mekatronika – Journal of Intelligent Manufacturing & Mechatronics
VOL. 01, ISSUE 02, 81 – 92
DOI: https://doi.org/10.15282/mekatronika.v1i2.4989

ORIGINAL ARTICLE

Random Synchronous Asynchronous PSO – A Particle Swarm Optimization Algorithm with a New Iteration Strategy

Nor Azlina Ab. Aziz[1], Nor Hidayati Abd Aziz[1], Tasiransurini Ab Rahman[2], Norrima Mokhtar[3], and Marizan Mubin[3]

[1]Faculty of Engineering and Technology, Multimedia University, Melaka, Malaysia.
[2]Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, Johor, Malaysia.
[3]Faculty of Engineering, University of Malaya, Kuala Lumpur, Malaysia.

ABSTRACT –Particle swarm optimisation (PSO) is a population-based stochastic optimisation algorithm. Traditionally the particles update sequence for PSO can be categorized into two groups, synchronous (S-PSO) or asynchronous (A-PSO) update. In S-PSO, the particles' performances are evaluated before their velocity and position are updated, while in A-PSO, each particle's velocity and position is updated immediately after individual performance is evaluated. In another study, a random asynchronous PSO (RA-PSO) has been proposed. In RA-PSO, particles are randomly chosen to be updated asynchronously, the randomness improves swarm's exploration. RA-PSO belongs to the asynchronous group. In this paper, a new category; hybrid update sequence is proposed. The new update sequence exploits the advantages of synchronous, asynchronous, and random update methods. The proposed sequence is termed as, random synchronous-asynchronous PSO (RSA-PSO). RSA-PSO divides the particles into groups. The groups are subjected to random asynchronous update, while the particles within a chosen group are updated synchronously. The performance of RSA-PSO is compared with the existing S-PSO, A-PSO, and RA-PSO using CEC2014's benchmark functions. The results show that RSA-PSO is superior to both A-PSO and RA-PSO, and as good as S-PSO

ARTICLE HISTORY
Received: 3 March 2019
Accepted: 6 Jun 2019

KEYWORDS
PSO
Random
Synchronous
Asynchronous

Introduction

Particle swarm optimisation (PSO) algorithm, was introduced by Kennedy and R. Eberhart in 1995 [1]. It looks for optimal solution of an optimisation problem by mimicking the social behaviour seen in nature, such as flock of birds looking for food. In PSO, these organisms are represented by a swarm of agents called particles. The particles move within the search area looking for optimal solution by updating their velocity and position. These values are influenced by the personal experience of the particles and their social interaction.

PSO has gained a lot of interest since its introduction. Numerous researches had been conducted involving PSO. Some of these works focuses on improvement of the algorithm through introduction of new parameters such as inertia weight [2] and constriction factor [3]. The PSO has also been improved to solve specific type of problems such as multiobjective optimization [4], discrete optimization [5-6] and dynamic optimization problem [7-8]. Application of PSO in solving real world optimization problem is also a popular trend. For example PSO had been successfully applied in robotics [9], biomedical optimization [10], and wireless sensor networks [11].

However, there are a few fundamental aspects of PSO which are not getting much attention, such as the synchronicity of the particle update sequence, which is also known as iteration strategy [12]. The traditional PSO iteration strategies can be divided into two categories, synchronous and asynchronous, as shown in Figure 1.

In the original PSO, a particle's information on the neighbourhood's best found solution is updated after the fitness of the whole swarm is evaluated. This version of PSO algorithm is known as synchronous PSO (S-PSO). The synchronous update in S-PSO provides perfect information on the fitness of the whole swarm based on the members' positions. Thus, allowing the swarm to choose the best neighbour and exploit the information of the best position provided by this neighbour. However, this could cause the particles to converge too fast. Many works has reported that synchronous update leads to a strong exploitation by S-PSO. However, according to [13], if the improvement of the best found solution is marginal, the synchronous update is not only reducing the exploration, but it also hinders the particles from exploiting and benefiting from the information available.

*CORRESPONDING AUTHOR | Nor Azlina Ab. Aziz | ✉ azlina.aziz@mmu.edu.my

mekatronika – Journal of Intelligent Manufacturing & Mechatronics
VOL. 01, ISSUE 02, 81 – 92
DOI: https://doi.org/10.15282/mekatronika.v1i2.2504

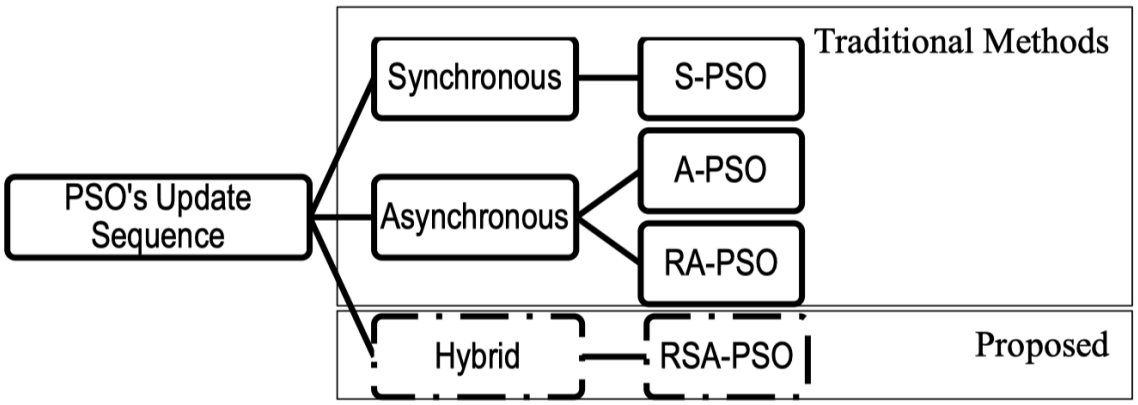

Figure 1. Categories of PSO's sequence update.

Another variation of PSO, known as asynchronous PSO (A-PSO), has been discussed in [14]. In A-PSO, the particles evaluate their fitness and update their velocity and position on their own without the need to synchronize with the whole swarm. As soon as a particle finished evaluating its fitness, it immediately identifies the best solution of the whole swarm and updates its velocity and position. Hence the particles updated at the beginning of an iteration use more information from the previous iteration, while particles at the end of the iteration use more information from the same iteration to determine the best solution of the swarm. This allows various leads from different best solutions, thus providing more exploration by the swarm.

