Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 2458-9403
Vol. 4 Issue 1, January - 2017

Improved Clustered Optimized Particle Swarm Optimization for Global Optimization in Multimodal Problems for Systems with Limited Resources

M.H. El-Saify[a], G.A. El-Sheikh[b], A.M. El-Garhy[c]
[a]Ph. D. student, Electronics, Communications, and Computers Department, Helwan University, Cairo, Egypt. mhelsaify@hotmail.com
[b]Head of the Electronics and Communications Department, Pyramids High Institute (PHI) for Engineering and Technology, 6-October, Giza, Egypt. gaelsheikh@gmail.com
[c]Electronics, Communications, and Computers Department, Faculty of Engineering, Helwan University, Cairo, Egypt. agarhy2003@yahoo.co.in

***Abstract*— This paper proposes an improved algorithm of Particle Swarm Optimization (PSO) based on stage-structured algorithm and added features to enhance the ability of finding the global optimum in multimodal multi-dimensional optimization problems, which may have local optima, using low computational effort. Search space clustering based on Euclidian distance, particles crossover and mutation are added features to enhance the ability of exploration and exploitation of the algorithm. The algorithm divides the optimization process into three stages, namely; *global search*, *local search* and *final stage*. The parameters of the PSO and the added features are modified by the algorithm in each stage to achieve the stage goal. Furthermore, if the performance is not satisfactory, the same algorithm is used to optimize the parameters of the proposed algorithm. Fifteen benchmark multimodal test functions, that can be expanded to multi-dimensions, are used to test the proposed algorithm and they demonstrated its superiority to find the global optimum over other algorithms with minimal computational effort which is vital for many modern/smart limited resources devices (such as smart phones, Internet of Things devices (IoT), self-driving cars, ... etc).**

Keywords*— particle swarm optimization (PSO); crossover; mutation; optimization; limited resources.

I. INTRODUCTION

The optimization process is a milestone in a tremendous number of applications that impact our modern life. Economics (e.g. finance, banking, insurance,...etc), operations research (improving decision making), mechanics (e.g. design problems), industry and many fields of engineering (e.g. electrical engineering, petroleum engineering, control, modeling,...etc) are common fields that regularly use optimization techniques. Since the complexity of the optimization problems increases, the researchers increasingly rely on the heuristic optimization methods. Heuristic optimization provides efficient solutions for the complex real-world problems that are too complex or too slow to be solved by exact methods. Although heuristic optimization is approximate, (i.e. it does not grantee that the solution is optimal), it provides quick good solution for large complex problems without the computational drawbacks of the exact methods. Among these heuristic techniques, Particle Swarm Optimization (PSO), Genetic Algorithms (GA) and their developments are still active topics since they were introduced [1-4].

Since PSO was introduced, it attracted many researches because of its simplicity, powerful and rapid convergence. On the other hand, as many other heuristic techniques, it may rapidly converge to a local optimum (trapped). A further drawback of the PSO is that stochastic approaches have problem dependent performance. Therefore, no single parameter settings are good for all problems. Changing the parameters of the PSO can affect both exploration and exploitation in a contradictory way. Therefore, it is a trade-off problem that requires adjustment according to the optimization problem in hand. The impact of the trade-off problem increases rapidly as the search space increases.

Many developments are introduced to PSO to overcome its drawbacks and enhance its performance [5]. Some researches develop the PSO algorithm itself [6, 7] and others introduce hybrid techniques with other optimization techniques [8-11]. The developed PSO, especially the hybrid ones, sometimes enhance the performance of the PSO at the expense of the PSO simplicity. In global optimization problems, the researchers usually test their algorithms' performances by applying high dimensional multimodal functions that have many local optima. The algorithm is evaluated according to its ability to find the global optimum without trapping in any local optimum. In this approach, sufficient number of iterations and population size is chosen according to the number of dimensions of the problem to allow the algorithms to achieve their best performances. Different approaches are applied in this research. At first, we study the degradation of algorithms' performance due to gradually increase of the problem dimensions without

sufficient afforded computations. In the second approach, we study how much ICO-PSO can use limited resources to achieve the same results of some well-known algorithms in the literature that used sufficient computations.

In addition to existing limited resources devices such as (smart phones, GPS-based devices, autonomous mobile robots, self-driving cars, ...etc), recently we have witnessed the rise of the term "Internet of Things (IoT)" where physical objects/devices are connected through internetworking in order to collect, exchange, analyze data or control objects. Vehicles, buildings, farms, medical monitoring devices, field operation assist for search and rescue operations are some of the applications that are subjected to IoT. This trend among others utilize huge amount of collected data that lead us to the other rising term "Big Data". In conclusion, we have many limited resources devices that have to deal with more complex problems with big data in hand. Powerful Algorithms, which needs minimal computational effort, is needed to do operations as optimization problems. Path planning optimization is an example of operations that sometimes need real time execution. This paper proposes an improved algorithm of PSO that maximize the ability of finding global optimum in multi-dimensional multimodal problems with minimal computational efforts. It uses low population size and requires low number of iterations that will result in significant reduction of required computation capability and memory. Moreover, it does not lack simplicity or low computational effort of the standard PSO. The proposed Improved Clustered Optimized PSO (ICO-PSO) algorithm adds external features to the PSO and divides the optimization process into three stages. ICO-PSO changes the parameters of the algorithm to achieve each stage goal. Moreover, this algorithm can be used to optimize its parameters to fit problems that are more complex. The rest of this paper is organized as follows; section II discusses the classic PSO algorithm while section III proposes the ICO-PSO algorithm in details. Finally, the experiments and results are discussed in section IV and the conclusion is depicted in section V. More details about the algorithm are listed in Appendix A.

