Date of publication MAR-30, 2022, date of current version MAR-02, 2022.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.21.1.2519

A Real-Value Parameter Function Optimization Algorithm using Repeated Adaptive Local Search

SURAPONG AUWATANAMONGKOL
School of Applied Statistics, National Institute of Development Administration, Bangkok, Thailand

Corresponding author: Surapong Auwatanamongkol (e-mail: surapong@as.nida.ac.th).

ABSTRACT A simple and easy to implement but very effective algorithm for solving real-value parameter optimization problems is introduced in this paper. The main idea of the algorithm is to perform a local search repeatedly on a prospective subregion where the optimal solution may be located. The local search randomly samples a number of solutions in a given subregion. If a new best-so- far solution has been found, the center of the search subregion is moved based on the new best-so-far solution and the size of the search subregion is gradually reduced by a predefined shrinking rate. Otherwise, the center of the search is not moved and the size of the search subregion is reduced using a predefined shrinking rate. This process is repeated for a number of instances so that the search is focused on a gradually smaller and smaller prospective subregion. To enhance the likelihood of achieving an optimal solution, many rounds of this repeated local search are performed. Each round starts with a smaller and smaller initial search space. According to the experiment results, the proposed algorithm, though very simple, can outperform some well-known optimization algorithms on some testing functions.

KEYWORDS Real-value Parameters; Function Optimization; Repeated Adaptive Local Search; Shrunk Subregion.

I. INTRODUCTION

REAL-value parameter Function Optimization plays a key role in solving many problems in Machine Learning, Science and Engineering. Some problems may have hundreds of real parameters to be considered for optimization. Many evolutionary optimization techniques including Genetic Algorithm, Evolutionary Strategy, Particle Swarm Optimization, Artificial Bee Colony and Differential Optimization have been proposed for solving real-parameter optimization problems [2, 9, 15, 21-24]. Most algorithms work well for problems with small numbers of parameters, but may fail to reach global optima when the number of parameters becomes very large [10]. Algorithm performance also depends on the complexity of the landscape for problems, e.g. modality, separability, ruggedness and deceptivity [16]. Hence, there is a need for an effective optimization algorithm to resolve high-dimensional, real-parameter problems.

A search strategy for an optimization algorithm needs to balance between exploration and exploitation pressures [17-20]. More emphasis on exploration can lead to a better chance of convergence to a global optimal solution, but such convergence would take more time to achieve. While more emphasis on exploitation can lead to faster convergence, the possibility of premature convergence to local optima may increase. To balance the two conflicting pressures, a good search strategy should encourage exploration during the early stages of the search and gradually encourage exploitation during the later stages of the search. Hence, the possibility of reaching the global optimal solution becomes higher while the convergence is not too slow to achieve. Based on the aforementioned search strategy, a new algorithm for solving real parameter optimization problems is proposed in this paper. The algorithm is based on repeating a local search for global optima around the center of a

potential subregion for the given search space. To encourage exploration during the earlier stages of the search, the initial search subregion covers the entire given search space. The local search is performed repeatedly to enhance the possibility of finding global optima. After finishing a local search, the subregion is gradually shrunk to a smaller subregion that may contain the global optima. Therefore, high exploration and low exploitation are endured during the early stages, while low exploration and high exploitation are endured during the later stages of the search, when the subregion becomes very small. The process is repeated for a number of iterations, after which a new round of repeated local search starts on the subregion with the current center, but with a smaller initial size than that of the previous one. Due to the stochastic process of the solution sampling in the algorithm, it is common to run the algorithm for many trials, and the best solution among the trials is reported.

In experiments, a number of benchmark functions were carried out to evaluate the performance of the proposed algorithm. The experimental results showed that the proposed algorithm was very effective in finding optimal solutions for the functions. The optimal solutions were also compared with those results from well-known optimization algorithms [1].