Recently, random A-PSO (RA-PSO) has been introduced [15-16]. The RA-PSO belongs to asynchronous update group. In RAPSO, particles to be updated are selected randomly. Therefore, in an iteration, some particles might be updated more than once while other particles may not be updated at all. The randomness helps to prevent particles from being trapped in local optima and encourage more exploration. This is due to various degree of information within the swarm, because some particles might not be updated for several iterations thus possessing outdated information while others might be updated more than once in a single iteration.

In this study, synchronous, asynchronous, and random updates are merged so that the advantages of each of these methods can be utilized and the weaknesses can be overcomed. The proposed random synchronous-asynchronous PSO (RSA-PSO) algorithm divides the particles into smaller groups. The group to be updated are randomly chosen one at a time, asynchronously. The particles within a chosen group are updated synchronously. The search for the optimal solution by the particles in RSA-PSO is led by the best member of the groups and the swarm's best. The RSA-PSO improves the performance of PSO by balancing the exploitation provided by synchronous update, with the exploration by random asynchronous update. The RSA-PSO is a method belongs to new category of update strategy; hybrid update strategy. The CEC2014's benchmark functions for single objective real-parameter numerical optimization are used to evaluate the performance of RSA-PSO and the existing methods, S-PSO, A-PSO and RA-PSO. The results of the existing methods show that stronger exploitation in S-PSO is crucial in ensuring good performance. The RSA-PSO performs as good as S-PSO which is the best update strategy among the traditional methods.

PSO's Update Sequence

Traditionally PSO is either implemented as a synchronous update algorithm or asynchronous update. Synchronous update method is the typical method used in PSO while asynchronous update is another available approach. Asynchronous update is a more accurate natural model, it increases the potential of parallelization of an algorithm [17].

Synchronous update

In PSO, the search for optimal solution is conducted by a swarm of P particles. At time t, particle ith has a position, $x_i(t)$, and velocity, $v_i(t)$. The position represents a solution suggested by the particle while velocity is the rate of change to the next position with respect to the current position. At the start of the algorithm, these two values (position and velocity) are randomly initialised. In the subsequent iterations the search process is conducted by updating these values until a position with ideal fitness is attained or maximum number of iteration, T, is reached. The position and velocity are updated using the following equations:

*CORRESPONDING AUTHOR | Nor Azlina Ab. Aziz | ✉ azlina.aziz@mmu.edu.my

$$v_i(t) = \omega v_i(t-1) + c_1 r_1\big(pBest_i(t) - x_i(t-1)\big) + c_2 r_2\big(gBest(t) - x_i(t-1)\big) \quad (1)$$

$$x_i(t) = v_i(t) + x_i(t-1) \quad (2)$$

To prevent the particles from venturing too far from the feasible region, the $v_i(t)$ value is clamped to $\pm V_{max}$. If the value of V_{max} is too large, the exploration range is too wide. However, if it is too small, particles will favour the local search. In equation (1), c_1 and c_2 are the learning factors that control the effect of the cognitive and social influence on a particle. Typically, both c_1 and c_2 are set to 2. Two independent random numbers r_1 and r_2 in the range of [0.0, 1.0] are incorporated in the velocity equation. These random terms provide stochastic behaviour to the particles. Inertia weight, ω, is a term added to control the particles momentum. The particles can switch to fine tuning by manipulating ω when a good area is found. To ensure convergence, a time decreasing inertia weight is more favourable than a fixed inertia weight. This is because a larger inertia weight at the beginning helps to find a good area through exploration and a small inertia weight towards the end (when typically a good area is already found) facilitates fine tuning.

An individual success in PSO is affected not only by the particle's own effort and experience but also by the information shared by its surrounding neighbours. As shown in equation (1), the particle's velocity is updated using $pBest_i(t)$, which is the best position found so far by particle ith and $gBest(t)$, which is the best position found by the swarm up to t th iteration.

The particle's position, $x_i(t)$, is updated using equation (2), in which a particle's next search is launched from its previous position, $x_i(t\text{-}1)$. Typically, $x_i(t)$ is bounded to prevent the particles from searching in an infeasible region. The fitness of $x_i(t)$ is evaluated by a problem-dependent fitness function. Therefore, for a swarm with P number of particles, the fitness evaluation is done P times per iteration. Thus, the maximum number of fitness evaluation by S-PSO in a run is $(P \times T)$.A new position, $x_i(t)$ with a better fitness than the $gBest(t\text{-}1)$ or $pBest_i(t\text{-}1)$ or both is saved as $gBest(t)$ or $pBest_i(t)$, otherwise the old values are adopted.

Since in S-PSO the $pBest_i$ and $gBest$ are updated after all the particles are evaluated, this version of PSO is also known as synchronous PSO (S-PSO). Synchronous update allows the particles to have a complete view of the whole swarm positions and their fitness before selecting $gBest$. Thus, allowing the swarm to exploit this information so that a better solution can be found. However, this may cause the particles in S-PSO to converge faster and prematurely. The S-PSO algorithm is shown in Figure 2.

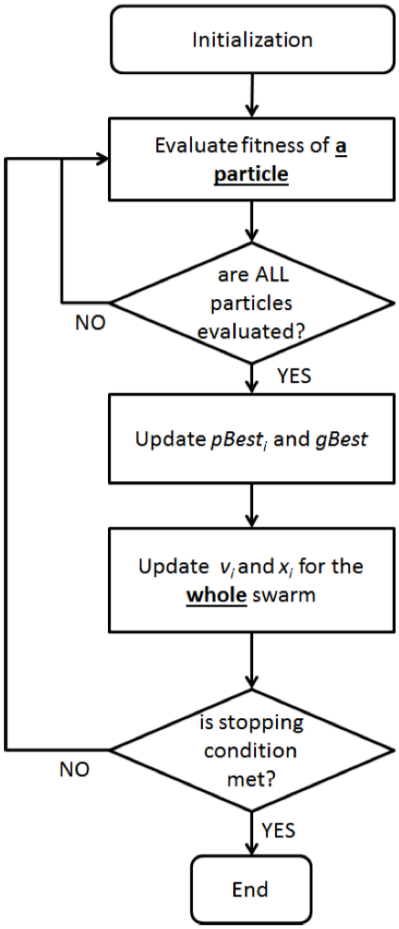

Figure 2. The flowchart of the S-PSO.

