	1.62$E-$005	8.02$E-$003	1.47$E+$000	1.33$E+$001	1.63$E+$000
	Worst	1.03$E+$00	3.14$E+$02	3.58$E+$005	2.00$E+$001	1.42$E+$003
4.00$E+$02	Mean	4.05$E-$01	8.99$E+$01	1.75$E+$004	1.84$E+$001	3.94$E+$002
	StDev	2.27$E-$01	5.87$E+$01	7.10$E+$001	2.00$E+$001	1.68$E+$002
	Mean	3.81$E-$01	8.06$E+$01	7.14$E+$004	2.47$E+$000	4.64$E+$002
	Best	7.90$E-$06	6.77$E-$03	2.32$E-$002	8.39$E-$003	4.56$E-$003
	Worst	2.18$E-$01	8.96$E+$01	2.47$E+$001	2.00$E+$001	7.77$E+$001
2.00$E+$03	Mean	2.91$E-$02	1.54$E+$01	5.00$E+$000	1.01$E+$001	1.52$E+$001
	StDev	1.52$E-$02	1.00$E+$01	2.01$E+$000	9.49$E+$000	7.34$E+$000
	Mean	5.36$E-$02	2.19$E+$01	6.12$E+$000	7.09$E+$000	1.79$E+$001
	Best	4.87$E-$07	7.33$E-$07	5.51$E-$004	1.05$E-$003	1.75$E-$004
	Worst	1.19$E-$01	2.22$E+$01	8.82$E+$000	2.00$E+$001	1.65$E+$001
1.00$E+$04	Mean	9.35$E-$03	5.81$E+$00	7.13$E-$001	7.14$E+$000	2.52$E+$000
	StDev	1.88$E-$04	9.71$E+$00	8.03$E-$003	6.56$E+$000	5.65$E-$002
	Mean	2.41$E-$02	6.07$E+$00	2.43$E+$000	6.22$E+$000	4.88$E+$000
Evals.	Meas.	f_6	f_7	f_8	f_9	f_{10}
	Best	7.07$E-$05	0.00$E+$00	−4.84$E+$000	5.67$E-$04	6.26$E-$02
	Worst	3.28$E+$00	0.00$E+$00	−2.14$E+$000	3.37$E-$03	9.55$E+$01
4.00$E+$02	Mean	6.98$E-$01	0.00$E+$00	−3.18$E+$000	9.45$E-$04	1.45$E+$01
	StDev	2.62$E-$01	0.00$E+$00	−3.02$E+$000	6.70$E-$04	2.20$E+$00
	Mean	9.46$E-$01	0.00$E+$00	7.59$E-$001	7.50$E-$04	2.35$E+$01
	Best	8.13$E-$06	0.00$E+$00	−6.51$E+$000	5.66$E-$04	6.18$E-$05
	Worst	4.49$E-$02	0.00$E+$00	−3.46$E+$000	6.20$E-$04	4.33$E+$00
2.00$E+$03	Mean	4.75$E-$03	0.00$E+$00	−4.70$E+$000	5.74$E-$04	3.52$E-$01
	StDev	2.18$E-$03	0.00$E+$00	−4.66$E+$000	5.68$E-$04	8.12$E-$02
	Mean	9.10$E-$03	0.00$E+$00	8.55$E-$001	1.32$E-$05	8.74$E-$01
	Best	1.01$E-$09	0.00$E+$00	−7.48$E+$000	5.66$E-$04	2.31$E-$06
	Worst	6.85$E-$03	0.00$E+$00	−3.92$E+$000	5.71$E-$04	7.67$E-$01
1.00$E+$04	Mean	8.92$E-$04	0.00$E+$00	−5.39$E+$000	5.67$E-$04	8.35$E-$02
	StDev	8.47$E-$06	0.00$E+$00	−5.19$E+$000	5.66$E-$04	2.79$E-$02
	Mean	1.76$E-$03	0.00$E+$00	9.49$E-$001	1.29$E-$06	1.84$E-$01

4.4. Comparative Study. The true value of each algorithm is shown for only when we compared it with the other algorithms solving the same problems. Therefore, the HSABA was compared with algorithms such as BA, SABA, FA, DE, and ABC. All the algorithms were solved using the same benchmark functions as proposed in Section 3.1 and used the parameter setting as explained in Section 3.2. The results of this comparative study obtained by optimizing the functions of dimension $D = 30$ are illustrated in Table 6. The best results in these tables are presented as bold.