II. PROPOSED ALGORITHM

To search for the optimal solution to a problem in a large search space, the proposed algorithm first defines an initial potential subregion to be searched as the given entire search space, then pursues multiple rounds of search for the optimal solution in the moved and shrunk subregion. Thus, high exploration and low exploitation are ensured in the earlier rounds. In the later rounds, low exploration and high exploitation become prominent as the subregion becomes smaller and smaller. For each round of the search, a local search is performed repeatedly for a number of iterations. Each local search samples a number of random solutions located within the subregion. If the fittest solution among the sample solutions is fitter than the best-so-far solution, the subregion is shifted so the new solution becomes the center of the subregion as well as the best-so-far solution, otherwise the center remains stationary. The size of the subregion is gradually decreased to reduce the scope of the next iteration of the local search, thus gradually enhancing the exploitation of the search. The shrinking rate of the subregion size is adaptive depending on the progress of the search. The fast predefined shrinking rate, α, is used if there is progress in the search, i.e. the center is moved to a new fitter solution, as seen in Fig. 1. Otherwise, the slower shrinking rate, β, is used instead so that the size of the subregion is reduced at a very low rate, as seen in Fig. 2.

For the sake of simplicity, a subregion size is defined by the width of the subregion for each dimension. The width of the subregion for each iteration of the local search can be calculated as follows:

$$W_{i+1}(j) = \begin{cases} W_i(j)/\alpha \text{ if a new best-so-far solution is} \\ \quad \text{found and becomes the center} \\ \quad \text{of the subregion,} \\ W_i(j)/\beta \text{ otherwise} \end{cases},$$

where

α is a fast predefined shrinking rate and β is a slow predefined shrinking rate, where $\alpha > \beta > 1.0$ e.g. $\alpha = 1.1$, $\beta = 1.01$. $W_i(j)$ is the width of the *j*th dimension of the subregion at the ith iteration of the local search. $W_1(j)$ is the initial width of the *j*th dimension of the subregion.

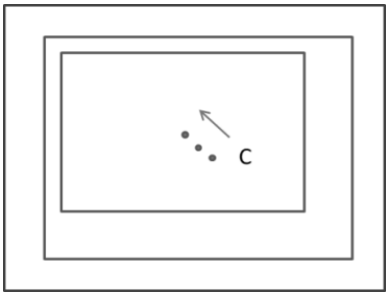

Figure 1. A subregion is shifted with its new center C at a new best-so-far solution and then shrunk with a fast shrinking rate.

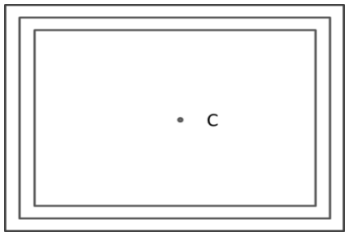

Figure 2. A subregion is stationary as no new best-so-far solution has been found and so it is shrunk with a slow shrinking rate.

It can be seen that the subregion size is reduced more quickly by the reduction factor, which increases exponentially, when a new best-so-far solution is found and the center of the subregion is moved. On the other hand, the subregion size is reduced very slowly by a much smaller shrinking rate when a new best-so-far solution is not found and the center of the subregion is not moved. Since the center of subregion is moved during the iterative local search, it is possible that some parts of the subregion need to be chopped off as they do not fit within the given search region. Hence, the widths of the subregion need to be adjusted accordingly.

Once a round of the local search completed, a new round of the repeated local search starts to increase the likelihood of reaching the global optima. The initial center of the subregion is set to the best-so-far solution and the initial size of the subregion is reduced adaptively in the same way as in the local search process as follows:

$$IS_i = \begin{cases} IS_{i-1} \bullet \alpha & \text{if a new best-so-far solution is} \\ & \text{found in the current round,} \\ IS_{i-1} \bullet \beta & \text{otherwise} \end{cases},$$

$$IW_i(j) = IW(j) / IS_i$$

where IS_i is the reduction factor to be used for the *i*th round of the repeated local search; IS_{i-1} is the reduction factor that has been used for the previous round; α is the fast predefined shrinking rate and β is the slow predefined shrinking rate. These two rates are the same ones used in the local search; $IW_i(j)$ is the initial width of the *j*th dimension of the subregion for the ith round of the local search; $IW(j)$ is the initial width of the *j*th dimension of the given search region; IS_i is the initial reduction factor for the first round of the local search and equals 1.0.