Asynchronous update

In synchronous update a particle needs to wait for the whole swarm to be evaluated before it can move to a new position and continue its search. Hence, the first evaluated particle is idle for the longest time, waiting for the whole swarm to be updated. PSO is a nature inspired algorithm. In nature, an individual is typically free to move without the need to synchronize its move with others. This concept is adopted in asynchronous update, where the particles are updated independently without synchronization with the whole swarm.

The flowchart in Figure 3 shows the A-PSO algorithm. A particle evaluates its fitness. After that the particle immediately selects $gBest$ and updates its $pBest$. The $gBest$ is selected depending on the swarm conditions during a particular particle's update process. Using the latest $gBest$ and $pBest$ a particle updates its velocity and position using the same equations as S-PSO. This process is then continued by the next particles until either ideal solution is found or T iterations are reached. Due to lack of synchronicity, in a single iteration the particles might use various values of $gBest$, leading to more exploration. Other than the variation of $gBest$ used, the lack of synchronicity in A-PSO solves the issue of the idle particles faced in S-PSO.

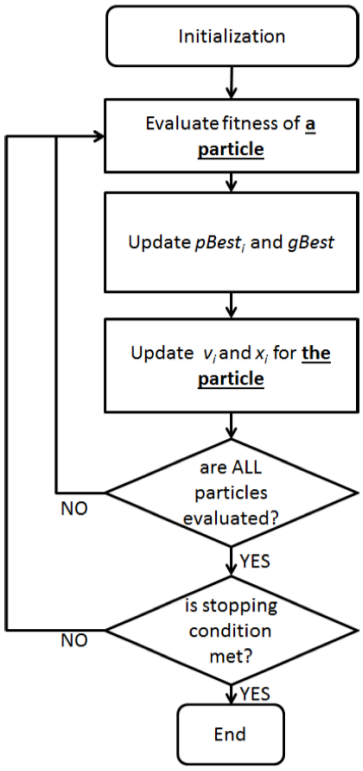

Figure 3. The flowchart of the A-PSO.

Even though, the flow of A-PSO is different than S-PSO, the fitness function is still called for P times per iteration, once for each particle. This is similar to S-PSO. Therefore, the maximum number of fitness evaluation is ($P \times T$).

Asynchronous update enables the update sequence of the particles to change dynamically and a particle to be updated more than once [15]. The change in the update sequence offers different levels of available information among the particles, this can prevent the particles from being trapped in local optima [15]. This is the characteristic manipulated by RA-PSO for improving the performance of the original A-PSO algorithm.

Random asynchronous PSO (RA-PSO) is a variation of A-PSO algorithm [15]. In RA-PSO the particles to be updated are chosen randomly with repetition allowed. Therefore, a particle can be updated more than once or none at all in a particular iteration. The randomness causes the swarm to have mixture state of particles by the end of each iteration, some particles possessing up to date information while some holding outdated information. This provides various degrees of information within the swarm. Since the selection of the particles is done randomly, the information flow is different from one iteration to another. This improves the exploration of the particles and prevents them from being trapped in local optima.

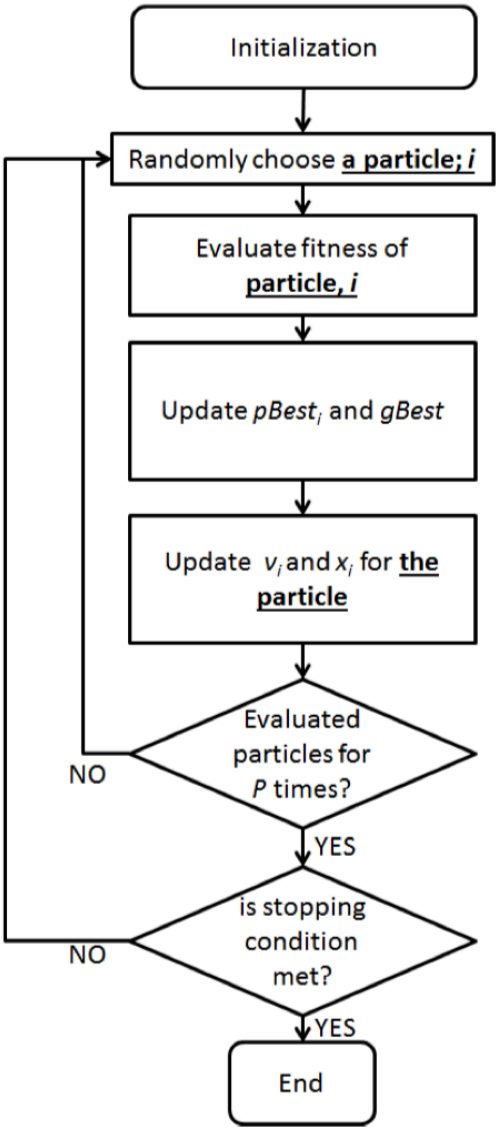

Figure 4. The flowchart of the RA-PSO.

The RA-PSO algorithm is presented in Figure 4. In each iteration, particles to be evaluated are chosen P times. Every time a particle is chosen, its fitness is evaluated followed by the velocity and position update. This is repeated for T iterations or until ideal fitness is obtained. Therefore, in a single run, RA-PSO performs at most ($P \times T$) fitness evaluations. This is similar to S-PSO and A-PSO algorithms.

The RA-PSO algorithm performs better than the original A-PSO in unimodal problems and as good as the original A-PSO in multimodal problems [15]. It also has a faster convergence rate than A-PSO. In a large neighbourhood swarm, RA-PSO performs better than S-PSO, as the randomness and asynchronicity of the particles are preventing them from being stagnant in local minima, which is the problem suffered by S-PSO.

