Research Article

Ensemble bat algorithm based on Hyper heuristic approach for solving unconstrained optimization problems

Wakas S. Khalaf[1]

[1]Department of Industrial Management, College of Economics and Administration, University of Baghdad

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 28 April 2021

Abstract Maintaining convergence and diversification in solving optimization is one of the most important challenges facing metaheuristic algorithms in general and the bat algorithm in particular. Many researchers have suggested some improvements to preserve the ability of the algorithm to find good solutions in a timely manner and also to move away as much as possible from landing on the local optimization zone. In this paper, a hyper-heuristic method was proposed to incorporate the behavior of three optimized algorithms from the bat algorithm. The method is based on the distribution of a specific implementation probability for each used algorithm and then updating this probability iteratively according to the results of each algorithm, and then we use random selection to determine the algorithm used in the current iteration. Some nonlinear models proposed in CEC2005 used to compare the efficiency of the proposed algorithm and compare its results with some state-of-the-art algorithms.

Keywords: Bat algorithm, Ensemble strategy, unconstrained optimization problems, Benchmark function

1. Introduction

Many scientific and engineering problems can be turned into numerical optimization problems for solving functions. A nonlinear unconstrained optimization problem can be defined as follows without sacrificing generality:

$$\min f(x)$$
$$s.t. \quad li \leq xi \leq ui \qquad i = 1,2,\dots,n \qquad (1)$$

Among them, $f(x)$ is the objective function, $x = (x_1, x_2, \dots, x_n)$ Î Rn is the n-dimensional decision variable, and li and ui are the upper and lower bounds of the variable xi respectively.

The conventional optimization approach based on gradient knowledge is difficult to find an effective solution for problem (1) because it is typically highly nonlinear. Intelligent optimization algorithms like the genetic algorithm, particle swarm optimization algorithm, differential evolution algorithm, ant colony algorithm, artificial immune algorithm, and others, when compared to conventional optimization methods, are more efficient. are a form of global search method that uses population iteration and has no problem requirements. It is high and does not require the problem's gradient knowledge, and it can converge to the problem's global optimal solution with a high probability. As a result, it's common in unconstrained optimization [1-5].

Yang [9] proposed the Bat Algorithm (BA), a multitude insight enhancement calculation that is utilized to display the characteristic pursuit of bats utilizing ultrasound. The streamlining issue's answer is alluded to as a bat in the pursuit space in BA. A wellness esteem is doled out to each bat. The bats can change the recurrence of their heartbeats, To adjust the best bat at present in the arrangement region as far as volume and outflow rate. Search for something. In taking care of unconstrained advancement issues, BA has been demonstrated to be better than Genetic algorithm (GA) and particle swarm optimization (PSO) calculations [9]. Therefore, BA can be utilized in an assortment of settings, including mathematical improvement, designing enhancement, obliged streamlining, and creation arranging issues.

Nonetheless, like other multitude knowledge streamlining calculations, essential BA likewise has inadequacies, for example, being inclined to fall into neighborhood optima and defer late combination. Focusing on the insufficiencies of fundamental BA, an incorporated cross breed BA is proposed to tackle the issue of unconstrained advancement. To consolidate different bat calculation renditions and super-heuristic stage disorder, establish the framework for improving the calculation variety, in this way instating the position and speed of bats. To organize the worldwide and nearby inquiry capacities of the calculation, a latency weight is presented in the speed update equation. Play out a Powell search on the flow best bat to accelerate union. The reenactment after effects of a few standard test capacities demonstrate the adequacy of the calculation.

In this paper, we will combine algorithms BA-DTFS proposed in [4] and BA-DTFS proposed in [5] and MBA proposed in [33] to propose a new hybrid algorithm from the bat search algorithms. The merging mechanism will depend on a hyper-heuristic approach to distribute the probability of implementation on each algorithm used, with the development of criteria for comparing the algorithm's results at each stage with some randomly generated values.

The paper is organized as follows. Section 2 describes the basic Bat algorithm with a detailed on algorithm and presents the proposed Ensemble of BAT algorithm variants (EBATV) with discerption . Section 3 provides details of benchmarking problems, parameter setting of the algorithms and compares their results Section 4 provides on results and discussion results .

2. Methodology

2.1 BAT Algorithm

Yang [9] proposed BA, a heuristic intelligent search algorithm inspired by bats' use of echolocation for predator detection. To solve the problem, first map a single bat to specific points in the search space, then use the position of a single bat in the search space as the fitness function's value. Optimize the objective purpose of the problem, The search process of the algorithm is modeled after that of bats looking for food and flying. The basic steps of the BA algorithm are as follows:

Step 1: Make t = 1 and initialize the algorithm's parameters.

