Choosing population sizes to enhance Brain Storm Optimization algorithms

Ricardo García-Ródenas, Luis Jiménez Linares, and Julio Alberto López-Gómez
{*Ricardo.Garcia, Luis.Jimenez, JulioAlberto.Lopez*}*@uclm.es*

***Abstract*—Nature-inspired algorithms (NIA) are a very powerful tool to solve plenty of complex science and engineering problems. In the last ten years, a new kind of NIA algorithms, so-called human-inspired algorithms, has arisen in optimization tasks obtaining promising results. In this paper, the effect of population size in Brain Storm Optimization algorithm (BSO) is studied with the purpose of choosing good enough population sizes which improve its performance. Moreover, it is also analyzed the possibility of using population samples instead of the whole population, studying its performance in terms of time and computational cost. To do that, hybrid functions of IEEE CEC competitions have been used as benchmark problems. The results show that twenty five individuals is a good enough population size in BSO algorithms while population sampling improves the performance of the algorithm.**

***Index Terms*—Nature-inspired algorithms, Human-inspired algorithms, Brain storm optimization, Population size, Population sampling**

I. Introduction

Traditionally, complex optimization problems have required effective and sophisticated methods to deal with them. In this context, Nature-Inspired Algorithms (NIA) appeared in the 1960s. One of the main features of NIA algorithms is they take their knowledge, contents and procedures departing from nature. Thus, algorithms like Particle Swarm Optimization (PSO) [1], Genetic Algorithm (GA) [2] or Ant Colony Optimization (ACO) [3] among others, have been widely applied in operational research.

During the last ten years, there has been a change in the source of inspiration to build NIA algorithms. Nowadays there is a trend which tries to model the behaviour of humans in problem solving. Along this line, a new kind of NIA algorithms has appeared which is called Human-Inspired Algorithms (HIA). There are currently six HIA algorithms which are the following: Seeker Optimization Algorithm (SOA) [4], Imperialist Competitive Algorithm (ICA) [5] , Social Emotional Optimization Algorithm (SEOA) [6], Teaching Learning-Based Optimization (TLBO) [7], Team Effectiveness Based Optimization (TEBO) [8] and Brain Storm Optimization algorithm (BSO) [9] which is the algorithm used in this paper.

This paper has two targets: On the one hand, it tries to choose good enough population sizes in Brain Storm Optimization algorithm when it is applied to hybrid functions. On the other hand, it studies whether is better to use the whole population in the algorithm or only a sample of it, in terms of computational cost and quality of solutions.

The rest of the document is structured as follows: section 2 studies the related work in BSO algorithm. Section 3 describes BSO algorithm and ADMBSO algorithm, which is the version used in the experiments of this paper. Later, section 4 defines the experiments carried out and studies the results obtained and finally, section 5 introduces the conclusions about this work and some future works.

Ricardo García-Ródenas, Department of Mathematics at University of Castilla la Mancha, Spain

Luis Jiménez Linares, Department of Technology and Information Systems of University of Castilla la Mancha, Spain

Julio Alberto López Gómez, Department of Technology and Information Systems at University of Castilla la Mancha, Spain.

II. Related Work

To the best of our knowledge, there are 75 papers, 8 theses and 5 patents in total on the development and application of the BSO algorithm. The current research in BSO is concerned with three main research lines which are the following:

- **Study and analysis of the algorithm.** In this line, all studies about new brainstorming operators [10], modifications [11], clustering methods [12], parameters adjustment [13], hybridizations [14], etc are grouped.
- **Algorithm's variants.** New variants of brain storm algorithms are required to solve successfully different complex optimization problems like multimodal [15], hybrid or multiobjective problems [16], among others.
- **BSO applications.** There is a lack of BSO applications to real world problems, although there are many applications in fields like wireless sensor networks [17], multiple UAV formation flights [18], stock index forecasting [19] and economic dispatch [20].

III. Brain Storm Optimization Algorithms

Brain Storm Optimization (BSO) algorithm is a human-inspired algorithm which was born in 2011 and developed by Shi [9]. This NIA algorithm is inspired by the human brainstorming process. In it, a group of people or brainstorming group proposes different ideas to solve a problem. Then, new ideas are generated from the existing ones in order to reach the optimum idea. The brainstorming process ends when an optimum idea has been achieved.