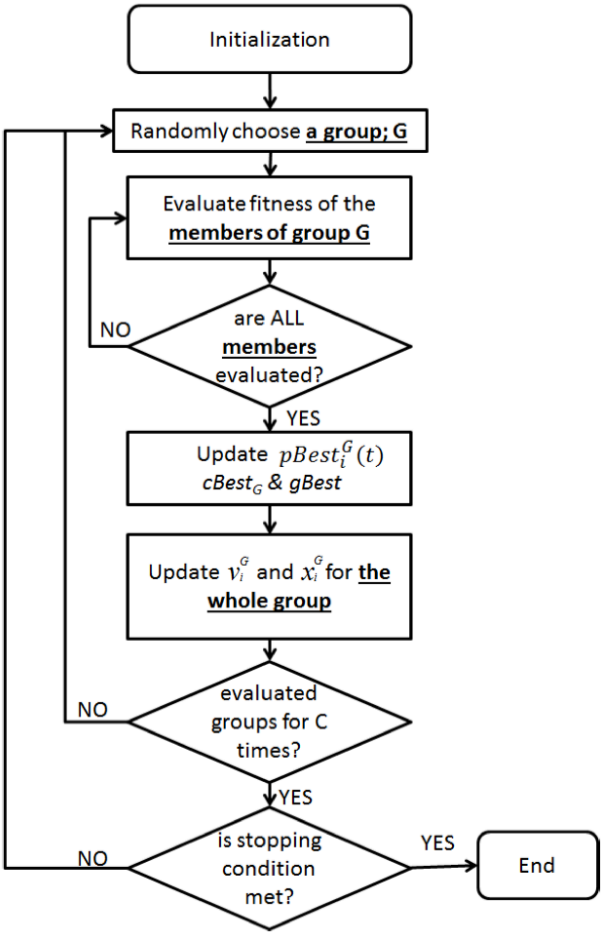

Figure 5. The flowchart of the RSA-PSO.

The Proposed Random Synchronous Asynchronous PSO

The advantage of synchronous update is strong exploitation which leads to good solution. Meanwhile, the variety of *gBest* information used in asynchronous update contributes to the strength of A-PSO which is diversity and exploration. RA-PSO enhances the exploration of APSO through randomization of the particles' update sequence. In the proposed random synchronous-asynchronous PSO (RSA-PSO), the advantage and strength of the synchronous, asynchronous, and random update methods are exploited so that a better variation of PSO is attained. The RSA-PSO update strategy does not fall within synchronous or asynchronous update, it is a hybrid method. The proposed algorithm is shown in Figure 5.

The algorithm starts with initialization of particles. The particles in RSA-PSO are divided into C groups, which consist of N number of particles each. Initially, C central particles, one for each group, are randomly placed in the search space. This is followed by random placement of (N-1) number of members for each group. The random placements of the members are within the radius of $\pm\Delta$ from the central particle of their respective group. The Δ is defined as the initial maximum distance of a particle from the central particle of its group. This parameter is only used once throughout the execution of the algorithm, which is during the initialization phase. The membership of the groups' remains fixed throughout the search process. The total number of particles, P, for RSA-PSO algorithm is $C \times N$.

In a single iteration, the algorithm randomly chooses the groups to be updated. The selection is done one by one for C times. Similar to RA-PSO, repetition is allowed. Therefore, there is a possibility that a group is updated more than once in a single iteration or may not be selected at all. Thus, at the end of an iteration, the swarm may consist of groups having updated velocities and positions, and groups with velocities and positions from previous search.

mekatronika – Journal of Intelligent Manufacturing & Mechatronics
VOL. 01, ISSUE 02, 81 – 92
DOI: https://doi.org/10.15282/mekatronika.v1i2.2504

Table 1. Parameter setting.

Parameter	Value
Number of runs	50
Number of iterations	2000
Number of particles	100
Velocity clamping, V_{max}	100
Inertia weight, ω	0.9-0.5
Cognitive coefficient, c_1	2
Social coefficient, c_2	2

Table 2. Setting for the additional parameters in RSA-PSO.

Parameter	Value
Number of group, C	10
Group size (particles per group)	10
Initial distance to group centre, Δ	50% of the size of the search space

The particles of a chosen group, G, are updated synchronously. The performance of all the members of the group is evaluated before their $pBest_i^G$ are identified. The $cBest_G$ is the best $pBest_i^G$ of group G. The velocity at iteration t of particle ith that belongs to group G, $v_i^G(t)$, is updated using the following equation:

$$v_i^G(t) = \omega v_i^G(t-1) + c_1 r_1 \left(cBest_G(t) - x_i^G(t-1)\right) + c_2 r_2 \left(gBest(t) - x_i^G(t-1)\right) \quad (3)$$

Equation (3) shows that the information used to update the velocity are the current group's best, $cBest_G(t)$ and the global best, $gBest(t)$. The best $cBest_G(t)$ is the $gBest(t)$. The position is updated as follows:

$$x_i^G(t) = v_i^G(t) + x_i^G(t-1) \quad (4)$$

In RSA-PSO, when a group is chosen, the particles within the group are evaluated. Hence, fitness function is called for N times per group. The groups to be updated are chosen randomly for C times per iteration. Therefore, in total, $C \times N$ times of fitness evaluation is conducted every iteration. $C \times N$ is equivalent to total number of particles in the swarm, P. Thus, the maximum number of fitness evaluation by RSA-PSO in a run which is limited to T iteration is $(P \times T)$. This is similar as S-PSO, A-PSO, and RA-PSO.

Experiments

The proposed RSA-PSO and the existing S-PSO, A-PSO, and RA-PSO were implemented using MATLAB. The parameter settings are summarized in Table I. Every experiment was subjected to 50 runs. The velocity of every particle is initialized randomly within the velocity clamping range, ±Vmax . The position of the particles was randomly initialized within the search space. A linear decreasing inertia weight ranging from 0.9 to 0.5 was employed. The cognitive and social learning factors were set to 2. The search was terminated once the number of iterations reaches 2000 or ideal fitness is attained. The parameters setting for the additional parameters in RSAPSO are given in Table II. Exclusively for RSA-PSO, the members of the groups were initialized randomly around the groups' central particles and based on Δ value.

The CEC2014's benchmark functions for single objective real-parameter numerical optimization are used here to evaluate the performance of RSA-PSO, S-PSO, A-PSO and RA-PSO. The benchmark functions consist of three rotated unimodal functions, thirteen multimodal problems, six hybrid functions and eight composition functions. The functions are listed in Table III. The multimodal functions have many local optima, the functions are either shifted or shifted and rotated, which increase their complexity. The hybrid functions consist of more than one function, thus naturally are more difficult to be solved. Composition functions combined unimodal, multimodal and hybrid functions with local optima trapped is set at the origin. The value of N shown in Table III for hybrid and composite functions indicates number of the basic functions used.