Step 2: Randomly generate solutions x_i^t and in the velocity for each solution v_i^t.

Step 3: Determine the fitness value of each solution and the population's best solution.

Step 4: Check to see if the algorithm satisfies the termination condition (whether it reaches the maximum number of iterations). The algorithm ends when it is satisfied, and the best solution is output. If not, proceed to step 5.

Step 5: Update the solution's velocity and position using equations (2), (3), and (4):

$$f_i = f_{min} + (f_{max} - f_{min}) . \beta \qquad (2)$$
$$v^{t+1}{}_i = v^t{}_i + (x_i^t - x^*) . f_i \qquad (3)$$
$$x^{t+1}{}_i = v^{t+1}{}_i + x_i^t \qquad (4)$$

Where f_i is the *i-th* bat's pulse frequency, f_{min} and f_{max} are the pulse frequency's minimum and maximum values, respectively, and is a uniformly distributed random number on [0, 1],], $v^{t+1}{}_i$and $v^t{}_i$ respectively. Is the i-th bat's flight speed in generations t + 1 and t, and $x_i^t$is the *i-th* bat's location in t, x^* The ideal position of the bat in the current community is $t,$ and the *i-th* bat at the position of t + 1 generation is $x^{t+1}{}_i$.

Step 6: Rand1 is a random number generator. If rand1> ri (ri is the *i-th* bat's pulse frequency), then perturb the current optimal bat location to get a new one, and then replace the old one.

Step 7: Generate a random number rand2. If $rand1 > Ai$ (Ai is the pulse intensity of the ith bat) and $f\ (xi)\ < f\ (x\ *)$, then move to the updated position.

Step 8: When the condition of Step 7 is satisfied, the pulse frequency r and pulse intensity A are updated according to equations (5) and (6):

$$r_i^{t+1} = r_i^0\left[1 - e^{(-\gamma\times t)}\right] \qquad (5)$$
$$A_i^{t+1} = \alpha . A_i^t \qquad (6)$$

In the formula, r_i^{t+1} denotes the *ith* bat's pulse frequency at t + 1 generation, r i0 denotes the *ith* bat's maximum pulse frequency, > 0 denotes the pulse frequency increase factor, A_i^{t+1} and A_i^t denote the sound strength of the *ith* bat transmitting pulses at generations t + 1 and t, respectively, , $\alpha \in [0,1]$ the pulse intensity attenuation; otherwise, let t = t + 1 and return to step 3, The flow of the standard Bat algorithm is as follows.

Algorithm 1: Standard BAT Algorithm

```
Begin
Set the relevant parameters for each bat.
Calculate each solution's fitness values;
While (The stopping condition of the algorithms is satisfied)
        Using Eq.4, update the position of each bat.
Assess the new position's fitness requirements;
If the current position's fitness is lower than the previous position's,
            Remove the old solution and replace it with the new one;
      End If
      Choose the best particle and save it;
End While
         Provide the best solution;
End
```

2.2 Ensemble of BAT algorithm variants (EBATV)

In this study, in order to collect multiple BAT variants, a multi-population framework (MPF) was proposed. MPF does not implement the combination of Population into subpopulation (PAP) algorithm through plan for time allocation particles immigration administrators and their strategies [20], but divides the entire PAP, including several indicator subpopulations and reward subpopulations. The indicator subgroups are the same size as the incentive subgroups, but they are much smaller. There is an indicator subgroup for each variant that makes up BAT. Any certain number of generations after the creation of the integration algorithm, the reward subgroup will be adaptively assigned to the BAT best performance variable. In this way, different BAT variants will evolve together, and the variant with the best performance during the evolution will get the most (total) resources.

2.2.1 Constituent BAT variants

In order for EBATV to work better, the constituent variables of BAT must be both strong and have different functions, so that they can assist each other in the evolution process, not just in competing resources. Many studies have shown that using different operators in algorithms is important [80-82]. Three highly efficient BAT variants, namely BA-DTFS [4], BAGW [5], and MBA [33], are used as synthesis algorithms in this study. Attempting to incorporate each variant of BAT into the integration process is impossible. MBA is a generic BAT variant, according to observational analysis, so these three algorithms were chosen as components. It typically dominates other BAT variables when solving the single peak optimization problem, and BAGW is solving some simple problems. Multimodal optimization is a very successful problem. BA-DTFS has shown exceptional success in the optimization of highly complex compounds [2, 24]. The following is a quick rundown of the three BAT variants.