This way, brainstorming departs from a set of random initial solutions. After initialization, new ideas are generated departing from the existing ones through convergence and divergence operations. Convergence operation is focused on searching new solutions in a set of search areas where good solutions are found. Formally, these areas corresponds with

local minima in the objective function. To carry out convergence operation, cluster analysis techniques have been widely applied, using mainly the $k-$means algorithm. Moreover, divergence operation is in charge of building new solutions. In brainstorming, new solutions can be generated departing from one or more existing solutions. For simplicity, BSO algorithm considers only two possibilities: generating a new solution departing from one existing solution or two. The first option corresponds with local search in an optimization algorithm and the second is related to exploration. The kind of generation will be made using a set of probabilities. After BSO, different variants of the algorithm have appeared with the purpose of improving BSO capabilities and guaranteeing a good tradeoff between exploration and exploitation. In this paper, Advanced Discussion Mechanism based on Brain Storm Optimization Algoritm (ADMBSO) [11] is used instead of the original BSO. ADMBSO has been used since it is the most recent and standard version of BSO algorithm. In it, exploration is encouraged in the first iterations and exploitation in the last iterations. Moreover, ADMBSO adds different rules to generate new ideas in each epoch in intra and inter cluster generation. On the one hand, inter cluster generation will be possible in this algorithm building a new idea from two ideas in the same cluster. Briefly, this algorithm is shown in Algorithm 1.

Algorithm 1 Advanced Discussion Mechanism based on BSO algorithm

1: POPULATION INITIALIZATION: Initialize n random solutions in population pop.
2: **while** Termination Condition is not satisfied **do**
3: $P_{intra} = P_{low} + P_{high}\frac{N_{cur}gen}{N_{max}gen}$
4: $P_{inter} = 1 - P_{intra}$
5: /* Update ϵ */
6: $\epsilon(t) = logsig\left(\frac{\frac{max_{iter}}{2} - n_{iter}}{k}\right) * random(t)$
7: POPULATION EVOLUTION: Generate n new ideas departing from the current population
8: **BEGIN Converging Operation**
9: Group pop in m clusters
10: **END Converging Operation**
11: **for** $i = 1$ to $card(pop)$ **do**
12: **BEGIN Diverging Operation**
13: **if** $rand() < P_{intra}$ **then**
14: /* Generate individuals from one cluster*/
15: **if** $rand() < p_{ceni}$ **then**
16: /* Add noise to center of the cluster */
17: $c^i_{new} = c^i_{old} + \epsilon(t) * random(t)$
18: **else if** $rand() < p_{indi}$ **then**
19: /*Combine two ideas from this cluster and add noise */
20: Select x_{old1} and x_{old2}
21: $x^i_{new} = (x^i_{old1} - x^i_{old2}) * random(t)$
22: **else**
23: /*Add noise to a random idea*/
24: $x^i_{new} = x^i_{old} + \epsilon(t) * random(t)$
25: **end if**
26: **else**
27: /*Generate individuals from two clusters*/
28: **if** $rand() < p_{cenii}$ **then**
29: /*Select two centroids c_{old1}, c_{old2}, combine them and add noise/*
30: $x^i_{old} = w_1 * c^i_{old1} + w_2 * c^i_{old2}$
31: $x^i_{new} = (c^i_{old1} - c^i_{old2}) * random(t)$
32: **else if** $rand() < p_{indii}$ **then**
33: /*Generate randomly an idea depart from two individuals of two clusters/*
34: Select x_{old1} and x_{old2}
35: $x^i_{old} = w_1 * x^i_{old1} + w_2 * x^i_{old2}$
36: $x^i_{new} = (x^i_{old1} - x^i_{old2}) * random(t)$
37: **else**
38: /* Generate randomly a new idea */
39: **end if**
40: **end if**
41: **END Diverging Operation**
42: REPLACE MECHANISM: If the new solution built is better than the current one, substitute it, otherwise, maintain the current solution
43: /* Update Solution */
44: **if** $f(x_{new}) < f(x_{old})$ **then**
45: $x_{old} = x_{new}$
46: **end if**
47: **end for**
48: **end while**

IV. Experiments

In this section, the two experiments carried out to choose good enough population sizes in Advanced BSO algorithm (ADMBSO) are described in detail.