It is possible that some parts of the initial subregion need to be chopped off as they do not fit within the given search region. Hence, the initial widths of the subregion need to be adjusted accordingly. Fig. 3 shows an example of how the width of the subregion is reduced gradually and adaptively during each round of the iterative local search. It can also be seen from the figure that the initial width of the subregion is reduced in a similar fashion after each round of a local search. Therefore, the search is repetitive and continually focuses on smaller and smaller subregion until an optimal solution, possibly the global one, is reached. It is also possible to execute this repeated local search process for many independent runs in order to further enhance the convergence likelihood to the global optima.

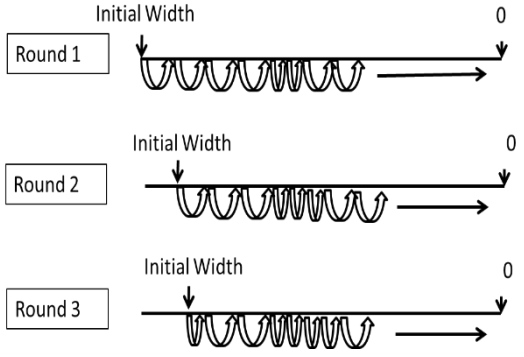

Figure 3. An example of how the width and the initial width of the search subregion are gradually and adaptively reduced.

The search process of the proposed algorithm can be described as a stochastic process represented by a Hidden Markov Model. States of the model correspond to states of the search at the beginning of each iteration. Each state can be described by the upper bound and lower bound values of all parameters, that define the boundary of the search subregion, as well as, a flag stating whether the global optimal solution lies within the search subregion or not. The flag is hidden since the position of the global optimal solution is unknown. The observation of the model is the fitness value of the solution at the center of the subregion.

A state transition of the model occurs between two consecutive iterations of the same round of the local search, or between the state of the last iteration of one round and the first iteration of the next round. The new state of a transition is dependent on how the new center and new boundary of the search subregion are determined. The goal of the search is to reach the final state which subregion is very small but still contains the global optimal solution.

The local search picks the best-so-far solution to be the new center of the search subregion. It is expected that the new center is getting closer to the global optimal solution, therefore, it would move the search subregion toward the optimal solution. The subregion is also shrunk gradually using small adaptive shrinking rates so the subregion gets smaller and smaller while there is a good chance that the subregion still contains the global optimal solution. After each round of the repeated local search, it is possible that the search subregion may miss the global optimal solution. Therefore, a new round of the repeated local search is performed with an initial size smaller than that of the previous round to narrow down the scope of the search. With many rounds of this repeated local search, there is a high possibility that the desired final state would be reached.

III. EXPERIMENTS AND RESULTS

Six real parameter functions were used to evaluate the performance of the proposed method. They comprised both unimodal and multimodal functions as follows:

f1: Sphere function (unimodal)
f2: Schwefel 2.22 problem (unimodal)
f3: Rosenbrock function (multimodal)
f4: Rastrigin function (multimodal)
f5: Griewank function (multimodal)
f6: Ackley function (multimodal)

A set of experiments were carried out to evaluate the performance of the proposed algorithm with the following parameter settings:

Fixed Parameters

Number of independent runs = 25
Number of rounds of execution for the repeated local search (P) = 1000

Varying Parameters

To study the effects of each varying parameter on the performance of the proposed algorithm, the other parameters are fixed at a baseline parameter setting as follows:

Number of Dimensions (D) = 100
Fast Shrinking Rate (α) = 1.1
Slow Shrinking Rate (β) = 1.01
Number of solutions sampled for each local search (N) = 100
Number of repetitions for the local search (M) = 100

Table 1. Varying the number of dimensions (D)