1. BA-DTFS

Triangle-flipping technique in the Bat algorithm (BA-DTFS),initially developed by *Cai* and et.al [4], is a simple but efficient BAT variant. A new position update strategy is used in BA-DTFS. The position category is listed below.

$$x^{t+1}{}_i = x^* + (x^{worst} - x_i^t).f_i \qquad (7)$$
$$x^{t+1}{}_i = x^t{}_i + (x^{worst} - x^*).f_i \qquad (8)$$
$$x^{t+1}{}_i = x^{worst} + (x^t{}_i - x^*).f_i \qquad (9)$$

In the above three triangle inversion strategy, all bats appear in those bats, and their positions have not been determined in the previous generation. (3) And (4). All bats can use triangle rotation strategy to update their speed and position.

2. BAGW

Bat algorithm with Gaussian walk (BAGW) Proposed by *Wang* and et. al [5] In this variant, Gaussian traces are used instead of the original uniform traces in local turbulence to improve local search capabilities. In addition, in order to maintain a higher pumping pressure, The speed update equation was also changed as a result of the increased strain. Finally, in order to maximize population diversity, the frequency is determined by each size and varies depending on the various bats in our modification. In the regular version, the location and velocity are described below.

$$v^{t+1}{}_i = (x_i^t - x^*).f_i \qquad (10)$$

$$x^{t+1}{}_i = x^* + \eta\mu \qquad (11)$$

The random numbers sampled by a standard Gaussian distribution are used among them. According to the study above, a large number of numbers are distributed in order to maximize the chances of escaping the local optimal value.

3. MBA

Bat calculation has been changed (MBA) The point of Ramli and et a proposition . [33] is to improve the investigation and utilization of the bat calculation to accomplish a quicker assembly speed. By consolidating new versatile size adjustments and new inertial weight changes, this can be refined.

Inertial weight influences the speed condition, which thusly influences the whole BA measure. The inertial weight esteem relies upon the speed. Speed can be estimated by the distance between the current best position and the current situation at emphasis t. The inactivity esteem ceaselessly diminishes alongside cycles and merges to emphasis t, which shows that the bat is nearer to acquiring prey (arrangement). Determined as follows:

$$w_t = (t_{max} - t).\sqrt{(f(x^t) - f(x^*))^2} \qquad (12)$$

$$v^{t+1}{}_i = v^t{}_i.w_t + (x_i^t - x^*).f_i \qquad (13)$$

2.2.2 Multi-Swarm based ensemble framework

The MPF divides the population into several indicator subgroups (each individual belongs to the constitutive BAT variant) and rewards them. We divide the entire population into three subgroups of indicators and a reward population because the MPF contains three BAT algorithm variants, namely BA-DTFS, BAGW, and MBA. Each generation activates the partition operator. P_1, P_2, and P_3 represent the three indicator subgroups, while P_4 represents the incentive subgroup. The index subgroups are all the same size. The indicator subgroup is much smaller than the reward subgroup in terms of size. Let pop represent the entire population. We've got

$$P = \bigcup_{i=1}^{4} P_i \qquad (14)$$

Let NP be the size of the P and NP_ibe the size of the P_i. w_i indicates the proportion between P_i and P. So we have

$$NP_i = w_i * NP \qquad (15)$$

$$\sum_{i=1}^{4} w_i = 1 \qquad (16)$$

We just have $w_1 = w_2 = w_3$. here. Each indicator subpopulation is assigned to a constituent BAT variant at random, and the reward subpopulation is also assigned to a BAT variant at random. The population partition procedure is done once for each generation. After each number of generations, the algorithm continues, Based on the relationship between the cumulative fitness improvements and the evaluations of the functions consumed, we evaluate the most effective BAT variant ((j_{best}) over the last century.

Algorithm 2: Algorithm Framework of EBATV.

```
1.  Initial parameters of EBATV including $ng$, $NP$, $wi$, $MaxG$ and $MaxFes$;
2.  Initial the parameters for BA-DTFS, BAGW and MBA;
3.  Set $\Delta \boldsymbol{f_i} = \mathbf{0}$ and $\Delta \boldsymbol{fes_i}$ for $i = 1, 2, 3$;
4.  Initialize the $P$ randomly distributed in the solution space;
5.  Set $NP_i = w_i * NP$ ;
6.  Randomly divide $P$ into $P_1$, $P_2$, $P_3$and $P_4$with respect to their sizes;
7.  Randomly select a subpopulation $P_i$ ($i = 1, 2, 3$) and combine $P_i$ with $P_4$. Let $P_i = P_i \cup P_4$
    and $NP_i = NP_i + NP_4$;
8.  Set $g = 0$;
9.  **while** $g \leq MaxG$ do
10. $g = g + 1$;
11. Execute BA-DTFS on $P_1$, update$P_1$ and calculate $\Delta \boldsymbol{f_1}$;
12. Execute BAGW on $P_2$, update $P_2$ and calculate $\Delta \boldsymbol{f_2}$;
13. Execute MBA on $P_3$, update$P_3$ and calculate $\Delta \boldsymbol{f_3}$;
14. Combine updated $P_1$, $P_2$ and $P_3$ into $P$, i.e., $P = \cup_{i=1}^{3} P_i$;
15. **if** $mod(g, ng) == 0$ **then**
16. $k = arg(max_{i=1,2,3}(\frac{\Delta f_i}{ng.NP_i}))$;
17. **end if**
18. Randomly partition $P$ into $P_1$, $P_2$, $P_3$and $P_4$4;
19. Let $P_k = P_k \cup P_4$, $k \in \{1, 2, 3\}$;
20. **End while**
```