A. *Experiment* 1*: Choosing population size in BSO*

The problem of choosing a good enough population size for a NIA algorithm is difficult to address. The reason is because it is generally problem-dependent and in general, the settings of one algorithmic parameter could be related to the settings of other algorithmic parameters, in terms of optimization performance. Besides, it depends on the particular version of BSO used, the dimensionality problem and the complexity of the search space. That is why, in this experiment, different population sizes widely applied in other algorithms have been tested over ADMBSO algorithm and over a set of concrete benchmark functions, in this case, hybrid functions from IEEE CEC conference competition.

In order to recommend good population sizes for BSO algorithms, the definition of robustness analysis proposed in [21] is taken. According to that, an optimization algorithm is robust if inequality

$$|f(x^*_{alg}) - f(x^*)| < \epsilon_{rel} \cdot |f(x^*)| + \epsilon_{abs} \qquad (1)$$

holds, where x^*_{alg} is the best value reached by the algorithm, x^* is a global minimum of the objective function f (it is assumed, given) and ϵ_{rel}, ϵ_{abs} are two parameters which control the accuracy of the algorithm.

Thus, different population sizes can be proposed and tested with different values of ϵ_{abs} in order to detect if different

population sizes are capable of reaching determined levels of accuracy. Moreover, the convergence speeds of the same algorithm with different population sizes can also be compared using the expression proposed in [21] and shown in equation 2:

$$s = \frac{NFE_{pop1}}{NFE_{pop2}} \quad (2)$$

where s is the convergence speed ratio, NFE_{pop1} is the number of functions evaluations which the algorithm with the population size $pop1$ needs to satisfy inequality (1) and NFE_{pop2} is the same using $pop2$. To do that, three ϵ_{abs} values have been proposed in order to test the performance and robustness of different population sizes while ϵ_{rel} has been fixed to $10e^{-4}$ like in [21]. These values are $\epsilon_{abs1} = 10e^{-1}, \epsilon_{abs2} = 10e^{-3}, \epsilon_{abs3} = 10e^{-5}$

Moreover, four population sizes have been tried in this experiment. Firstly, one hundred ideas has been taken as a reference value, since it is used in most of the articles published in BSO and cited here. After that, and following the fact that lower values are efficient in other metaheuristics like in PSO, other population sizes have been proposed. Thus, pop_{25} has twenty five ideas or individuals, pop_{50} has fifty ideas, pop_{75} has seventy five individuals and pop_{100} has one hundred ideas. The sample of benchmark functions consists of eight objective functions with thirty dimensions taken from IEEE CEC conference competitions. They are well-known optimization problems. In this paper, we have focused on hybrid functions, which are the most complex benchmarks in CEC competition. They are the following: rotated hybrid composition function 1, rotated hybrid composition function 1 with noise in fitness, rotated hybrid composition 2, rotated hybrid composition 2 with a narrow basin for the global optimum, rotated hybrid composition 3, rotated hybrid composition 3 with high condition number matrix, rotated hybrid composition function 4 and rotated hybrid composition function 4 without bounds. These problems are named as P1,P2,...,P8.

Then, ADMBSO algorithm has been executed for each population size using the parameters employed in [11]. The configuration is the following: $m = 5, P_{cen} = 0.7, P_{ind} = 0.2, P_{rnd} = 0.1, P_{cens} = 0.7, P_{low} = 0.2, P_{high} = 0.7$. The total number of function evaluations computed in each execution is 100000. Ten trials have been run for each population size in order to minimize the random effect in the experiment.

Average values of mean, maximum, minimum, variance and standard deviation have been obtained for each population size. The total time spent in all executions is reported too. These results are shown in table I. The minimum values for each measure are typed in bold. Thus, it is possible to see that pop_{25} needs less time than the rest of populations to achieve its results and it is the population which reaches the minimum values regarding the rest of population sizes in more test functions. Moreover, figure 1 shows the temporal cost of each population size for each problem. This way, it is possible to see that pop_{25} provides the best results regarding quality of solution and temporal cost.

