New Trends in Software Methodologies, Tools and Techniques
H. Fujita et al. (Eds.)
IOS Press, 2014

doi:10.3233/978-1-61499-434-3-160

Synchronous Gravitational Search Algorithm vs Asynchronous Gravitational Search Algorithm: A Statistical Analysis

Nor Azlina AB. AZIZ[a,b,1], Zuwairie IBRAHIM[c], Sophan Wahyudi NAWAWI[d], Shahdan SUDIN[d], Marizan MUBIN[a,1] , and Kamarulzaman AB. AZIZ[b]
[a]*Faculty of Engineering, University of Malaya, Malaysia*
[b]*Multimedia University, Malaysia*
[c]*Faculty of Electrical & Electronic Engineering,Universiti Malaysia Pahang, Malaysia*
[d]*Faculty Electrical Engineering, Universiti Teknologi Malaysia, Malaysia*

Abstract. Gravitational search algorithm (GSA) is a new member of swarm intelligence algorithms. It stems from Newtonian law of gravity and motion. The performance of synchronous GSA (S-GSA) and asynchronous GSA (A-GSA) is studied here using statistical analysis. The agents in S-GSA are updated synchronously, where the whole population is updated after each member's performance is evaluated. On the other hand, an agent in A-GSA is updated immediately after its performance evaluation. Hence an agent in A-GSA is updated without the need to synchronize with the entire population. Asynchronous update is more attractive from the perspective of parallelization. The results show that both implementations have similar performance.

Keywords. Gravitational search algorithm, asynchronous, synchronous, statistical analysis, optimization

Introduction

Gravitational search algorithm (GSA) is a population based metaheuristic algorithm which is a physic inspired. It was introduced in 2009 by E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi [1]. Agents in GSA look for optimal solution by emulating the interaction between masses in the universe. The Newton's law of universal gravitational state that, two masses in universe attracts each other and the attraction force is directly proportional to the product of the two masses and inversely proportional to the square of the distance between them.

The masses in universe are represented in GSA by agents. The agents are distributed within the search area. The agents' masses are changed based on the fitness of the solution according to their position in the search space. An agent with good solution has a bigger mass therefore the attraction force exerted by this agent is bigger.

[1] Corresponding authors: Faculty of Engineering, University of Malaya, 50603 Kuala Lumpur, Malaysia; Email: azlina.aziz@mmu.edu.my, marizan@um.edu.my

GSA is gradually gaining attention from research community [2]. It has been found superior to some well-established optimization algorithms, such as Central Force Optimization (CFO), Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) [1]. In [3], GSA was compared with GA in solving the cell placement problem of VLSI circuits design process, the results show that GSA has a better performance than GA. GSA and a modified PSO algorithm were applied for synthesis of scanned thinned array in [4], which GSA was found to outperform the modified PSO.

Many variations of GSA have been introduced either to improve on results, speed of convergence, computational requirements or to cater to specific problem. A fast discrete GSA (FDGSA) is proposed in [5] to solve discrete optimization problem. A new variation of GSA to solve multiobjective optimization problem, vector evaluated GSA (VEGSA) is presented in [6]. In [7] GSA is merged with PSO algorithm to benefit from the exploitation ability of PSO and exploration ability of GSA.

Here, the performance of the original GSA which is synchronously updated (S-GSA) is statistically compared with asynchronous GSA (A-GSA) [8]. In S-GSA the agents update their mass using information of the best and worst performer after the whole population update their performance. S-GSA is the original GSA. A-GSA was introduced in [8]. In A-GSA, an agent updates its mass as soon as its own performance is evaluated. Therefore, the agents in A-GSA use mixture of updated and outdated agents' positions to update their mass. Since the agents in A-GSA can be updated independent of one another, this version of GSA algorithm has a better potential for parallelization.