Function	D = 25			D = 50			D = 100			D = 500		
	Best	**Mean**	**SD**	**Best**	**Mean**	**SD**	**Best**	**Mean**	**SD**	**Best**	**Mean**	**SD**
f1	2.86E-17	2.95E-15	4.70E-15	1.80E-15	2.71E-12	4.13E-12	**0**	**0**	0	**0**	**0**	**0**
f2	1.62E-06	8.48E-06	5.81E-06	2.23E-10	1.15E-08	1.27E-08	**8.53E-14**	**1.15E-13**	9.78E-15	167.87	177.36	4.81
f3	**7.76E-11**	**31.75**	10058.30	31.75	2985.77	10058.31	90.79	5495.25	10861.84	676.85	242940.22	890655.99
f4	**8.95**	**19.88**	5.33	60.84	88.93	16.03	308.20	378.47	48.70	5981.01	6764.93	355.72
f5	7.77E-16	**6.14E-14**	1.20E-13	**0**	4.14E-03	5.69E-03	**0**	1.97E-03	3.56E-03	6.37E-05	1.73E-03	4.35E-03
f6	7.96E-09	4.58E-08	3.18E-08	4.31E-08	1.13E-06	1.21E-06	**4.93E-14**	**6.14E-14**	5.12E-15	2.20E-13	2.33E-13	8.35E-15

From the experimental results shown in Table 1, it can be seen that the proposed algorithm can perform very well, even when the number of dimensions is increased for some testing functions. Therefore, the optima convergence for the proposed algorithm would probably depend very much on the complexity of the landscape of the search region.

Table 2. Varying the Fast Shrinking Rate (α)

Functio	α = 1.05			α = 1.1			α = 1.2			α =1.5		
n	**Best**	**Mean**	**SD**	**Best**	**Mean**	**SD**	**Best**	**Mean**	**SD**	**Best**	**Mean**	**SD**
f1	2.21E-10	4.16E-09	8.21E-09	**0**	**0**	0	**0**	**0**	0	96464.36	151061.56	26632.55
f2	1.89E-06	5.44E-06	2.40E-06	**8.53E-14**	**1.15E-13**	9.78E-15	53.59	71.62	11.69	152.45	174.65	10.15
f3	81.69	7478.75	16852.97	**90.79**	**5495.25**	10861.84	203.89	590028.14	1063736.62	54266447963.21	133147772769.20	46918668735.16
f4	**289.31**	**362.60**	**37.27**	308.20	378.47	48.70	377.79	498.93	62.69	90692.28	154720.55	33702.27
f5	8.88E-12	1.97E-03	4.67E-03	**0**	**1.97E-03**	3.56E-03	3.33E-16	4.73E-03	7.96E-03	23.97	39.61	10.12
f6	6.52E-06	2.05E-05	8.64E-06	**4.93E-14**	**6.14E-14**	5.12E-15	5.28E-14	6.46E-14	4.35E-15	19.97	19.99	0.01

From the experimental results shown in Table 2, it can be seen that the repeated local search can miss the optimal solution if the fast shrinking rate gets too large and the search subregion is shrunk too quickly. On the other hand, if the search subregion is shrunk too slowly, it may take more time and probably require a higher number of solutions to sample in order to converge to an optimal solution. From the results, the optimal fast shrinking rate for most of the testing functions is around 1.1.

Table 3. Varying the Slow Shrinking Rate (β)

Function	β =1.005			β =1.01			β=1.05			β = 1.10		
	Best	**Mean**	**SD**	**Best**	**Mean**	**SD**	**Best**	**Mean**	**SD**	**Best**	**Mean**	**SD**
f1	1.71E-11	2.68E-09	6.39E-09	0	0	0	0	0	0	0	0	0
f2	1.78E-12	8.77E-10	3.36E-09	**8.53E-14**	**1.15E-13**	9.78E-15	1.05E-04	1.1	2.27	37.62	74.46	20.67
f3	92.89	9925.52	15781.58	**90.79**	**5495.25**	10861.84	92.09	11518.28	16069.47	87.79	7436.93	13447.97
f4	**259.49**	**352.18**	**52.10**	308.20	378.47	48.70	405.63	489.66	49.98	505.05	626.10	72.68
f5	0	2.46E-03	4.00E-03	**0**	**1.97E-03**	3.56E-03	0	1.97E-03	3.55E-03	1.11E-16	3.35E-03	5.21E-03
f6	1.34E-06	4.48E-05	4.85E-05	**4.93E-14**	**6.14E-14**	5.12E-15	6.35E-14	6.95E-14	4.68E-15	20.96	20.96	7.11E-15