It can be seen from Table 6 that the HSABA outperformed the results of the other algorithms sixfold (i.e., by f_1, f_2, f_3, f_5, f_9, and f_{10}), DE twice (i.e., by f_4 and f_7), while the other algorithms once (i.e., SABA by f_6 and ABC by f_8). BA and FA did not outperform any other algorithm in the test.

In order to evaluate the quality of the results statistically, Friedman tests [29, 30] were conducted to compare the average ranks of the compared algorithms. Thus, a null hypothesis is placed that states the following: two algorithms are equivalent and therefore their ranks should be equal. When the null hypothesis is rejected, the Bonferroni-Dunn test [31] is performed. In this test, the critical difference is calculated between the average ranks of those two algorithms. If the statistical difference is higher than the critical difference, the algorithms are significantly different.

Three Friedman tests (Figure 2) were performed regarding data obtained by optimizing ten functions of three different dimensions according to five measures. As a result, each algorithm during the tests (also the classifier) was compared with regard to the 10 functions × 5 measures; this means 50 different variables. The tests were conducted at the significance level 0.05. The results of the Friedman nonparametric test can be seen in Figure 1 that is divided into three diagrams. Each diagram shows the ranks and confidence intervals (critical differences) for the algorithms under consideration with regard to the dimensions of the functions. Note that the significant difference between two algorithms is observed if their confidence intervals denoted by thickened lines in Figure 1 do not overlap.

Table 5: Results according to dimensions.

D	Meas.	f_1	f_2	f_3	f_4	f_5
10	Best	$4.87E-07$	$7.33E-07$	$5.51E-04$	$1.05E-03$	$1.75E-04$
	Worst	$1.19E-01$	$2.22E+01$	$8.82E+00$	$2.00E+01$	$1.65E+01$
	Mean	$9.35E-03$	$5.81E+00$	$7.13E-01$	$7.14E+00$	$2.52E+00$
	StDev	$1.88E-04$	$9.71E+00$	$8.03E-03$	$6.56E+00$	$5.65E-02$
	Mean	$2.41E-02$	$6.07E+00$	$2.43E+00$	$6.22E+00$	$4.88E+00$
30	Best	$6.68E-05$	$8.48E-04$	$1.19E-03$	$1.73E-01$	$6.94E-03$
	Worst	$5.58E-01$	$2.69E+02$	$1.57E+03$	$2.00E+01$	$3.57E+03$
	Mean	$7.71E-02$	$4.63E+01$	$1.02E+02$	$9.44E+00$	$2.70E+02$
	StDev	$2.85E-02$	$2.99E+01$	$1.41E+01$	$6.62E+00$	$3.06E+01$
	Median	$1.26E-01$	$7.42E+01$	$3.18E+02$	$6.63E+00$	$7.85E+02$
50	Best	$7.16E-05$	$2.03E-02$	$5.77E-05$	$1.71E-03$	$3.06E-02$
	Worst	$8.81E-01$	$4.48E+02$	$8.08E+02$	$2.00E+01$	$7.47E+03$
	Mean	$1.70E-01$	$9.88E+01$	$7.07E+01$	$1.11E+01$	$9.00E+02$
	StDev	$8.47E-02$	$5.02E+01$	$4.45E+00$	$1.20E+01$	$1.26E+02$
	Mean	$2.29E-01$	$1.27E+02$	$1.89E+02$	$8.59E+00$	$2.12E+03$
Evals.	Meas.	f_6	f_7	f_8	f_9	f_{10}
10	Best	$1.01E-09$	$0.00E+00$	$-7.48E+00$	$5.66E-04$	$2.31E-06$
	Worst	$6.85E-03$	$0.00E+00$	$-3.92E+00$	$5.71E-04$	$7.67E-01$
	Mean	$8.92E-04$	$0.00E+00$	$-5.39E+00$	$5.67E-04$	$8.35E-02$
	StDev	$8.47E-06$	$0.00E+00$	$-5.19E+00$	$5.66E-04$	$2.79E-02$
	Mean	$1.76E-03$	$0.00E+00$	$9.49E-01$	$1.29E-06$	$1.84E-01$
30	Best	$1.96E-08$	$0.00E+00$	$-1.76E+01$	$3.51E-12$	$8.10E-04$
	Worst	$2.17E-01$	$0.00E+00$	$-7.09E+00$	$2.02E-11$	$1.22E+02$
	Mean	$2.63E-02$	$0.00E+00$	$-1.30E+01$	$6.06E-12$	$2.72E+01$
	StDev	$1.29E-03$	$0.00E+00$	$-1.36E+01$	$3.85E-12$	$1.37E+00$
	Mean	$4.82E-02$	$0.00E+00$	$2.10E+00$	$4.16E-12$	$3.90E+01$
50	Best	$4.72E-05$	$0.00E+00$	$-2.83E+01$	$1.21E-20$	$4.57E-01$
	Worst	$9.88E+00$	$0.00E+00$	$-1.40E+01$	$9.57E-20$	$7.43E+02$
	Mean	$5.59E-01$	$0.00E+00$	$-1.95E+01$	$2.44E-20$	$2.34E+02$
	StDev	$1.24E-02$	$0.00E+00$	$-1.91E+01$	$1.63E-20$	$2.35E+02$
	Mean	$1.97E+00$	$0.00E+00$	$3.25E+00$	$1.99E-20$	$2.04E+02$