Furthermore, regarding robustness analysis, table II shows the convergence speed of different population sizes in comparison with the population of one hundred individuals (used in [11] and considered as reference). Note that problems 3 and 4 do not appear in this table because they did not achieve any robustness threshold. In this manner, when s values are less than 1, it means that the proposed configuration is better than pop_{100} taken as reference. If it is greater than 1, the performance of pop_{100} is better. The best rates are typed in bold. It is possible to conclude that all the sizes tested in this paper are better than p_{100} used in the most of articles published about BSO.

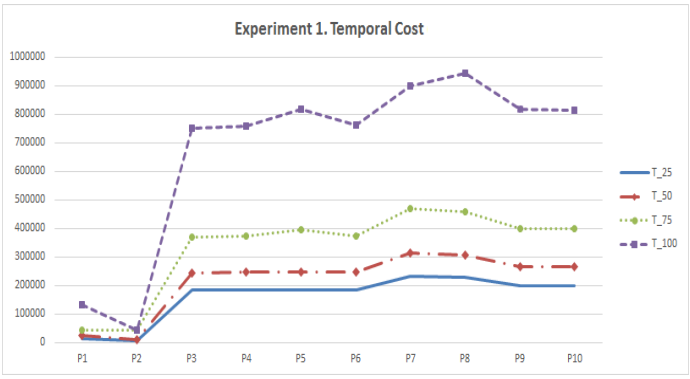

Fig. 1. Temporal Cost for different population sizes

TABLE II
Analysis of convergence speed in ADMBSO algorithm

	s1	s2	s3
P1_25	**0,86**	-	-
P1_50	0,90	-	-
P1_75	0,93	-	-
P2_25	-	-	-
P2_50	**0,96**	-	-
P2_75	0,98	-	-
P5_25	**0,56**	-	-
P5_50	0,75	-	-
P5_75	0,99	-	-
P6_25	**0,59**	-	-
P6_50	1,04	-	-
P6_75	0,77	-	-
P7_25	**0,72**	0,95	
P7_50	0,82	1,01	1,01
P7_75	1,09	**0,89**	**0,90**
P8_25	1,05	1,19	1,17
P8_50	**0,92**	**0,64**	**0,63**
P8_75	1,01	0,96	0,94

B. Experiment 2: Population sampling in BSO

One solution to improve the quality of population could be to choose a sample of the population in the search process. It would imply a better initial population but include the problematic of choosing a good sample which combines good fitness values in the individuals chosen and good ideas diversity. Besides, the sampling process adds computational cost to the algorithm.

To do that, a multinomial probability distribution can be established according to the fitness values. This way, it will determine the probability of choosing an individual in the population according the fitness values of the whole population.