The solutions found by the algorithms tested are presented using boxplot. A boxplot shows the quality and also the consistency of an algorithm's performance. The size of the box shows the magnitude of the variance of the results. Thus, smaller box suggests a more consistent performance of an

*CORRESPONDING AUTHOR | Nor Azlina Ab. Aziz | ✉ azlina.aziz@mmu.edu.my

algorithm. Since the benchmark functions used in this study are minimisation problems, a lower boxplot is desirable as it indicates better quality of the solutions found.

The boxplots show that the solutions found are not normally distributed, therefore, the algorithms are compared using nonparametric test option in the KEEL software [18]. The test chosen is the Friedman test with significance level $\alpha = 0.05$. This test is suitable for comparison of more than two algorithms [19]. If the Friedman statistic value is lesser than the critical value, this signifies the algorithms tested are identical to each other. Otherwise, significant differences exist and the algorithms are then compared using a post hoc procedure. The Holm post hoc procedure is employed in this work to pin point where does the difference occurs.

Results and Discussions

RSA-PSO vs S-PSO, A-PSO, RA-PSO

The boxplots in Figure 6 show the quality of the results for unimodal test functions obtained by the S-PSO, A-PSO, RA-PSO, and the proposed RSA-PSO algorithms. It can be observed from the figures that RSA-PSO consistently gives good performance in all unimodal functions tested with small variance. It has better performance than A-PSO and RA-PSO. In comparison to the performance of S-PSO, RSA-PSO is as good as S-PSO.

The results of the test on multimodal problems are shown in Figure 7. Similar to the boxplots for unimodal test functions, the boxplots show that the variance of the solutions found by RSA-PSO and S-PSO are small. The variance proves the consistency of their performance. The quality of the solutions found by RSA-PSO is on par with S-PSO and better than A-PSO and RA-PSO.

Same observation are made for hybrid functions (Figure 8) and composition function (Figure 9). RSA-PSO is on par with S-PSO. Both update sequence have low variance signifying consistent performance, while A-PSO and RA-PSO perform poorer and with bigger variance.

The algorithms' mean for each test functions and their average rank are shown is Table IV. The means are shown in the boxplots using the * symbol. The Friedman statistic value shows that significant differences exist between the algorithms. Therefore, the Holm procedure is conducted. The results in Table V show that the performance of RSA-PSO is on par with S-PSO and both RSA-PSO and S-PSO have significant difference in performance with A-PSO and RAPSO. Good performance of RSA-PSO is contributed by the fact that the particles are learning and exploiting information from two particles with better information, which are *gBest* and *cBest*. The random asynchronous update of the groups provides exploration in RSA-PSO thus avoiding premature convergence.

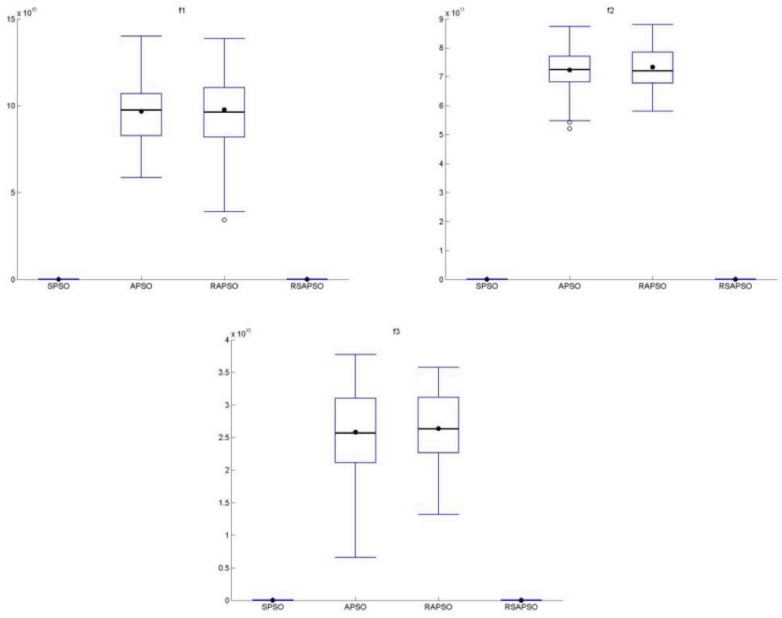

Figure 6. Result of experiments on unimodal functions.