Asynchronous update metaheuristic algorithms have been reported to possess higher exploration ability [9]. This is due to lack of synchronization on the information used to guide the search, which in the case of GSA are best and worst performers of the population. Unlike other metaheuristic algorithms such as particle swarm optimization (PSO), GSA's agents do not have memory, hence, their search is strongly driven by current situation of the population. Lack of memory causes more exploration by the agents. Therefore it is interesting to observe how asynchronous update which is known to cause more exploration will affect the performance of GSA. In [8], the performance of S-GSA and A-GSA was compared using set of benchmark functions consisting of unimodal and multimodal functions. However, no statistical analysis was conducted, thus this could lead to bias and in accurate findings. Therefore, statistical analysis on the performance of S-GSA and A-GSA is conducted here using a set of more challenging test functions which are extracted from 2014 IEEE Congress on Evolutionary Computation (CEC2014) Special Session and Competition on single objective real-parameter numerical optimization [10].

This paper is organized as follows. The original GSA algorithm, S-GSA, is presented in section 1 followed by the A-GSA algorithm in section 2. The performance of A-GSA is compared with S-GSA using a set of benchmark functions in section 3 and the results are statistically analysed. Finally, the work is concluded in section 4.

1. Synchronous GSA

As mention in the previous section, GSA is inspired by how masses in universe attract each other. The attraction force is influenced by the mass of the body and also the distance between them. A body with bigger mass exerts a higher attraction force

while attraction force between masses is reduced when the distance between them is bigger.

The universe is the search area in GSA while the masses are the agents. Each of the agents has mass which is based on the fitness of the solution suggested by the position of the agent. An agent's position, X_i, in the search space represents the solution suggested. The position of agent i^{th} is,

$$X_i = \left(x_i^1, x_i^2, \dots, x_i^d\right) \quad i = 1,2, \dots, N \tag{1}$$

Therefore the position of an agent at dimension d^{th} is denoted as, x_i^d. The initial values of the agents' positions are randomly initialized according to the search area.

The fitness of each agent's position is evaluated using problem dependent fitness function. Given that an agent's fitness at iteration t is represented as, $fit_i(t)$, the mass of the agents is calculated as follow;

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{2}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{3}$$

The $best(t)$ and $worst(t)$ represent the best and worst fitness among the agents in the population. These values are selected depending on whether the problem to be optimized is a maximization or minimization problem. Assuming a minimization problem, the definitions of these values are as follow;

$$best(t) = min\{fit_1(t), fit_2(t), \dots, fit_N(t)\} \tag{4}$$

$$worst(t) = max\{fit_1(t), fit_2(t), \dots, fit_N(t)\} \tag{5}$$

The masses are then used to calculate the force of an agent towards other agents, $F_{ij}^d(t)$.

$$F_{ij}^d(t) = G(t)\frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon}\left(x_j^d(t) - x_i^d(t)\right) \tag{6}$$

$R_{ij}(t)$ is the Euclidian distance between agent ith and jth. A small constant ε is added to avoid division by zero when the distance between the agents is zero.

$M_{pi}(t)$ and $M_{aj}(t)$ are passive and active gravitational mass of agent ith and jth respectively. *G(t)* is the gravitational constant at time t. The equations for $M_{pi}(t)$, $M_{aj}(t)$ and *G(t)* are;

$$M_{pi}(t) = M_{ai}(t) = M_i(t) \tag{7}$$

$$G(t) = G_0 \times e^{-\beta\frac{t}{T}} \tag{8}$$

G_o is the gravitational constant at the start of the universe. This is typically set to 100. β is another constant and normally set to 20. T is the total number of iteration.

The total force acting on agent i[th] is;

$$F_i^d(t) = \sum_{j=1, j\neq i}^{N} rand_j F_{ij}^d(t) \quad (9)$$

where, $rand_j$ is a random number in the interval [0,1].

The agents in GSA are also subjected to Newton's law of motion, where the acceleration of a body is directly proportional and in the same direction as the net force acting on itself and inversely proportional to its mass. According to this law of motion, the acceleration of agent i[th] over dimension d[th], $\alpha_i^d(t)$ can be calculated using the following equation;

$$\alpha_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \quad (10)$$

The agents' velocities and positions are then updated using the equations below;

$$v_i^d(t) = rand_i^d \times v_i^d(t-1) + \alpha_i^d(t) \quad (11)$$

$$x_i^d(t) = x_i^d(t-1) + v_i^d(t) \quad (12)$$

The flow of the original GSA algorithm is shown in Figure 1(a). The fitness of the whole population is evaluated first before best and worst values are identified. The agents in the population are then synchronously moved to new position.