From the experimental results shown in Table 3, it can be seen that the repeated local search can miss the optimal solution similarly to the case of the fast shrinking rate when the slow shrinking rate gets too large and the search subregion is shrunk too quickly. On the other hand, if the search subregion is shrunk too slowly, it can take more time and probably require a higher number of solutions to sample in order to converge to an optimal solution. From the results, the optimal slow shrinking rate for most of the testing functions is around 1.01. It should be noted that when the slow shrinking rate is set to be the same as the fast shrinking rate, i.e. there is no slow shrinking rate in this case as $\alpha = \beta = 1.10$, the algorithm converges to a poor solution for all testing functions except f1 function. This means the adaptive scheme, which employs both fast and slow shrinking rates for adjusting the shrinking rate, can help the local search achieve much better solutions.

Table 4. Varying the number of randomly sampled solutions for each local search (N)

Function	N = 50			N = 100			N = 200		
	Best	**Mean**	**SD**	**Best**	**Mean**	**SD**	**Best**	**Mean**	**SD**
f1	0	0	0	0	0	0	**0**	**0**	0
f2	1.14E-13	6.19E-14	2.15E-01	8.53E-14	1.15E-13	9.78E-15	**8.53E-14**	**9.78E-14**	1.41E-14
f3	92.34	11946.41	16500.34	90.79	5495.25	10861.84	**90.72**	**4650.82**	10543.46
f4	**265.45**	427.07	86.76	308.20	378.47	48.70	309.19	**357.43**	37.11
f5	**0**	3.94E-03	4.83E-03	**0**	**1.97E-03**	3.56E-03	**0**	4.04E-03	5.90E-03
f6	5.64E-14	6.85E-14	4.82E-15	4.93E-14	6.14E-14	5.12E-15	**4.57E-14**	**5.74E-14**	4.98E-15

From the experimental results shown in Table 4, it can be seen that a better solution can be achieved by the proposed algorithm if more solutions are sampled for each local search, resulting in a higher chance of finding the optimal solution.

Table 5. Varying the number of repetitions for the local search (M)

Function	M = 25			M = 50			M = 100		
	Best	**Mean**	**SD**	**Best**	**Mean**	**SD**	**Best**	**Mean**	**SD**
f1	**0**	1.92E-18	9.17E-18	1E-20	7.02E-17	1.68E-16	**0**	**0**	0
f2	4.78E-01	4.02	2.96	1.14E-13	1.19E-13	1.14E-14	**8.53E-14**	**1.15E-13**	9.78E-15
f3	92.85	16042.80	46750.74	**87.81**	5586.96	12623.40	90.79	**5495.25**	10861.84
f4	367.85	459.87	62.64	322.12	434.54	56.17	**308.20**	**378.47**	48.70
f5	**0**	2.66E-03	5.02E-03	**0**	2.86E-03	5.29E-03	**0**	**1.97E-03**	3.56E-03
f6	2.35E-12	7.48E-11	8.54E-11	2.69E-11	2.44E-09	2.68E-09	**4.93E-14**	**6.14E-14**	5.12E-15

Table 6. Effects of the increases of the parameter values on the possibility of optima convergence for the proposed algorithm

Parameter	Optima Convergence Possibility
Number of Dimensions (D) ↑	Depending on the complexity of the landscape of the search region
Fast Shrinking Rate (α) ↑	↓
Slow Shrinking Rate (β) ↑	↓
Number of solutions sampled for each local search (N) ↑	↑
Number of repetitions for the local search (M) ↑	↑

Comparisons of the optimal results achieved from the proposed algorithm against those achieved by some other well-known optimization algorithms are shown in Table 7. The results of the well-known algorithms, namely MABC[26], GOABC[27], CoDE[28,29], FA[30], BA[31], BSA[32,33,34], BDS[32,33,34], SDS[32,33,34] and PSCS[1] are based on [1]. The results are the mean optimal values for 30 runs. The number of dimensions is 50 and the total number of function evaluations for each run is at least 2,000,000. The standard deviations of the results are shown in the parentheses. The proposed algorithm was executed for 30 runs. The number of sampled solutions, N, was 200 and the number of repetitions, M, was 10. The total number of function evaluations for each run was at least 2,000,000.