$$j_{best} = \boldsymbol{argmax}_{i=1,2,3}\left(\frac{\Delta \boldsymbol{f_i}}{\Delta \boldsymbol{fes_i}}\right) \qquad (17)$$

where $\Delta \boldsymbol{f_i}$ is the cumulative function fitness improvements attributed by the *ith* constituent BAT variant over the last ng generations, and $\Delta \boldsymbol{fes_i}$ is the consumed number of function evaluations. The reward subpopulation will be awarded to the highest performing constituent BAT variant for the next ng generations. The above-mentioned best-performing BAT variant determination and reward subpopulation assignment operators are run on a regular basis, with n g being the time. We ensure that the best upgrade velocity approach uses the most computational resources with this definition. Algorithm 2 describes the EBATV algorithm architecture.

3. Experimental study

I. Benchmark problem and comparative algorithms

"Any elevated performance over one class of problems is exactly compensated for in performance over another class," according to the No Free Lunch theorem. In other words, a meta-heuristic can perform

admirably on one set of problems while performing poorly on another set of problems. Without a doubt, Every year, the NFL keeps this area of study active, resulting in improvements to existing methods and the introduction of new meta-heuristics. We used a broad collection of standard benchmark functions, which are listed in Table I, to completely evaluate the output of the EBATV algorithm without coming to a biased conclusion about any specific problems. The ten benchmark functions are divided into three categories: unimodal functions (F_1 to F_7) and multimodal functions (F_8 to F_{13}).Despite the fact that this set of benchmark functions has been widely adopted by other researchers [3,] their dimensions are relatively small (up to 30) as compared to those of real-world optimization problems. Figure 1 shows the space of variables and objective function for each problem.

Comparative algorithms

We put EBATV to the test with 10 and 30 variables on the CEC2005 [39] benchmark problems. BAT [9], BA-DTFS [4], BAGW [5], and MBA [33] are some of the more recent algorithms that have been chosen as comparatives. PSO [38], PSO [38], PSO [38], PSO [38 Every comparative algorithm's parameters are identical to those in the original paper. For all algorithms, the population size is 20 and the number of function evaluations is 5000. the outcomes of 30 simulation runs on ten and thirty-dimensional problems

$Test\ Function$	n	S	f_{min}
$F_1(x) = \sum_{i=1}^{n} {x_i}^2$	0	[−100,100]	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	0	[−10,10]	0
$F_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	0	[−100,100]	0
$F_4(x) = max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	0	[−100,100]	0
$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - {x_i}^2)^2 + (x_i - 1)^2]$	0	[−30,30]	0
$F_6(x) = \sum_{i=1}^{n} [(x_i + 0.5)^2]$	0	[−100,100]	0
$F_7(x) = \sum_{i=1}^{n} i{x_i}^4 + random(0,1)$	0	[−1.28,1.28]	0
$F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert}$	0	[−500,500]	-12,569.487
$F_9(x) = \sum_{i=1}^{n} [{x_i}^2 - 10\cos(2\pi x_i) + 10]$	0	[−5.12,5.12]	0
$F_{10}(x) = 20 \exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{n} {x_i}^2}{n}}\right) - \exp\left(\frac{\cos(2\pi x_i)}{n}\right) + 20 + e$	0	[−32,32]	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} {x_i}^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{i}\right) + 1$	0	[−600,600]	0
$F_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	0	[−50, 50]	0
$F_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	0	[−50,50]	0