TABLE I
ADMBSO STATISTICAL RESULTS FOR HYBRID CEC BENCHMARK FUNCTIONS

	Mean	Best	Worst	Variance	Std	Time
P1_25	**2,93E+02**	2,08E+02	1,55E+03	1,79E+04	1,33E+02	**1,82E+05**
P1_50	2,96E+02	2,21E+02	**1,10E+03**	**1,28E+04**	**1,13E+02**	2,44E+05
P1_75	3,57E+02	**1,66E+02**	1,06E+03	3,86E+04	1,96E+02	3,68E+05
P1_100	3,38E+02	2,21E+02	1,123E+03	1,54E+04	1,24E+02	7,50E+05
P2_25	3,94E+02	2,81E+02	1,54E+03	3,87E+04	1,96E+02	**1,84E+05**
P2_50	3,74E+02	2,40E+02	**1,21E+03**	3,68E+04	1,91E+02	2,46E+05
P2_75	3,76E+02	2,38E+02	1,54E+03	2,93E+04	1,71E+02	3,72E+05
P2_100	**3,58E+02**	**2,21E+02**	1,39E+03	**2,68E+04**	**1,64E+02**	7,59E+05
P3_25	9,01E+02	9,00E+02	1,45E+03	3,17E+02	1,7E+01	**1,84E+05**
P3_50	9,00E+02	9,00E+02	1,02E+03	**5,70E+01**	**7,59E+00**	2,46E+05
P3_75	8,74E+02	8,57E+02	1,41E+03	1,98E+03	4,45E+01	3,93E+05
P3_100	**8,68E+02**	**8,53E+02**	**9,06E+02**	3,21E+01	1,79E+01	8,17E+05
P4_25	9,00E+02	9,00E+02	**9,18E+02**	**5,39E+00**	**2,32E+00**	**1,84E+05**
P4_50	8,78E+02	8,66E+02	1,17E+03	2,73E+02	1,65E+01	2,46E+05
P4_75	**8,63E+02**	**8,47E+02**	1,39E+03	2,39E+02	4,89E+01	3,74E+05
P4_100	8,82E+02	8,66E+02	1,43E+03	8,33E+02	2,88E+01	7,63E+05
P5_25	**7,11E+02**	5,15E+02	1,48E+03	7,75E+ 04	2,78E+02	**2,33E+05**
P5_50	7,43E+02	**5,03E+02**	1,54E+03	8,17E+04	2,85E+02	3,12E+05
P5_75	8,91E+02	5,99E+02	**1,44E+03**	8,48E+04	2,91E+02	4,70E+05
P5_100	7,93E+02	5,33E+02	1,48E+03	**6,72E+04**	**2,59E+02**	8,99E+05
P6_25	**6,64E+02**	**5,38E+02**	2,48E+03	6,65E+04	2,57E+02	**2,29E+05**
P6_50	8,14E+02	6,69E+02	2,08E+03	**5,42E+04**	**2,32E+02**	3.07E+05
P6_75	7,54E+02	5,83E+02	1,91E+03	6,97E+04	2,64E+02	4,60E+05
P6_100	7,93E+02	6,07E+02	**1,87E+03**	6,90E+04	2,62E+02	9,42E+05
P7_25	**4,20E+02**	**2,29E+02**	1,57E+03	**1,37E+05**	**3,70E+02**	**1,97E+05**
P7_50	4,52E+02	2,40E+02	1,53E+03	1,57E+05	3,96E+02	2,64E+05
P7_75	5,25E+02	2,39E+02	1,52E+03	1,97E+05	4,44E+02	3,99E+05
P7_100	4,90E+02	2,39E+02	**1,52E+03**	1,72E+05	4,15E+02	8,17E+05
P8_25	4,12E+02	2,31E+02	1,65E+03	1,22E+05	3,49E+02	**1,98E+05**
P8_50	**4,03E+02**	**2,30E+02**	**1,48E+02**	1,31E+05	3,62E+02	2,66E+05
P8_75	4,17E+02	2,37E+02	1,51E+03	**1,15E+05**	**3,40E+02**	4,00E+05
P8_100	4,35E+02	2,44E+02	1,525E+03	1,28E+05	3,58E+02	8,15E+05

Thus, the probability of choosing an individual i is computed through equation.

$$P(i) = \frac{\frac{1}{f(i)}}{\sum_{j=1}^{n} \frac{1}{f(j)} + \epsilon} \quad (3)$$

where n is the population size, $f(i)$ is the fitness value for individual i and ϵ is a value near to zero to avoid to divide between 0. In previous experiment pop_{25} has been chosen as the best option in terms of population size. Now, ADMBSO algorithm has been executed using a population of one hundred individuals, but in this case, only twenty five individuals are selected to execute the algorithm. The selection mechanism is based on the multinomial probability distribution.

Now, ADMBSO algorithm with a sampled population of twenty five individuals have been executed under the same conditions as Experiment 1. Mean values of mean, minimum, maximum, variance and standard deviation as well as total time (in seconds) have been computed. These results are shown in table III. The results which improve the original results of pop_{25} are typed in bold.

With regard to the results, it can be noted this way of sampling population improves the original results of pop_{25}. Concretely, the major improvements are made in the last two problems which corresponds with the most difficult in CEC competitions (where pop_{25} did not improve the results of the rest of population sizes) and it is now the best configuration. Furthermore, it can be seen that the time is not improved in this case. To show that, figure 2 shows the temporal cost of the population sizes made in experiment 1 in comparison with the temporal cost of population sampling.