Table 7. Performance comparison between the proposed algorithm and others

Algorithms	f1	f2	f3	f4	f5	f6
MABC	0 (0)	8.294e-012 (7.675e-012)	36.2419 (31.0792)	27.4042 (56.7605)	1.1191e-014 (2.0167e-015)	0(0)
GOABC	4.590e-008 (1.026e-007)	0.3459 (0.1311)	4.98838e+002 (8.0716e+002)	1.1952 (2.1599)	5.2013e-004 (9.2477e-004)	0.0024 (0.0055)
CoDE	0 (0)	9.093e-048 (2.567e-047)	0.3987 (1.2271)	0.4975 (0.9411)	4.4409e-015(0)	0 (0)
FA	7.465e-101(7.142e-102)	0.0532 (0.0251)	45.8660 (0.8307)	93.9239 (41.6611)	5.3468e-014 (1.1621e-014)	2.220e-017 (4.965e-017)
BA	2.7120e-005 (3.023e-006)	32.0319 (5.5715)	9.5638 (2.4789)	1.0328e+002 (22.6817)	16.7048 (0.7936)	18.7555 (41.9249)
BSA	2.201e-261 (0)	0.0309 (0.0266)	0.9966 (1.7711)	0.3482 (0.6674)	2.7355e-014 (4.5343e-015)	0.0013 (0.0033)
BDS	0 (0)	2.293e -013 (3.594e-013)	9.8809 (20.8574)	0.0497 (0.2224)	1.0302e-014 (3.3157e-015)	0 (0)
SDS	0(0)	1.319e-016 (1.755e-016)	5.264e-027 (1.8936 e-026)	0.8457 (1.2616)	1.3500e-014 (2.9330e-015)	8.6131e-004 (0.0038)
PSCS	0(0)	5.830e-020 (1.301e-019)	2.5590e-028 (2.0639e-028)	0 (0)	4.4409e-015 (0)	0 (0)
Proposed Algorithm						
Mean	0	5.7790e-014	7.6997e+03	1.2533e+02	5.8290e-03	4.574e-014
Standard deviation	0	5.1018e-015	1.5834e+04	24.9333	6.5551e-03	7.0459e-015
Best	0	5.6843e-014	37.8987	80.5302	0	3.5083e-014
α, β	1.1,1.01	1.1,1.01	1.05,1.01	1.05,1.005	1.1,1.01	1.1,1.01

From the experimental results shown in Table 7, the PSCS performed the best while the proposed algorithm outperformed some other algorithms e.g GOABC, BA and BSA for some test functions except f3 and f4. For these two functions, the proposed algorithm achieved variable results for different runs with high means and high standard deviations, though, the best solutions achieved by the proposed algorithm are much better than the means. So, for

some complex functions like these two functions, several runs are needed for the proposed algorithm to reach good optimal solutions.

IV. DISCUSSION

It should be noted that the proposed algorithm utilizes a sampling method for a local search. It does not need to maintain a population of solutions during the search. Hence, it requires very small amount of memory to run. The exploration and exploitation aspects of the search can be easily controlled by the number of sampled solutions and the shrinking rates. The high exploration of the search is achieved with a large number of the sampled solutions and/or very slow shrinking rates. While, the exploitation of the search is assured when fast shrinking rates are employed. However, according to the experiments, for a multimodal and very complex landscape functions, e.g. f3 and f4, the algorithm may need a number of runs with a large number of sampled solutions as well as very slow shrinking rates to reach good optimal solution. This can cause longer time to run the algorithm. To alleviate this problem, a parallel sampling technique may be used to do the local search so the computation can be speeded up.