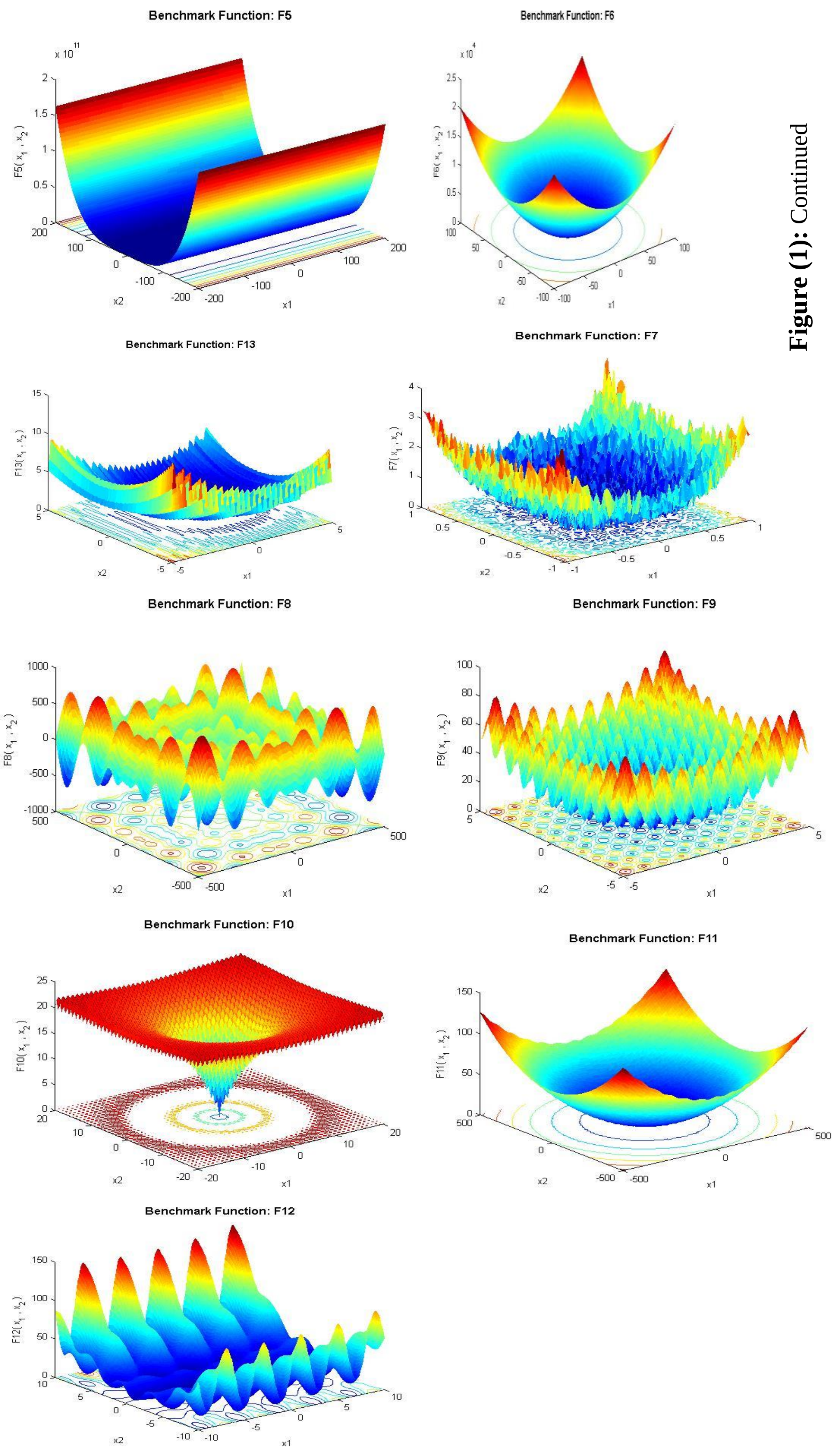

Figure (1): Continued

Figure (1): Continued

Results and discussion results

Table (2)