2. Asynchronous GSA

Asynchronous update method is a more accurate natural evolution model and increases the potential of parallelization of the algorithms [11, 12]. In other optimization algorithms such as PSO and GA, asynchronous update method had been proposed and successfully implemented [11, 13]. In PSO, asynchronous update increases its exploration ability. PSO is an algorithm which is strong in exploitation, therefore, prone to premature convergence.

Figure 1(b) shows the flow of the A-GSA algorithm. In A-GSA an agent's position and velocity are updated as soon as the agent's performance is evaluated without waiting for the entire population to be evaluated. Therefore, when an agent position is updated the *best* and *worst* agents are identified using mix of updated positions and old positions and the respective fitness of these positions. This mixture of information encourages more exploration by agents [9].

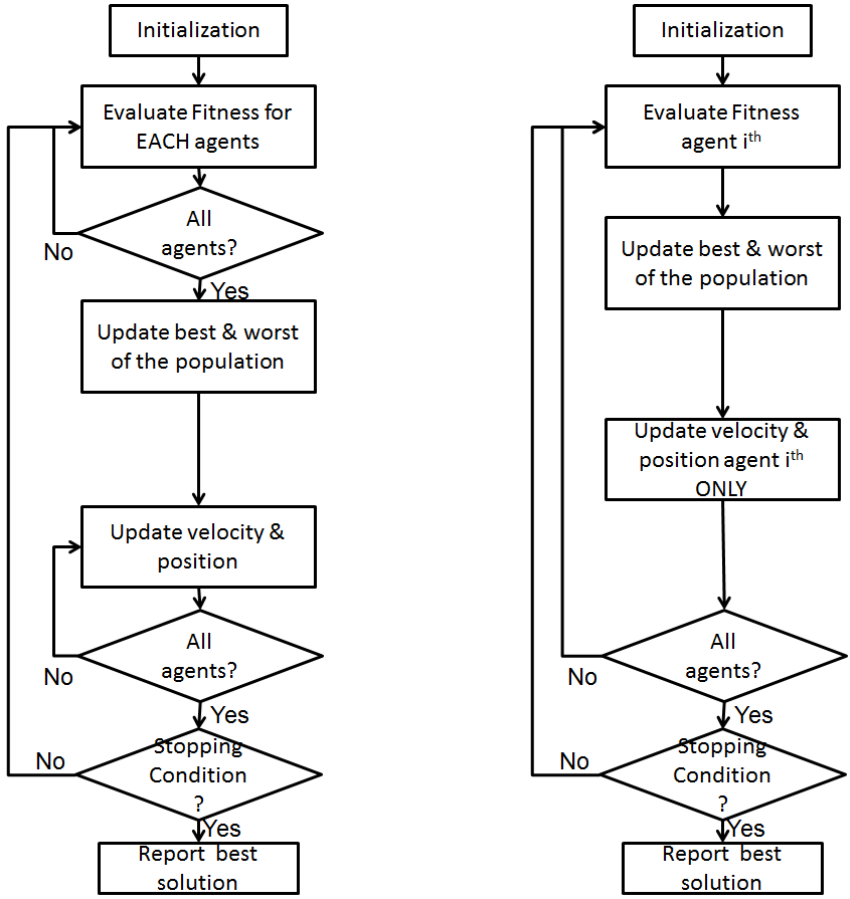

Figure 1. Flowchart for (a) S-GSA (b) A-GSA

3. Results and Discussion

The S-GSA and A-GSA algorithms are implemented here without parallelization and their performances are compared. The algorithms are tested using a set of 8 benchmark functions consisting of three unimodal and five multimodal functions. These benchmark functions are extracted from the CEC2014 test suite for single objective real-parameter numerical optimization competition [10]. Table 1 lists the benchmark functions used, while Table 2 shows the parameters for the benchmark functions, the dimension is set to 30, thus making these as high dimensional problem, while the other parameters follow CEC2014's guidelines.

The parameters for GSA are listed in 0. Each function was subjected to 50 runs and the position giving the lowest fitness per run was recorded. In a run, the maximum iteration, T, was set to 1000. The number of agents used, N, is 10. The values for G_o, β and ε follow the recommended values in [1].