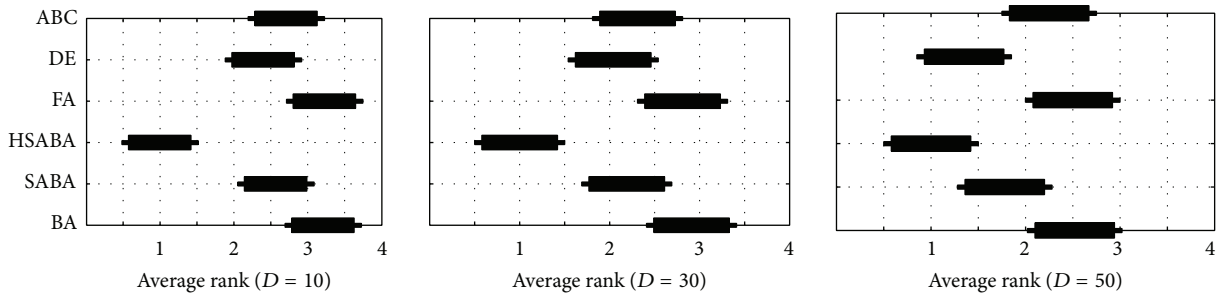


Figure 2: Results of the Friedman nonparametric test on the specific large-scale graph instances.

The first two diagrams in Figure 1 show that the HSABA significantly outperforms the results of the other algorithms by solving the benchmark functions with dimensions $D = 10$ and $D = 30$. Although the other algorithms are not significantly different between each other the algorithms DE, SABA, and ABC are substantially better than the BA and FA. The third diagram shows that the HSABA remains significantly better than the ABC, FA, and BA, while the critical difference between DE and SABA was reduced and therefore becomes insignificant. On the other hand, this reduction caused the difference between DE and the other algorithms in the test, except SABA, to become significant.

Table 6: Obtained results of algorithms ($D = 30$).