F	EBATV	BAT	BA-DTFS	BAGW	MBA	PSO
$F1$	**0 (0)**	4.256907 (0.853176)	5.27E-09 (1.20E-08)	0.010641 (0.002843)	3.608328 (1.165552)	0.001183 (0.000896)
$F2$	**0 (0)**	38.91408 (13.61953)	1.80E+01 1.14E+01)	0.024057 (0.005229)	7.92319 (1.293073)	38.73776 (39.26777)
$F3$	**0 (0)**	55102.18 (8468.547)	11456.74 (6234.571)	38801.77 (5446.878)	14.82813 (9.467119)	4556.914 (2101.319)
$F4$	**0 (0)**	53.16875 (5.370203)	9.92344 (2.532465)	16.3683 (1.621525)	0.836258 (0.050729)	17.46107 (3.047367)
$F5$	**1.62E-11 (4.75E-11)**	6002.508 (3391.25)	27336.77 (43265.75)	25.68089 (0.37179)	125.1801 (36.43224)	71.69537 (97.10704)
$F6$	**0 (0)**	5.894768 (1.212925)	3.31E-09 (5.35E-09)	0.011301 (0.003368)	17.03625 (3.616703)	0.00146 (0.000946)
$F7$	**5.83E-06 (4.09E-06)**	0.193691 (0.056757)	0.566405 (1.698824)	0.05396 (0.012901)	10.39285 (2.744191)	0.234667 (0.092227)
$F8$	-8360.08 (97.3293)	**-9.98E+3 (2.13E+3)**	-9476.92 (533.1439)	-7143.01 (204.1972)	-116.961 (0.434419)	-5472.81 (7.52E+01)
F9	**0 (0)**	204.4521 (11.61949)	97.9248 (32.26454)	134.7475 (12.49901)	56.85804 (14.6505)	82.48186 (21.29521)
F10	**8.88E-16 (0)**	2.800032 (0.276656)	2.06E-05 (2.86E-05)	0.0362327 (0.0055153)	2.656609 (0.200383)	3.65361 (1.57675)
F11	**0 (0)**	1.040986 (0.017718)	0.0105738 (0.0119873)	0.102645 (0.122785)	0.163409 (0.057642)	0.071146 (0.028119)
F12	**1.57E-32 (2.88E-48)**	31232.32 (27016.25)	0.1299941 (0.2318388)	0.034312 (0.01729)	1.447892 (0.683756)	12.2892 (6.110684)
F13	2.778063 (0.594543)	165577.7 (127488.9)	**2.20E-03 (4.63E-03)**	0.106449 (0.026226)	0.080988 (0.049143)	27.81043 (17.1326)

Table (3)

F	EBATV	BAT	BA-DTFS	BAGW	MBA	PSO
F1	1	0.997956	1	0.999995	0.998268	0.999999
F2	1	0.20894	0.634089	0.999511	8.39E-01	0.212524
F3	1	0	0.792082	0.295822	0.999731	0.917301
F4	1	0.157781	0.842808	0.740718	0.986753	0.723408
F5	1	0.780424	0	0.999061	0.995421	0.997377
F6	1	0.997066	1	0.999994	0.99152	0.999999
F7	1	0.981364	0.945501	0.994809	0	0.977421
F8	0.670388	0.802131	7.61E-01	0.571408	0	0.435575
F9	1	0	5.21E-01	0.340934	0.7219	0.596571
F10	1	0.800043	0.999999	0.997413	0.810285	0.739087
F11	1	0.963474	0.999629	0.996398	0.994266	0.997504
F12	1	0	0.999996	0.999999	0.999954	0.999607
F13	0.999983	0	1	0.999999	1	1.00E+00
SUM	12.670371	6.6892	10.4961	10.9361	10.3371	10.5964
Rank	1	6	2	5	4	3
No.best	11/13	0/13	3/13	0/13	1/13	1/13

Result of comparative between the algorithms

Table (4)

F	EBATV	BAT	BA-DTFS	BAGW	MBA	PSO
F1	1	0.999796	1	0.999999	0.999721	1
F2	1	0.653163	0.709686	0.999867	0.96707	0
F3	1	0.362296	0.53052	0.589836	0.999287	0.841765
F4	1	0.770297	0.891677	0.930642	0.99783	0.869653
F5	1	0.921618	0	0.999991	0.999158	0.997756
F6	1	0.999712	1	0.999999	0.999143	1

F7	1	0.994493	0.835146	0.998748	0.733703	0.991051
F8	0.9545	0	0.749851	0.904317	1	0.964892
F9	1	0.639868	0	0.612608	0.545926	0.339981
F10	1	0.965371	0.999996	0.99931	0.974918	0.802637
F11	1	0.99959	0.999723	0.997158	0.998666	0.999349
F12	1	0	0.999991	0.999999	0.999975	0.999774
F13	0.999995	0	1	1	1	0.999866
SUM	12.954495	8.306204	9.71659	12.0324	12.2153	10.8067
Rank	1	4	3	2	6	5
No.best	11/13	7/13	8/13	9/13	3/13	5/13

Discussion Result

Table (2) shows that the EBATV generated better result than all algorithms in function $F1- F4$ *and* $F6$ the EBATV algorithm get the optimal solution in standard deviations equal to zero that meaning the EBATV algorithm get the optimal solution in all running. in function $F1- F2$ the best result after EBATV was BA-DTFS algorithm and the worst result in BAT algorithm. And in function $F3$ the best result after EBATV found by using MBA and the worst result found by using BAT, And in function $F4$ the best result after EBATV found by using MBA and the worst result found by using BAT, in function $F5$ the best result found by EBATV algorithm and then BAGW algorithm and worst result found by BA-DTFS , in function $F6$ the best result after EBATV found by using BA-DTFS and the worst result found by using MBA , in function $F7$ the best result found by EBATV algorithm and then BAGW algorithm and worst result found by MBA.