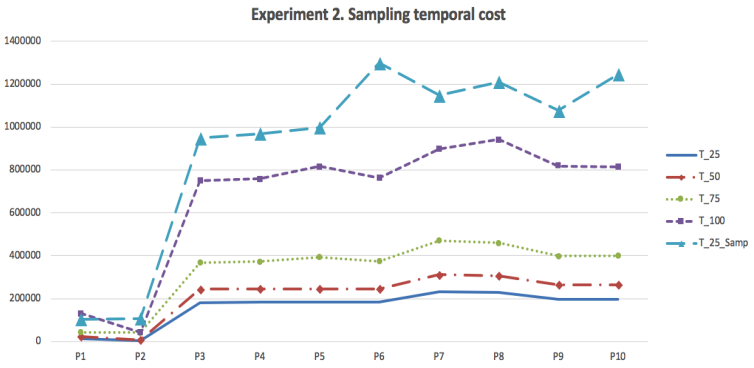

Fig. 2. Temporal Cost with original population sizes and sampling

Regarding robustness analysis, table IV shows the convergence speed (s) of sampling population with respect to pop_{100} which was considered as reference. Once again, values which are equals or better than pop_{25} convergence speeds are typed in bold. The results show that sampling population increase the convergence speed of the algorithm in comparison with the original version.

V. Conclusions and further works

In this article the problem of choosing good enough population sizes in BSO algorithms, has been carried out. To do

TABLE III
STATISTICAL RESULST OF POPULATION SAMPLING

	Mean	Best	Worst	Variance	Std	Time
P1_25_Sam	**2,91E+02**	2,27E+02	**9,11E+02**	**1,42E+04**	**1,19E+02**	9,49E+05
P2_25_Sam	**3,63E+02**	**2,81E+02**	**1,12E+02**	**1,46E+04**	**1,20E+02**	9,68E+05
P3_25_Sam	**9,00E+02**	**9,00E+02**	**1,29E+03**	**2,44E+02**	**1,56E+01**	9,97E+05
P4_25_Sam	**9,00E+02**	**9,00E+02**	9,84E+01	7,51E+00	2,74E+00	1,29E+06
P5_25_Sam	7,14E+02	5,28E+02	1,56E+02	7,897E+04	2,81E+02	1,14E+06
P6_25_Sam	8,17E+02	6,78E+02	**1,76E+03**	**4,51E+02**	**2,12E+02**	1,21E+06
P7_25_Sam	**3,76E+02**	**2,26E+01**	1,64E+03	**9,26E+04**	**3,04E+02**	1,07E+06
P8_25_Sam	**3,82E+02**	**2,63E+01**	**1,604E+02**	**9,84E+04**	**3,13E+02**	1,24E+05

TABLE IV
CONVERGENCE SPEED OF SAMPLING POPULATION

	s1	s2	s3
P1_Sam	**0,55**	0,88	**0,86**
P2_Sam	**0**	-	-
P3_Sam	0,87	-	-
P4_Sam	-	-	-
P7_Sam	**0,53**	**-**	**-**
P8_Sam	**0,95**	**-**	**-**
P9_Sam	**0,70**	**0,66**	**0,66**
P10_Sam	**0,85**	**0,66**	**0,65**

that, different population sizes have been defined in order to study the solutions achieved and the time spent for each size. After that, the best population size has been chosen and an strategy of sampling population has been carried out in order to study if this strategy improves the results of the original algorithm. The main conclusions reached are the following:

- Despite the fact that one hundred individuals have been widely used in the majority of articles published about BSO algorithms, it has been demonstrated that less individuals guarantee better results in terms of quality of solution in hybrid functions.
- Sampling population is a good solution to enhance the results of the original algorithm in terms of solutions achieved in the case of hybrid functions.
- Sampling population strategy speeds up the convergence speed of ADMBSO algorithm when it is applied over hybrid functions.

As future works, these experiments can be applied over different benchmark functions and new sampling strategies can be studied in order to reduce the temporal cost of the algorithm, as well as speeding up NIA algorithms through local search procedures in order to guarantee better robustness thresholds.

ACKNOWLEDGMENT

This research was supported by *Ministerio de Economía, Industria y Competitividad*-FEDER EU grants with number TRA2016-76914-C3-2-P

REFERENCES

[1] Y. Shi and R. C. Eberhart, "A Modified Particle Swarm Optimizer," in *Proceedings of IEEE International Conference on Evolutionary Computation*. IEEE Computer Society, May 1998, pp. 69–73.