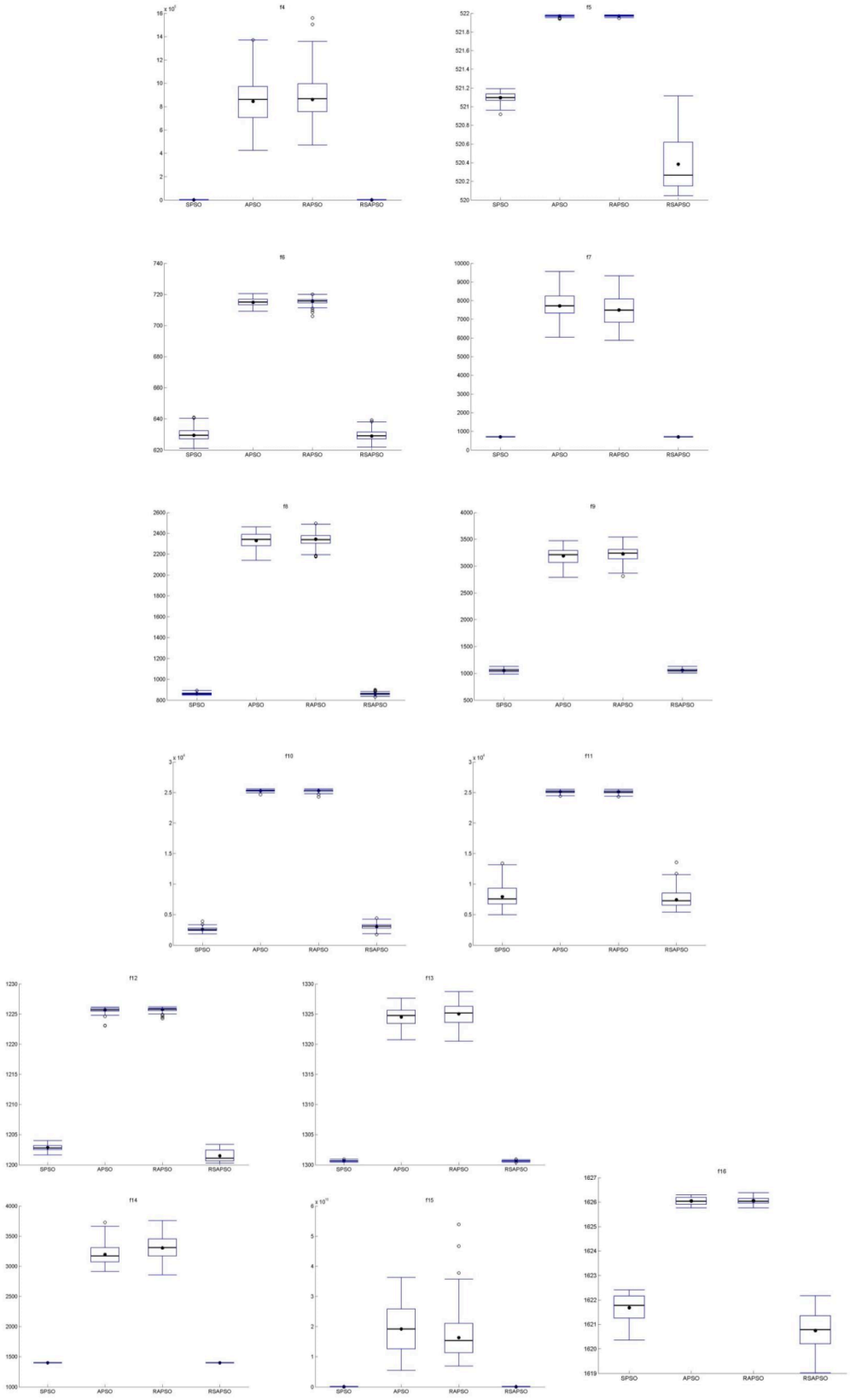

Figure 7. Result of experiments on multimodal functions.

Table 3. The CEC2014 test suite.

Function	Type of function	Ideal fitness
f1	Rotated High Conditioned Elliptic	100
f2	Rotated Bent Cigar	200
f3	Rotated Discus	300
f4	Shifted and Rotated Rosenbrock's	400
f5	Shifted and Rotated Ackley's	500
f6	Shifted and Rotated Weierstrass	600
f7	Shifted and Rotated Griewank's	700
f8	Shifted Rastrigin's	800
f9	Shifted and Rotated Rastrigin's	900
f10	Shifted Schwefel's	1000
f11	Shifted and Rotated Schwefel's	1100
f12	Shifted and Rotated Katsura	1200
f13	Shifted and Rotated HappyCat	1300
f14	Shifted and Rotated HGBat	1400
f15	Shifted and Rotated Expanded Griewank's plus Rosenbrock's	1500
f16	Shifted and Rotated Expanded Scaffer's	1600
f17	Hybrid function 1 (N=3)	1700
f18	Hybrid function 2 (N=3)	1800
f19	Hybrid function 3 (N=4)	1900
f20	Hybrid function 4 (N=4)	2000
f21	Hybrid function 5 (N=5)	2100
f22	Hybrid function 6 (N=6)	2200
f23	Composition function 1 (N=5)	2300
f24	Composition function 2 (N=3)	2400
f25	Composition function 3 (N=3)	2500
f26	Composition function 4 (N=5)	2600
f27	Composition function 5 (N=5)	2700
f28	Composition function 6 (N=5)	2800
f29	Composition function 7 (N=3)	2900
f30	Composition function 8 (N=3)	3000

Table 4. Average result.

Function	S-PSO	A-PSO	RA-PSO	RSA-PSO
f1	23317467.9762	96780027600.4945	96362016971.3520	20791531.9021
f2	1617145.2788	715631068852.1800	733952978460.1720	225780.7340
f3	20228.3905	25847082349.5529	26382211765.3402	7162.2740
f4	640.0601	856055.8889	886953.4268	643.5569
f5	521.0929	521.9687	521.9714	520.3845
f6	629.9419	714.8964	715.2115	629.3239
f7	700.0133	7721.0542	7500.9251	700.0159
f8	861.7606	2330.8271	2337.1607	860.9943
f9	1050.6320	3191.1875	3220.4362	1056.6355
f10	2592.7238	25277.0029	25257.9856	3048.3761
f11	8036.2319	25106.3156	25075.7069	7620.5766
f12	1202.8467	1225.5416	1225.6789	1201.4784
f13	1300.6240	1324.5121	1325.0278	1300.6324
f14	1400.5917	3207.6215	3305.0808	1400.5936
f15	1520.1586	19223967301.7630	18152570714.1591	1528.6849
f16	1621.6836	1626.0480	1626.0554	1620.7492
f17	2738876.6182	46499778121.8483	47802762561.2766	2859546.6989
f18	2697.6237	149643976950.6680	145441630392.0710	2593.0217
f19	1968.2023	106943.4179	111365.6296	1970.9642
f20	12246.3069	56038827597.0697	61576967650.9417	5040.4760
f21	1850848.7084	30661865535.3322	31687490044.9671	1291397.7185
f22	3121.5135	2112075226.7286	1852691609.1719	3111.3453
f23	2648.0627	24277.1419	25511.0805	2648.0427
f24	2676.3158	17339.1314	17956.4314	2676.6297
f25	2721.7840	10137.9569	10200.4733	2722.0877
f26	2771.4052	12577.4309	12364.0413	2776.4294
f27	3731.1536	98384.8497	93086.0752	3792.7514
f28	5374.6678	87626.3895	79320.8276	5287.5968
f29	11664505.2623	61162227650.7580	61435815978.3012	203485844.1391
f30	39538.7201	1475809138.3202	1620269889.7606	40752.5674
Average Friedman Rank	1.5	3.3333	3.6667	1.5