Additionally in correlation with the outcomes delivered by EBATV. From Table (2), it is obvious to see that for F9 to F12 of the test benchmark capacities, EBATV extraordinarily beat different calculations. For instance, on work F11, EBATV tracked down the worldwide least in completely run while different calculations created less fortunate outcomes for this situation and in work F9 EBATV tracked down the worldwide least in totally run and different calculations produced exceptionally less fortunate outcomes for this situation, and on work F10 and F12 EBATV tracked down the best outcome thought about on different calculations while in F10 the close to result to EBATV found by BA-DTFS.in work F8 EBATV was beated by PSO,MBA and BAGW, and in work F13 EBATV was outflanked by PSO, BAT, while all calculations were outflanked in other four capacities and EBATV get the best outcome in other four capacities.

Finally why using normalize for all result in 13 functions to determine the best algorithms using the following equation:

$$Norm = \frac{max - x}{max - min} \qquad (18)$$

Where x is the value of objective obtained using any algorithm and max the maximum value obtained when solving function using all algorithms and min the minimum value obtained when solving function using all algorithms when the algorithm gets the norm=1 that meaning the algorithm gets the best result compared with the other algorithms and when the norm=0 that meaning the algorithm gets the worst result compared with the other algorithms. In table (3) we calculate the normalize for the average result for all function and we can found the EBATV comparison with the other algorithm ranked the first algorithm and obtained for the best result in 11 functions from 13 and we can see that the order of the search performance of these ten algorithms is EBATV>BA-DTFS>PSO>MBA>BAGW>BAT

In table (4) we calculate the normalize for the standard deviation result for 50 runs for all function and we can found the EBATV comparison with the other algorithm ranked the third algorithm while obtained for the best SD in 11 functions from 13 and the BA-DTFS obtained the best SD 9 only and MBA in 12 we can see that the order of the search performance of these ten algorithms is EBATV>BAGW> BA-DTFS >BAT>PSO>MBA.

4. Conclusion

To all the more likely address the UCOPs, a coordinated BA is proposed. three improved forms are considered in the combination technique. Likewise, one determination instrument with steady likelihood is utilized to change the likelihood of every procedure. To demonstrate the predominance of the calculation, the test work is utilized to contrast EBATV and different calculations. The test results show that the effectiveness of the calculation is improved. In future exploration, the exhibition of the calculation will proceed to improve and can be applied to different applications.

Future trends recommend using the multiplicity of societies with the bat algorithm, relying on some of the mechanism for dividing societies, and the possibility of using the technique of hyper-extension in directing societies and using some improvement processes to obtain highly efficient solutions.

References

A. A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, Evol. Comput., IEEE Trans. 13 (2) (2009) 398–417.

B. C. Li, S. Yang, T.T. Nguyen, A self-learning particle swarm optimizer for global optimization problems, Syst., Man, Cybern., Part B: Cybern., IEEE Trans. 42 (3) (2012) 627–646.

C. C.P. Gomes, B. Selman, Algorithm portfolios, Artif. Intell. 126 (1) (2001) 43–62.

D. Cai, Xingjuan, et al. "Bat algorithm with triangle-flipping strategy for numerical optimization." International Journal of Machine Learning and Cybernetics 9.2 (2018): 199-215.

E. Cai, Xingjuan, et al. "Bat algorithm with Gaussian walk." International Journal of Bio-Inspired Computation 6.3 (2014): 166-174.

F. E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, J.R. Woodward, A classification of Hyper-Heuristic approaches, in: Handbook of Metaheuristics, Springer, 2010, pp. 449–468.

G. E.K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, Hyper-heuristics: a survey of the state of the art, J. Oper. Res. Soc. 64 (12) (2013) 1695–1724.

H. Hamzaçebi, C. (2008). Improving genetic algorithms’ performance by local search for continuous function optimization. Applied Mathematics and Computation, 196(1), 309-317.

I. Yang XS (2010) A new metaheuristic bat-inspired algorithm. Comput Knowl Technol 284:65–74

J. Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, Evol. Comput., IEEE Trans. 15 (1) (2011) 55–66.

K. Zhang Zhihui，Zhang Jun，Li Yun，et al.Orthogonal learning particle swarm optimization[J].IEEE Transactions on Evolutionary Computation，2011，15（6）：832-847.

L. R. Mallipeddi, P.N. Suganthan, Q.-K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft. Comput. 11 (2) (2011) 1679–1696.

M. S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, Evol,. Comput., IEEE Trans. 15 (1) (2011) 4–31.

N. Wang Yong，Cai Zixing，Zhang Qingfu.Differential evolution with composite trial vector generation strategies and control parameters[J].IEEE Transactions on Evolutionary Computation，2011，15（1）：55-66.