F	Meas.	BA	SABA	HSABA	FA	DE	ABC
f_1	Mean	$1.16E+00$	$1.03E+00$	**$7.71E-02$**	$6.65E-01$	$1.05E+00$	$1.09E+00$
	StDev	$1.15E+00$	$1.04E+00$	$2.85E-02$	$6.40E-01$	$2.22E-02$	$1.23E-01$
f_2	Mean	$9.28E+02$	$7.02E+02$	**$4.63E+01$**	$2.44E+02$	$2.28E+02$	$7.33E+01$
	StDev	$8.90E+02$	$6.80E+02$	$2.99E+01$	$2.35E+02$	$1.33E+01$	$2.24E+01$
f_3	Mean	$2.84E+06$	$3.67E+05$	**$1.02E+02$**	$1.12E+02$	$4.57E+02$	$5.18E+02$
	StDev	$2.95E+06$	$2.61E+05$	$1.41E+01$	$1.01E+02$	$2.27E+02$	$4.72E+02$
f_4	Mean	$2.00E+01$	$2.00E+01$	$9.44E+00$	$2.11E+01$	**$1.77E+00$**	$7.17E+00$
	StDev	$2.00E+01$	$2.00E+01$	$6.62E+00$	$2.11E+01$	$3.17E-01$	$1.03E+00$
f_5	Mean	$9.45E+03$	$9.13E+03$	**$2.70E+02$**	$6.78E+03$	$7.57E+03$	$2.64E+03$
	StDev	$9.52E+03$	$9.14E+03$	$3.06E+01$	$6.75E+03$	$4.40E+02$	$3.30E+02$
f_6	Mean	$5.87E-02$	**$1.45E-05$**	$2.63E-02$	$5.19E+00$	$1.77E+02$	$1.63E+02$
	StDev	$6.53E-05$	$1.46E-05$	$1.29E-03$	$5.14E+00$	$7.12E+01$	$1.96E+02$
f_7	Mean	$0.00E+00$	$0.00E+00$	$0.00E+00$	$-3.81E-30$	**$-2.76E-175$**	$-1.76E-136$
	StDev	$0.00E+00$	$0.00E+00$	$0.00E+00$	$-3.73E-30$	$0.00E+00$	$8.79E-136$
f_8	Mean	$-8.62E+00$	$-8.51E+00$	$-1.30E+01$	$-5.15E+00$	$-1.07E+01$	**$-2.30E+01$**
	StDev	$-8.39E+00$	$-8.36E+00$	$-1.36E+01$	$-5.35E+00$	$6.70E-01$	$6.98E-01$
f_9	Mean	$1.57E-11$	$1.41E-11$	**$6.06E-12$**	$1.70E-04$	$2.46E-11$	$1.10E-11$
	StDev	$1.03E-11$	$1.08E-11$	$3.85E-12$	$4.72E-05$	$1.20E-12$	$1.91E-12$
f_{10}	Mean	$2.76E+02$	$2.04E+02$	**$2.72E+01$**	$1.32E+04$	$3.78E+01$	$2.53E+02$
	StDev	$2.82E+02$	$2.17E+02$	$1.37E+00$	$1.32E+04$	$8.74E+00$	$3.15E+01$

5. Conclusion

The original BA algorithm gained better results by optimizing the lower-dimensional problems. When this algorithm was tackled with the harder problems of higher dimensions the results became poorer. In order to improve performance on high dimensional problems, almost two mechanisms are proposed in the publications: firstly, self-adaptation of control parameters and secondly hybridization of the original algorithm with problem-specific knowledge in the form of local search.

In this paper, both mechanisms were incorporated within the original BA algorithm in order to obtain the hybrid self-adaptive HSABA. Thus, the self-adaptive mechanism was borrowed from the self-adaptive jDE, while four different DE strategies were used as a local search. Finally, the evolution of the original BA via self-adaptive SABA and finally to hybrid HSABA is presented in detail.

On the one hand, the experimental work focuses on discovering the characteristics of the HSABA, while on the other hand, it focuses on the comparative study, in which the HSABA was compared with its predecessors BA and SABA as well as with other well-known algorithms, like FA, DE, and ABC. The results of this extensive work showed that the HSABA significantly outperformed the results of all the other algorithms in the tests. Therefore, the assumption taken from publications that self-adaptation and hybridization are appropriate mechanisms for improving the results of population-based algorithms was confirmed.