V. CONCLUSION

In this paper, a simple but very effective algorithm for solving real-value parameter function optimization is proposed. The algorithm enhances the possibility of convergence to an optimal solution by repeating the local search many times on a gradually shrinking search subregion. The center of the subregion is also moved to the most recently found solution. The gradually shrinking size of the search subregion allows for the exploration of the search during the early iterations when the search subregion is still relatively large and gradually allows the exploitation of the search during later iterations when the search space becomes very small. The shrinking rate of the search subregion is also designed to be adaptive. A high shrinking rate is employed when a new best-so-far solution has been found. Otherwise, a small shrinking rate is used instead. This helps the local search to remain focused on the prospective subregion while at the same time not shrinking the size of the subregion too much, causing the search to miss the optimal solution. This iterative local search process is also repeated for many rounds. In each round, the best-so-far of the previous round is used as the initial center and the initial size of the search subregion is also reduced by an adaptive rate using the same strategy mentioned previously. According to the experiments, the algorithm outperforms some optimization algorithms and performs comparable with some others for some testing functions. However, the proposed algorithm is much simpler and easier to implement with very minimal memory requirements.

References

[1] X. Li, M. Yin, "A particle swarm inspired cuckoo search algorithm for real parameter optimization," *Soft Computing*, vol. 20, pp. 1389-1413, 2016.

[2] E. K. Nyarko, R. Cupec, D. Filko, "A comparison of several heuristic algorithms for solving high dimensional optimization problems," *International Journal of Electrical and Computer Engineering Systems*, vol. 5, no. 1, pp. 1-8, 2014.

[3] X. Xia, J. Liu, Z. Hu, "An improved particle swarm optimizer based on tabu detecting and local learning strategy in a shrunk search space," *Applied Soft Computing*, vol. 23, pp. 76-90, 2014.

[4] S. Das, S. S. Mullick, P. N. Suganthan, "Recent advances in differential evolution – An updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1-30, 2016.

[5] J. F. Qiu, J. W. Wang, D. Y., J. Xie, and N. Z. Yao, "A parameter adaptive artificial bee colony algorithm for real-parameter optimization," *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 9, pp. 34-39, 2013.

[6] F. Luo, J. Zhao, Z. Y. Dong, "A new metaheuristic algorithm for real parameter optimization: Natural aggregation algorithm," *Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 94-103.

[7] M. Duan, H. Yang, S. Wang, Y. Liu, "Self-adaptive dual-strategy differential evolution algorithm," *PLoS ONE*, vol. 14, e0222706, 2019.

[8] T. Eltaeib, A. Mahmood, "Differential evolution: A survey and analysis," *Applied Sciences*, vol. 8, issue 10, 1945, 2018.

[9] K. R. Opara, J. Arabas, "Differential evolution: A survey of theoretical analyses," *Swarm and Evolutionary Computation*, vol. 44, pp. 546-558, 2018.

[10] P. Agarwal, S. Mehta, "Empirical analysis of five nature-inspired algorithms on real parameter optimization problems," *Artificial Intelligence Review*, vol. 50, pp. 383-439, 2018.

[11] M. Mavrovouniotis, F. M. Muller, S. Yang, "Ant colony optimization with local search for dynamic traveling salesman problems," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1743-1756, 2017.

[12] D. Zhan, J. Qian, Y. Cheng, "Balancing global and local search in parallel efficient global optimization algorithms," *Journal of Global Optimization*, vol. 67, pp. 873-892, 2017.

[13] W. Deng, H. Zhao, L. Zou, G. Li, X. Yang, D. Wu, "A novel collaborative optimization algorithm in solving problems," *Soft Computing*, vol. 21, pp. 4387-4398, 2017.

[14] A. Nakib, S. Ouchraa, N. Shvai, L. Souquet, and E.-G. Talbi, "Deterministic metaheuristic based on fractal decomposition for large-scale optimization," *Applied Soft Computing*, vol. 61, pp. 468–485, 2017.

[15] S. Tuo, J. Zhang, X. Yuan, L. Yong, "A new differential evolution algorithm for solving multimodal optimization problems with high dimensionality," *Soft Computing – A Fusion of Foundations, Methodologies and Applications*, vol. 22, issue 13, pp. 4361-4388, 2018.