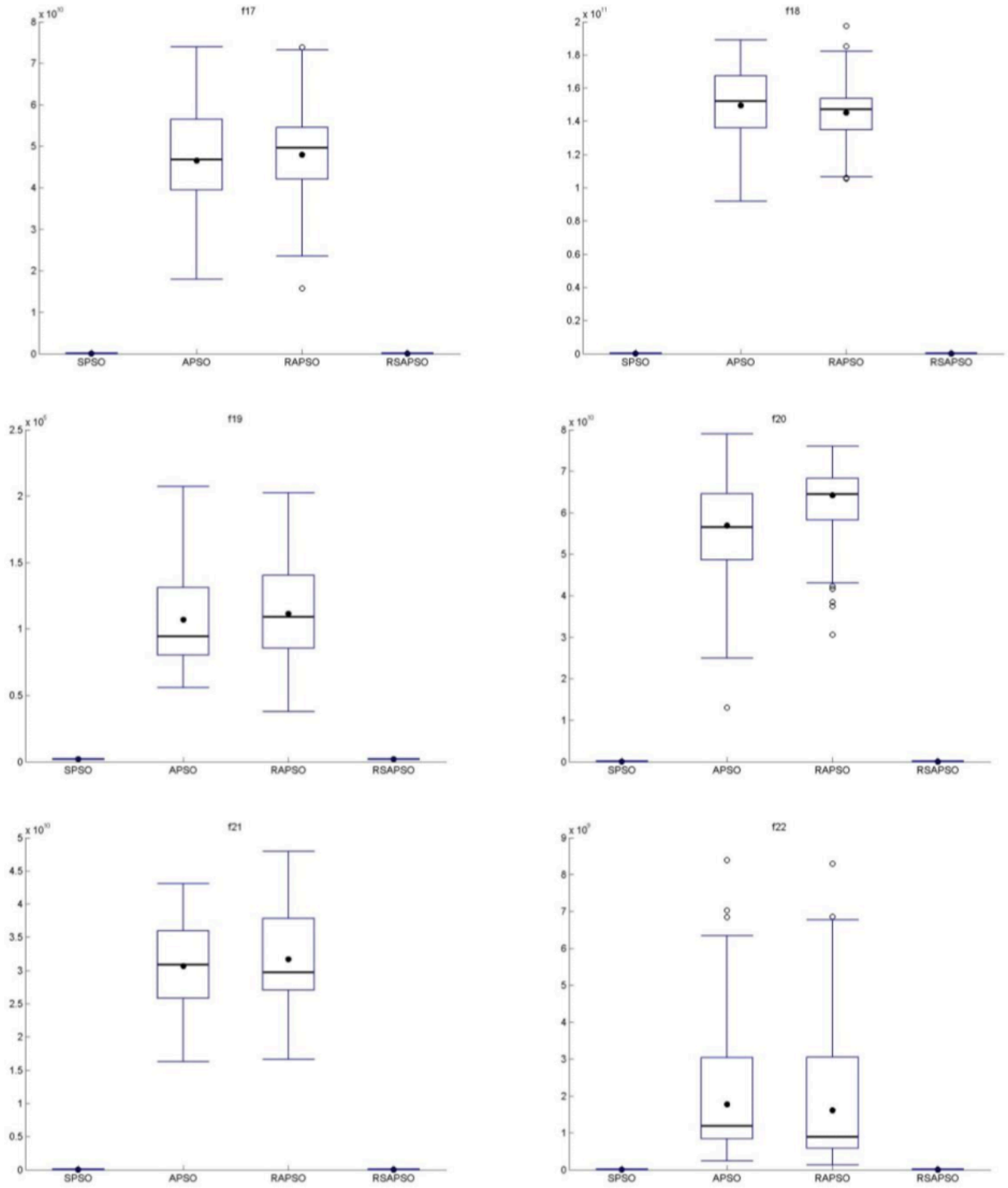

Figure 8. Result of experiments on hybrid functions.