O. W. Gong, Á. Fialho, Z. Cai, H. Li, Adaptive strategy selection in differential evolution for numerical optimization: an empirical study, Inf. Sci. (Ny) 181 (24) (2011) 5364–5386.

P. W. Gong, A. Zhou, Z. Cai, A multioperator search strategy based on cheap surrogate models for evolutionary optimization, Evol. Comput., IEEE Trans. 19 (5) (2015) 746–758.

Q. S.-Z. Zhao, P.N. Suganthan, Q. Zhang, Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes, Evol. Comput., IEEE Trans. 16 (3) (2012) 442–446

R. X. Chen, Y.-S. Ong, M.-H. Lim, K.C. Tan, A multi-facet survey on memetic computation, IEEE Trans. Evol. Comput. 15 (5) (2011) 591–607. 186 G. Wu et al. / Information Sciences 423 (2018) 172–186

S. P. Cowling, G. Kendall, E. Soubeiga, A Hyperheuristic Approach to Scheduling a Sales Summit, in: Practice and Theory of Automated Timetabling III, Springer, 2001, pp. 176–190.

T. F. Peng, K. Tang, G. Chen, X. Yao, Population-based algorithm portfolios for numerical optimization, Evol. Comput., IEEE Trans. 14 (5) (2010) 782–800.

U. Yang X S.Nature inspired meta-heuristic algorithms[M]. 2nd ed.Frome，UK：Luniver Press，2010：97-104.

V. R. Mallipeddi, P.N. Suganthan, Ensemble of constraint handling techniques, IEEE Trans. Evol. Comput. 14 (4) (2010) 561–579.

W. R. Mallipeddi, S. Mallipeddi, P.N. Suganthan, Ensemble strategies with adaptive evolutionary programming, Inf. Sci. (Ny) 180 (9) (2010) 1571–1581.

X. G. Jia, Y. Wang, Z. Cai, Y. Jin, An improved (μ+ λ)-constrained differential evolution for constrained optimization, Inf. Sci. (Ny) 222 (2013) 302–322.

Y. H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, J.-s. Pan, Multi-strategy ensemble artificial bee colony algorithm, Inf. Sci. (Ny) 279 (2014) 587–603.

Z. G. Xiong, D. Shi, X. Duan, Multi-strategy ensemble biogeography-based optimization for economic dispatch problems, Appl. Energy 111 (2013) 801–811.

AA. P. Moscato, et al., On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms, Caltech concurrent computation program, C3P Report (1989) 826 (1989).

BB. N. Krasnogor, J. Smith, A tutorial for competent memetic algorithms: model, taxonomy, and design issues, Evol. Comput., IEEE Trans. 9 (5) (2005) 474–488.

CC. Y.-S. Ong, M.-H. Lim, N. Zhu, K.-W. Wong, Classification of adaptive memetic algorithms: a comparative study, Systems, Man, Cybern., Part B: Cybern., IEEE Trans. 36 (1) (2006) 141–152.

DD. K. Tang, F. Peng, G. Chen, X. Yao, Population-based algorithm portfolios with automated constituent algorithms selection, Inf. Sci. (Ny) 279 (2014) 94–104.

EE. Yang X S，Gandomi A H. Bat algorithm：a novel approach for global engineering optimization[J].Engineering Computation，2012，29（5）：464-483.

FF. Gandomi A H，Yang X S，Alavi A H，et al.Bat algorithm for constrained optimization tasks[J].Neural Computing & Applications，2013，22（6）：1239-1255.

GG. Ramli, M. R., et al. "Enhanced convergence of Bat Algorithm based on dimensional and inertia weight factor." Journal of King Saud University-Computer and Information Sciences 31.4 (2019): 452-458.

HH. S.-Z. Zhao, P.N. Suganthan, Q. Zhang, Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes, Evol. Comput., IEEE Trans. 16 (3) (2012) 442–446.

II. W. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Inf. Sci. (Ny) 178 (15) (2008) 3096–3109.

JJ. Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, Q. Tian, Self-adaptive learning based particle swarm optimization, Inf. Sci. (Ny) 181 (20) (2011) 4515–4538.

KK. C. Li, S. Yang, T.T. Nguyen, A self-learning particle swarm optimizer for global optimization problems, Syst., Man, Cybern., Part B: Cybern., IEEE Trans. 42 (3) (2012) 627–646.

LL. Kennedy, James, and Russell Eberhart. "Particle swarm optimization." Proceedings of ICNN'95-International Conference on Neural Networks. Vol. 4. IEEE, 1995.

MM. Suganthan, Ponnuthurai N., et al. "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization." KanGAL report 2005005.2005 (2005): 2005.