[16] P. Caamano, F. Bellas, J. A. Bacerra, and R. J. Duro, "Evolutionary algorithm characterization in real parameter optimization problems," *Applied Soft Computing*, vol. 13, pp. 1902-1921, 2013.

[17] H. Zhang, J. Sun, T. Liu, K. Zhang, Q. Zhang, "Balancing exploration and exploitation in multi-objective evolutionary optimization," *Information Sciences*, vol. 497, pp. 129-148, 2019.

[18] G. Kaur, S. Arora, "Chaotic whale optimization algorithm," *Journal of Computational Design and Engineering*, vol. 5, issue 3, pp. 275-284, 2018.

[19] M. Crepinsek, S.-H. Liu, M. Mernik, "Exploration and exploitation in evolutionary algorithm: A survey," *ACM Computing Surveys*, vol. 45, issue 3, pp. 1-33, 2013.

[20] X.-S. Yang, "Nature-inspired optimization algorithms: Challenges and open problems," *Journal of Computational Science*, vol. 46, article 101104, 2020.

[21] J. Brest, M. S. Maucec, B. Boskovic, "Single objective real-parameter optimization: Algorithm JSO," *Proceedings of the 2017 IEEE Congress on Evolutionary Computation*, 2017, pp. 1311-1318.

[22] H. Peng, C. Deng, Z. Wu, "Best neighbor-guided artificial bee colony algorithm for continuous optimization problems," *Soft Computing*, vol. 23, pp. 8723–8740, 2019.

[23] J. Ding, J. Liu, K. R. Chowdhury, W. Zhang, Q. Hu, J. Lei, "A particle swarm optimization using local stochastic search and enhancing diversity for continuous optimization," *Neurocomputing*, vol. 137, pp. 261-267, 2014.

[24] I. Ciornei, E. Kyriakides, "Hybrid ant colony-genetic algorithm (GAAPI) for global continuous optimization," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 42, n. 1, pp. 234-245, 2012.
[25] M. Kaedi, "Fractal-based algorithm: a new metaheuristic method for continuous optimization," *International Journal of Artificial Intelligence*, vol. 15, issue 1, pp.76-92, 2017.
[26] B. Akay, D. Karaboga, "A modified artificial bee colony algorithm for real-parameter optimization," *Information Sciences*, vol. 192, pp.120-142, 2012.
[27] M. El-Abd, "Generalized opposition-based artificial bee colony algorithm," *Proceedings of the IEEE Congress on Evolution Computation (CEC)*, 2012, pp. 1-4.
[28] Y. Wang, Z. Cai, Q. Zhang, "Enhancing the search ability of differential evolution through orthogonal crossover," *Information Sciences*, vol. 18, issue 1, pp.153–177, 2012.
[29] Y. Wang, Z. Cai, Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolution Computations*, vol. 15, issue 1, pp. 55-66, 2011.
[30] X. S. Yang, "Firefly algorithms for multimodal optimization," *Stochastic algorithms: foundations and applications, SAGA 2009, Lecture Notes in Computer Sciences*, vol. 5792, pp. 169-178.
[31] X. S. Yang, H. Gandomi Amir, "Bat algorithm: A novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, 2012, pp. 464-483.
[32] P. Civicioglu, "Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm," *Computers and Geosciences*, vol. 46, pp. 229-247, 2012.
[33] P. Civicioglu, "Backtracking search optimization algorithm for numerical optimization problems," *Applied Mathematics and Computation*, vol. 219, issue 15, pp. 8121-8144, 2013.
[34] P. Civicioglu, "Circular antenna array design by using evolutionary search algorithms," *Progress Electromagnetics Research B*, vol. 54, pp. 265-284, 2013.

SURAPONG AUWATANAMONGKOL received the bachelor degree of electrical engineering from Chulalongkorn University, Thailand, the master degree of computer science from Georgia Institute of Technology, USA, and the doctoral degree of computer science from Southern Methodist University, USA. He is an associate professor at the school of Applied Statistics, National Institute of Development Administration, Bangkok, Thailand. His current research interests include Evolutionary Computation, Machine Learning, Data and Image Analytics.