Table 1. Benchmark Functions

Modality	Function	Function Definition	Basic Function	F_i^*
Unimodal	Rotated High Conditioned Elliptic	$F_1(x) = f_1(\mathbf{M}(x - o_1)) + F_1^*$	$f_1(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2$	100
	Rotated Bent Cigar	$F_2(x) = f_2(\mathbf{M}(x - o_2)) + F_2^*$	$f_2(x) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2$	200
	Rotated Discus	$F_3(x) = f_3(\mathbf{M}(x - o_3)) + F_3^*$	$f_3(x) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2$	300
Multimodal	Shifted and Rotated Rosenbrock	$F_4(x) = f_4\left(\mathbf{M}\left(\frac{2.048(x - o_4)}{100}\right) + 1\right) + F_4^*$	$f_4(x) = \sum_{i=1}^{D-1} [100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2]$	400
	Shifted and Rotated Ackley	$F_5(x) = f_5(\mathbf{M}(x - o_5)) + F_5^*$	$f_5(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + e$	500
	Shifted and Rotated Weierstrass	$F_6(x) = f_6\left(\mathbf{M}\left(\frac{0.5(x - o_6)}{100}\right)\right) + F_6^*$	$f_6(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{kmax}\left[a^k \cos(2\pi b^k (x_i + 0.5))\right) - D\sum_{k=0}^{kmax}[a^k \cos(2\pi b^k \cdot 0.5)]\right]$ $a = 0.5, b = 3, kmax = 20$	600
	Shifted and Rotated Griewank	$F_7(x) = f_7\left(\mathbf{M}\left(\frac{600(x - o_7)}{199}\right)\right) + F_7^*$	$f_7(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	700
	Shifted Rastrigin	$F_8(x) = f_8\left(\mathbf{M}\left(\frac{5.12(x - o_6)}{100}\right)\right) + F_8^*$	$f_8(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$	800

Table 2. Parameters of Benchmark Functions

Parameter	Value
Dimension, D	30
Shifted global optimum, $o_{il} = [o_{i1}, o_{i,2}, \ldots, o_{iD}]^T$	randomly distributed in $[-80, 80]^D$
Search range	$[-100, 100]^D$
Rotation matrix, $\mathbf{M_i}$	function dependent

Table 3. Parameters of GSA

Parameter	Value
Number of agents, N	10
G_0	100
β	20
T	1000
ε	$2.2204e^{-16}$

Table 4. Average Results

	S-GSA	A-GSA
f_1	1.1779E+09	1.2733E+09
f_2	8.2312E+10	8.0093E+10
f_3	1.7107E+05	1.6205E+05
f_4	1.5930E+04	1.5262E+04
f_5	5.2111E+02	5.2109E+02
f_6	6.4250E+02	6.4234E+02
f_7	1.4496E+03	1.4562E+03
f_8	1.2043E+03	1.1979E+03

The results are presented here using boxplot (Figure 2 and Figure 3). The symbol * in the figure represents the mean while ○ represents outliers. The Wilcoxon signed rank test with significant level, α, equal to 0.05 is used to look for any significant difference between S-GSA and A-GSA. Table 4 shows the mean results for each test functions, which are then used for the Wilcoxon test.

The Wilcoxon test indicates that there is no significant difference between S-GSA and A-GSA.This is confirmed by the boxplots in Figure 2and Figure 3. The size and the location of the box plot for both S-GSA and A-GSA are similar to each other. This findings contradict what was reported in [8], which confirms the important of proper statistical analysis methods in ensuring accurate and unbiased evaluation of metaheuristic algorithms.

The findings observed here confirmed that GSA has strong exploration ability. This is due to the fact that the agents of GSA are driven by present *best* and *worst* values, instead of using memory. Therefore, implementing asynchronous update in GSA does not contribute to much significant difference. This also shows that GSA is a good candidate for parallelization through asynchronous update implementation, which is the preferred method in parallelization, as A-GSA performs as good as S-GSA.

The results also show that GSA is only able to find near optimal solution for shifted and rotated Ackley and shifted and rotated Weierstrass. This is due to the nature of the other test functions where their global optimum are not protruding as these two functions, hence stronger exploitation is needed to locate them, which is an aspect lacking in GSA.