[2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[3] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: Optimization by a colony of cooperating agents," *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART B*, vol. 26, no. 1, pp. 29–41, 1996.

[4] C. Dai, Y. Zhu, and W. Chen, "Seeker optimization algorithm," in *Computational Intelligence and Security*, Y. Wang, Y.-m. Cheung, and H. Liu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 167–176.

[5] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition," *2007 IEEE Congress on Evolutionary Computation*, pp. 4661–4667, 2007.

[6] Z. Cui, Z. Shi, and J. Zeng, "Using social emotional optimization algorithm to direct orbits of chaotic systems," in *Swarm, Evolutionary, and Memetic Computing*, B. K. Panigrahi, S. Das, P. N. Suganthan, and S. S. Dash, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 389–395.

[7] R. Rao, V. Savsani, and D. Vakharia, "Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems," *Computer-Aided Design*, vol. 43, no. 3, pp. 303 – 315, 2011.

[8] X. Feng, M. Ji, Z. Li, X. Qu, and B. Liu, "Team effectiveness based optimization," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, June 2017, pp. 2248–2257.

[9] Y. Shi, "Brain storm optimization algorithm," in *Advances in Swarm Intelligence*, Y. Tan, Y. Shi, Y. Chai, and G. Wang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 303–309.

[10] D. Zhou, Y. Shi, and S. Cheng, "Brain storm optimization algorithm with modified step-size and individual generation," in *Advances in Swarm Intelligence*, Y. Tan, Y. Shi, and Z. Ji, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 243–252.

[11] Y. Yang, Y. Shi, and S. Xia, "Advanced discussion mechanism-based brain storm optimization algorithm," vol. 19, 10 2015.

[12] J. Chen, Y. Xie, and J. Ni, "Brain storm optimization model based on uncertainty information," in *2014 Tenth International Conference on Computational Intelligence and Security*, Nov 2014, pp. 99–103.

[13] Z. Zhan, W. Chen, Y. Lin, Y. Gong, Y. Li, and J. Zhang, "Parameter investigation in brain storm optimization," in *2013 IEEE Symposium on Swarm Intelligence (SIS)*, April 2013, pp. 103–110.

[14] R. García-Ródenas, L. J. Linares, and J. A. López-Gómez, "A cooperative brain storm optimization algorithm," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, June 2017, pp. 838–845.

[15] X. Guo, Y. Wu, and L. Xie, "Modified brain storm optimization algorithm for multimodal optimization," in *Advances in Swarm Intelligence*, Y. Tan, Y. Shi, and C. A. C. Coello, Eds. Cham: Springer International Publishing, 2014, pp. 340–351.

[16] X. Guo, Y. Wu, L. Xie, S. Cheng, and J. Xin, "An adaptive brain storm optimization algorithm for multiobjective optimization problems," in *Advances in Swarm and Computational Intelligence*, Y. Tan, Y. Shi, F. Buarque, A. Gelbukh, S. Das, and A. Engelbrecht, Eds. Cham: Springer International Publishing, 2015, pp. 365–372.

[17] R. Ramadan and A. Khedr, "Brain storming algorithm for coverage and connectivity problem in wireless sensor network," 04 2016.

[18] H. Qiu, H. Duan, and Y. Shi, "A decoupling receding horizon search approach to agent routing and optical sensor tasking based on brain storm optimization," *Optik - International Journal for Light and Electron Optics*, vol. 126, no. 7, pp. 690 – 696, 2015.

[19] J. Wang, R. Hou, C. Wang, and L. Shen, "Improved v -support vector regression model based on variable selection and brain storm optimization for stock price forecasting," *Applied Soft Computing*, vol. 49, pp. 164 – 178, 2016.

[20] K. R. Ramanand, K. R. Krishnanand, B. K. Panigrahi, and M. K. Mallick, "Brain storming incorporated teaching–learning–based algorithm with application to electric power dispatch," in *Swarm, Evolutionary, and Memetic Computing*, B. K. Panigrahi, S. Das, P. N. Suganthan, and P. K. Nanda, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 476–483.

[21] F. Kang, J. Li, and Z. Ma, "Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions," *Information Sciences*, vol. 181, no. 16, pp. 3508 – 3531, 2011.