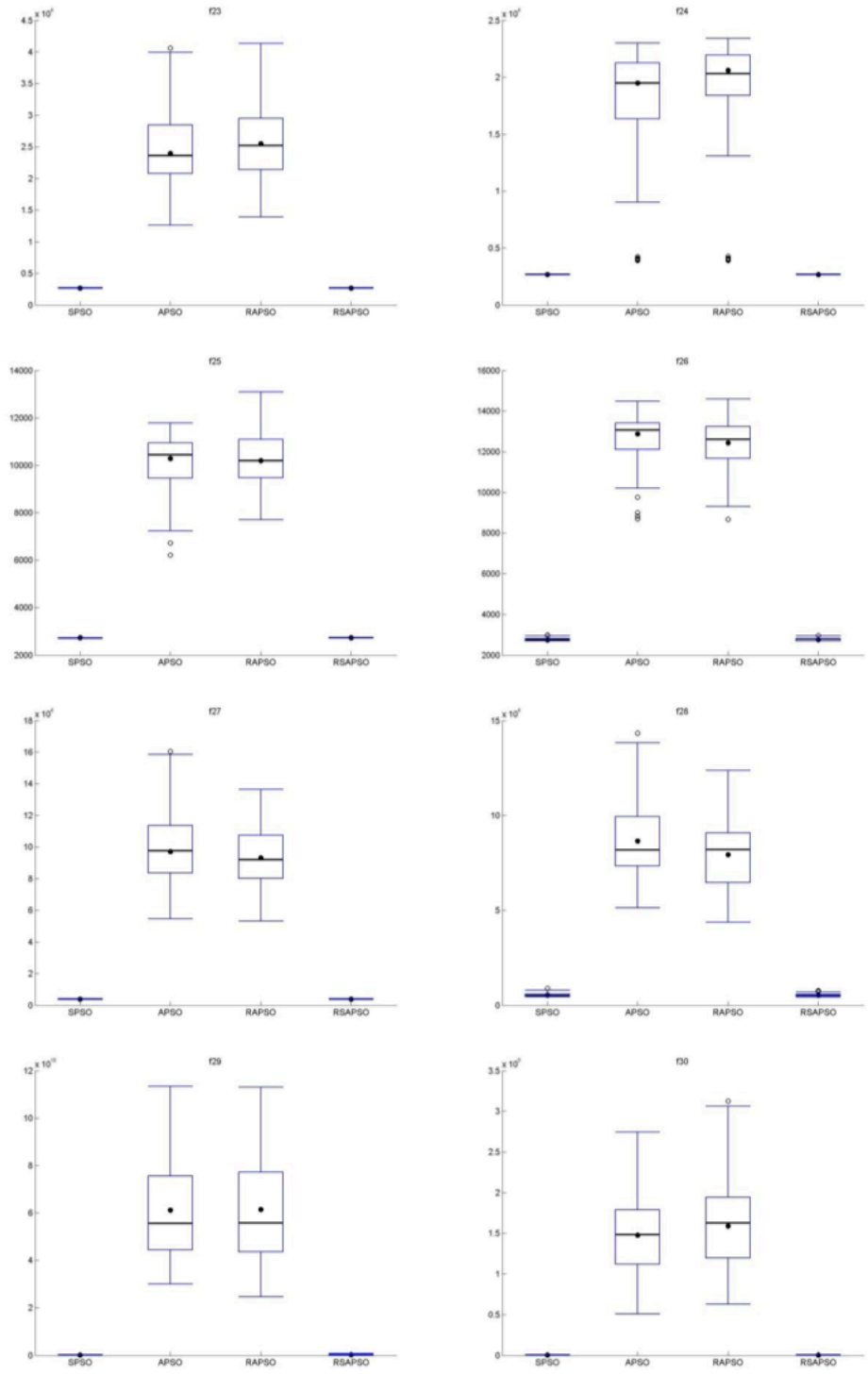

Figure 9. Result of experiments on composite functions.

Table 5. Statistical analysis.

Algorithms	z	p	Holm
RA-PSO vs. RSA-PSO	6.5	0	0.008333
S-PSO vs RA-PSO	6.5	0	0.01
A-PSO vs RSA-PSO	5.5	0	0.0125
S-PSO vs A-PSO	5.5	0	0.16667
A-PSO vs RA-PSO	1	0.317311	0.025
S-PSO vs RSA-PSO	0	1	0.0500

Conclusions

Random synchronous-asynchronous PSO algorithm (RSA-PSO) a new method belongs to hybrid update strategy is proposed in this paper. The particles in RSA-PSO are updated synchronously in groups. The groups, however, are chosen randomly and asynchronously updated, one group after another. A group's search is led by the group's best performer, $cBest_G$, and the best member of the swarm, *gBest*. The algorithm benefits from good exploitation and fine tuning provided by synchronous update, while it takes advantage of the exploration in the asynchronous update. The exploration is further enhanced by the random selection of the group to be updated. Statistical analysis performed shows that the performance of RSA-PSO is better than A-PSO and RA-PSO and on par as S-PSO.

References

[1] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. International Conference on Neural Networks, vol. 4, pp. 1942-1948.

[2] Shi, Y. and Eberhart, R. (1998). A modified particle swarm optimizer. IEEE International Conference on Evolutionary Computation, pp. 69-73.

[3] Clerc, M. and Kennedy, J. (2002). The particle swarm – explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation, vol. 6, no. 1, pp. 58-73.

[4] Shi, Y. and Eberhart, R. (1998). A modified particle swarm optimizer. IEEE International Conference on Evolutionary Computation, pp. 69-73.

[5] Kennedy, J. and Eberhart, R. (1997). A discrete binary version of the particle swarm algorithm. IEEE International Conference on Systems, Man, and Cybernetics.

[6] Ibrahim, I., Yusof, Z.M., Nawawi, S.W., Rahim, M.A.A., Khalil, K., Ahmad, H., and Ibrahim, Z. (2012). A novel multi-state particle swarm optimization for discrete combinatorial optimization problems. Fourth International Conference on Computational Intelligence, Modelling, and Simulation, pp. 18-23.

[7] Li, C. and Yang, S. (2012). A general framework of multipopulation methods with clustering in undetectable dynamic environments. IEEE Transactions on Evolutionary Computation, vol. 16, issue 4, pp. 556-577.

[8] Yang, S. and Li, C. (2010). A clustering particle swarm optimier for locating and tracking multiple optima in dynamic environments. IEEE Transactions on Evolutionary Computation, vol. 14, issue 6, pp. 959-974.

[9] Xue, S., Zhang, J. and Zeng, J. (2009). Parallel asynchronous control strategy for target search with swarm robots. International Journal of Bio-Inspired Computation, vol. 1, no 3, pp. 151-163.

[10] Mohamad, M.S., Omatu, S., Deris, S., Yoshioka, M., Abdullah, A. and Ibrahim, Z. (2013). An enhancement of binary particle swarm optimization for gene selection in classifying cancer classes. Algorithms for Molecular Biology, vol. 8, no. 15.

[11] Aziz, N.A., Mohemmed, A., and Zhang, M. (2010). Particle swarm optimization for coverage maximization and energy conservation in wireless sensor networks. Applications of Evolutionary Computation, pp. 51-60.

[12] Englebrecht, A.P. (2013). Particle swarm optimization: iteration strategies revisited. BRICS Congress on Computational Intelligence and 11th Brazillian Congress on Computational Intelligence, pp. 119-123.

[13] Voglis, C.A., Parsopoulos, K.E., and Lagaris, I.E. (2012). Particle swarm optimization with deliberate loss of information. Soft Computing, vol. 16, no. 8, pp. 1373-1392.

[14] Carlisle, A. and Dozier, G. (2001). An off-the-shelf PSO. Workshop on Particle Swarm Optimization.

[15] Rada-Vilela, J., Zhang, M., and Seah, W. (2011). Random asynchronous PSO. The 5th International Conference on Automation, Robotics, and Applications, pp. 220-225.

[16] Rada-Vilela, J., Zhang, M., and Seah, W. (2013). A performance study on synchronicity and neighborhood size in particle swarm optimization. Soft Computing, vol. 17, no. 6, pp. 1019-1030.

[17] Koh, B.-I, George, A.D., Haftka, R.T. and Fregly, B.J. (2006). Parallel asynchronous particle swarm optimization. International Journal for Numerical Methods in Engineering, vol. 67, no. 4, pp. 578-595.

[18] http://sci2s.ugr.es/keel/download.php

[19] Derrac, J., Garcia, S., Molina, D. and Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation, vol. 1, no. 1, pp. 3-18.

*CORRESPONDING AUTHOR | Nor Azlina Ab. Aziz | ✉ azlina.aziz@mmu.edu.my