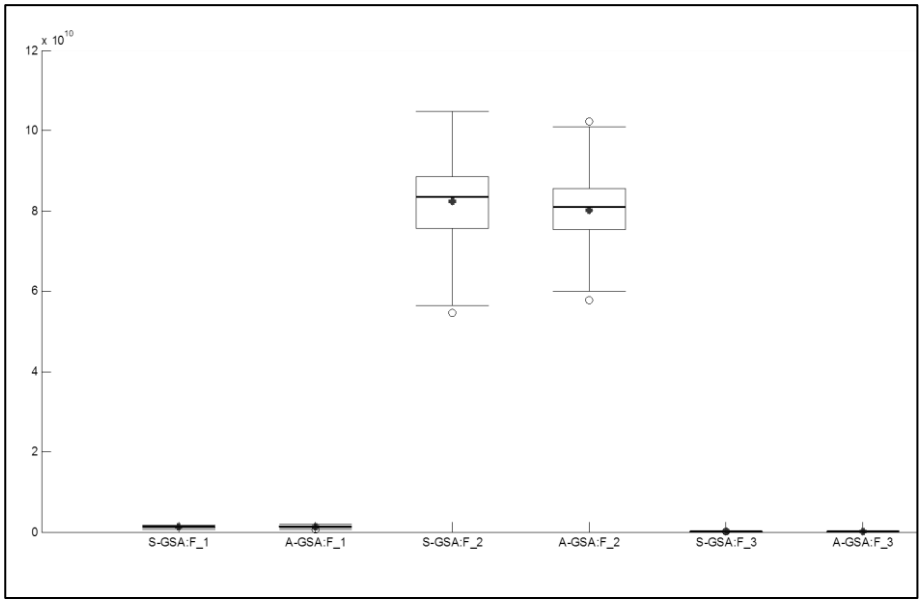

Figure 2. Boxplot for Test on Unimodal Functions (F_1, F_2 and F_3)

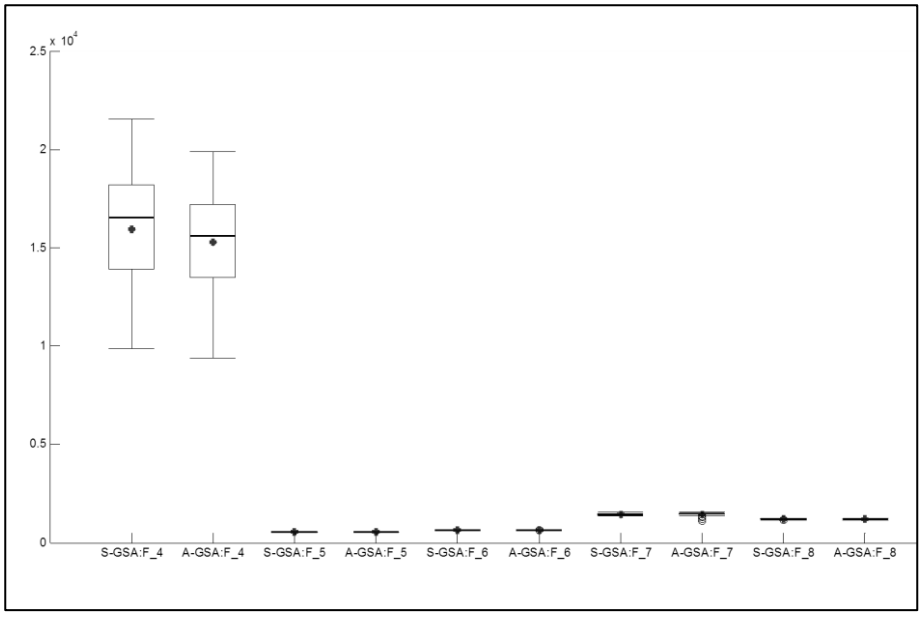

Figure 3. Boxplot for Test on Multimodal Functions (F_4, F_5, F_6, F_7 and F_8)

4. Conclusion

Two variants of GSA algorithm known as S-GSA and A-GSA are studied and compared here. The agents in S-GSA are updated synchronously while agents in A-GSA are updated asynchronously. The results are statistically analysed and it is found that both versions of GSA have similar performance. A-GSA performed as good as S-GSA but has a higher parallelization potential. The finding also shows that GSA has good exploration ability but lacking in exploitation. Therefore, enhancement of exploitation by GSA is a good research topic to be explored in the future.