In the future work, this started work will be continued with experiments on the large-scale global optimization problems.

Conflict of Interests

All authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 687–697, 2008.

[2] I. J. Fister, D. Fister, and X. S. Yang, "A hybrid bat algorithm," *Electrotechnical Review*, vol. 80, no. 1-2, pp. 1–7, 2013.

[3] I. F. Jr, D. Fister, and I. Fister, "A comprehensive review of cuckoo search: variants and hybrids," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 4, pp. 387–409, 2013.

[4] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[5] I. Fister, I. J. Fister, X. S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*, vol. 13, pp. 34–46, 2013.

[6] I. Fister, X. S. Yang, D. Fister, and I. Fister Jr, "Firefly algorithm: a brief review of the expanding literature," in *Cuckoo Search and Firefly Algorithm*, pp. 347–360, Springer, New York, NY, USA, 2014.

[7] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, IEEE, December 1995.

[8] I. J. Fister, X. S. Yang, I. Fister, J. Brest, and D. Fister, "A brief review of nature-inspired algorithms for optimization," *Electrotechnical Review*, vol. 80, no. 3, 2013.

[9] C. Darwin, *The Origin of Species*, John Murray, London, UK, 1859.

[10] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, The MIT Press, Cambridge, Mass, USA, 1992.

[11] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*, Springer, Berlin, Germany, 2003.

[12] I. Rechenberg, *Evolutionstrategie: Optimirung Technisher Systeme Nach Prinzipen des Biologischen Evolution*, Fromman-Hozlboog, Stuttgard, Germany, 1973.

[13] I. Fister, M. Mernik, and B. Filipič, "Graph 3-coloring with a hybrid selfadaptive evolutionary algorithm," *Computer Optimization and Application*, vol. 54, no. 3, pp. 741–770, 2013.

[14] C. Blum and X. Li, "Swarm intelligence in optimization," in *Swarm Intelligence: Introduction and Applications*, C. Blum and D. Merkle, Eds., pp. 43–86, Springer, Berlin, Germany, 2008.

[15] I. Fister Jr., X. S. Yang, K. Ljubic, D. Fister, J. Brest, and I. Fister, "Towards the novel reasoning among particles in pso by the use of rdf and sparql," *The Scientific World*. In press.

[16] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NISCO '10)*, C. Cruz, J. Gonzlez, G. T. N. Krasnogor, and D. A. Pelta, Eds., vol. 284 of *Studies in Computational Inteligence*, pp. 65–74, Springer, Berlin, Germany, 2010.

[17] I. J. Fister, S. Fong, J. Brest, and F. Iztok, "Towards the self-adaptation in the bat algorithm," in *Proceedings of the 13th IASTED International Conference on Artificial Intelligence and Applications*, 2014.

[18] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[19] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution a Practical Approach to Global Optimization*, Springer, New York, NY, USA, 2005.

[20] D. Karaboga and B. Akay, "A comparative study of Artificial Bee Colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[21] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[22] M. Črepinšek, M. Mernik, and S.-H. Liu, "Analysis of exploration and exploitation in evolutionary algorithms by ancestry trees," *International Journal of Innovative Computing and Applications*, vol. 3, no. 1, pp. 11–19, 2011.

[23] S. Rao, *Engineering Optimization: Theory and Practice*, John Wiley & Sons, New York, NY, USA, 2009.

[24] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[25] S. Wright, "The roles of mutation, inbreeding, crossbreeding, and selection in evolution," in *Proceedings of the 6th International Congress on Genetics*, pp. 355–366, 1932.

[26] M. Jamil and X. S. Yang, "A literature survey of benchmark functions for global optimisation problems," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.

[27] X. S. Yang, "Appendix A: test problems in optimization," in *Engineering Optimization*, X. S. Yang, Ed., pp. 261–266, John Wiley & Sons, Hoboken, NJ, USA, 2010.

[28] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.

[29] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, pp. 675–701, 1937.

[30] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *The Annals of Mathematical Statistics*, vol. 11, pp. 86–92, 1940.

[31] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.