Acknowledgement

This research is funded by the Department of Higher Education of Malaysia under the Fundamental Research Grant Scheme (4F374, FRGS/1/2012/TK06/MMU/03/7) and Dana Pembudayaan Penyelidikan (RDU121403). This work is also funded by Universiti Malaya under UM-UMRG Scheme (CG031-2013) and UM-Postgraduate Research Grant (PG097-2013A). The authors also would like to acknowledge the anonymous reviewers for their invaluable comments and insights.

References

[1] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A Gravitational Search Algorithm," *Inf. Sci. (Ny).*, vol. 179, no. 13, pp. 2232–2248, Jun. 2009.

[2] T. Nadzion, T. Ibrahim, T. Marapan, S. H. Hasim, A. F. Zainal, N. Omar, N. A. Nordin, H. I. Jaafar, K. Osman, Z. A. Ghani, S. Faisal, M. Hussein, and A. T. P. Published, "A Brief Analysis of Gravitational Search Algorithm (GSA) Publication from 2009 to," in *International Conference Recent treads in Engineering & Technology (ICRET'2014)*, 2014, no. May 2013, pp. 16–24.

[3] T. Eldos and R. Al Qasim, "On The Performance of the Gravitational Search Algorithm," vol. 4, no. 8, pp. 74–78, 2013.

[4] A. Chatterjee, G. Mahanti, and N. Pathak, "Comparative performance of gravitational search algorithm and modified particle swarm optimization algorithm for synthesis of thinned scanned concentric ring array," *Prog. Electromagn. ...*, no. September 2010, pp. 331–348, 2010.

[5] H. C. Shamsudin, A. Irawan, Z. Ibrahim, A. F. Z. Abidin, S. Wahyudi, M. A. A. Rahim, and K. Khalil, "A Fast Discrete Gravitational Search Algorithm," in *Computational Intelligence, Modelling and Simulation (CIMSiM), 2012 Fourth International Conference on*, 2012, pp. 24–28.

[6] Z. Ibrahim, B. Muhammad, K. H. Ghazali, K. S. Lim, S. W. Nawawi, and Z. M. Yusof, "Vector Evaluated Gravitational Search Algorithm (VEGSA) for Multi-objective Optimization Problems," in *Computational Intelligence, Modelling and Simulation (CIMSiM), 2012 Fourth International Conference on*, 2012, pp. 13–17.

[7] S. Mirjalili and S. Z. M. Hashim, "A new hybrid PSOGSA algorithm for function optimization," in *Computer and Information Application (ICCIA), 2010 International Conference on*, 2010, pp. 374–377.

[8] N. A. Ab Aziz, Z. Ibrahim, S. W. Nawawi, M. Mubin, I. Ibrahim, and M. Z. Mohd Tumari, "Synchronous vs Asynchronous Gravitational Search Algorithm," in *First International Conference on Artificial Intelligence, Modelling & Simulation*, 2013, pp. 29–34.

[9] J. Rada-Vilela, M. Zhang, and W. Seah, "A performance study on synchronicity and neighborhood size in particle swarm optimization," *Soft Comput.*, vol. 17, no. 6, pp. 1019–1030, Feb. 2013.

[10] J. Liang, B. Qu, and P. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization," no. December 2013, 2013.

[11] V. Coleman, "The DEME Mode: An Asynchronous Genetic Algorithm," University of Massachusetts, Amherst, MA, USA, 1989.
[12] B.-I. Koh, A. D. George, R. T. Haftka, and B. J. Fregly, "Parallel asynchronous particle swarm optimization.," *Int. J. Numer. Methods Eng.*, vol. 67, no. 4, pp. 578–595, Jul. 2006.
[13] A. Carlisle and G. Dozier, "An Off-The-Shelf PSO," in *In Workshop on Particle Swarm Optimization*, 2001.