II. STANDARD PSO AND ITS DEVELOPMENTS

Since PSO was introduced [12, 13], it has been used in many applications [14-16]. The PSO algorithm mimics the social behavior of bird flocking or fish schooling. Therefore, each particle in the swarm represents a particular solution of the problem in a population-based algorithm. The particles move in the search space to find the best solution of the problem. The particles initial positions and velocities are randomly generated then it is updated according to each particle previously best position and the overall swarm best position.

If the search space is D-dimensional space, then we can represent the position and velocity of the i^{th} particle of the swarm by D-dimensional vectors X_i and V_i respectively. Each particle maintains a memory of its previous best position P_i while the whole swarm best found position is denoted by P_{gbest}.

These terms can be represented mathematically as follows:

$$\begin{aligned} &X_i=(x_{i1}, x_{i2}, \dots , x_{id}, \dots , x_{iD}) \\ &V_i=(v_{i1}, v_{i2}, \dots , v_{id}, \dots, v_{iD}) \\ &P_i=(p_{i1}, p_{i2}, \dots , p_{id}, \dots, p_{iD}) \\ &P_{gbest}=(p_{g1}, p_{g2}, \dots , p_{gd}, \dots, p_{gD}) \\ &where\ \ i=1, 2, \dots , S\ \ and\ \ d=1, 2, \dots , d, \dots , D \end{aligned} \quad (1)$$

S is the population size of the swarm and D is the number of dimensions of the search space. The positions and velocities of the swarm particles at the iteration number $z+1$ are updated as follows:

$$v_{id}^{z+1} = w^{z+1} v_{id}^{z} + c_p r_1^z (p_{id}^z - x_{id}^z) + c_g r_2^z (p_{gd}^z - x_{id}^z) \quad (2)$$

$$x_{id}^{z+1} = x_{id}^z + v_{id}^{z+1} \quad (3)$$

where r_1 and r_2 are random numbers. w is the inertia weight which controls the influence of the previous velocity on the new velocity. It usually starts with big value to boost the global *exploration* of the search space, then decreases gradually to enhance the local *exploitation* of the nearby region. While *exploration* refers to exploring the search space widely to find good solutions, the *exploitation* refers to refinement of the search around good solution to find the best solution in this region. In many researches [7, 8, 16], w is chosen to decrease linearly from 0.9 to 0.4. w is calculated as follows:

$$w = 0.9 - \frac{0.5}{N} z \quad (4)$$

where N is the total or maximum number of iterations.

c_p and c_g are positive constants called coefficient of the self-recognition component and coefficient of the social component respectively. As declared by their names, c_p will determine how much the particle's movement will be influenced by its previously known best position while c_g determines how much the particle's movement will be influenced by the global best position of the whole swarm.

Many developments are performed in the PSO algorithm to enhance its performance. Some techniques are designed to avoid trapping in local optima in multimodal problems sacrificing the rapid convergence rate in unimodal problems while some techniques do the opposite. Local (ring) topological structure PSO (LPSO), introduced in [17], is designed for solving multimodal problems. It is based on a local topology to avoid premature convergence that may lead to trapping in local optima. In another research [18], quadratic interpolation PSO (QIPSO) is introduced for global optimization problems. It is based

on using quadratic crossover operator. The Fully Informed Particle Swarm algorithm (FIPS) [19] develops the canonical particle swarm by making the individuals fully informed of the entire neighborhood. This technique guides the particles to find the best solution. Other researchers suggested Dynamic Multiswarm PSO (DMS-PSO) [20] where the whole swarm is frequently divided and regrouped into small swarms. The information is exchanged among the swarms. In Comprehensive Learning PSO (CLPSO) [21], a learning strategy is suggested to discourage premature convergence. This strategy enables learning among the swarm historical best information. The fuzzy set theory is utilized to adjust PSO acceleration coefficients adaptively in Adaptive Fuzzy PSO algorithm (AFPSO) [22]. Further enhancement is achieved by incorporating AFPSO with quadratic interpolation and crossover operator to form a new variant AFPSO-QI [22]. In the global optimization of multimodal problems, the researchers face a trade-off problem between convergence rate and escaping the local optima. Therefore, some good algorithms in solving multimodal problems suffer bad performance or slow convergence rate in solving unimodal problems. Fusion Global-Local-Topology Particle Swarm Optimization (FGLT-PSO) algorithm [6] performs a global search over the entire search space with a fast convergence speed using hybridizing two local and global topologies in PSO to jump out from local optima. Finding the best fitting technique for each application is crucial to get the best outcome of the optimization process.

The aforementioned algorithms and many others are tested in terms of solution accuracy, which is finding the global optimum, and convergence rate. However, theses algorithms are tested in normal conditions, where adequate number of particles and iterations are used. In this work, we represent improved PSO technique, ICO-PSO, which find global maximum in multi-dimensional multimodal problems with limited population size and number of iterations.

III. ICO-PSO TECHNIQUE

The proposed ICO-PSO adds the following features to the standard PSO algorithm:

- *Crossover and mutation*: In order to avoid trapping in local optima, the particles do a crossover operation with a randomly chosen particle of the population. The new particle explores a different area of the search space and allows escaping the local optimum. Furthermore, mutation is done to the particles to allow more diversity of the searched space. Crossover and mutation are done according to certain probabilities that are changed by the algorithm during the optimization.
- *Search space clustering*: A predefined number of clusters is chosen. According to this number and the size of the search space, each particle classifies its own cluster every iteration. The cluster size is the size of the search space divided by the predefined number of clusters. The particle movement is influenced by its own known best position and best known positions of the swarm in the same cluster. The number of clusters, consequently the cluster size, is changed during the optimization.
- *Dividing the optimization process into three stages*: Each stage changes the algorithm parameters to achieve its goal. These stages are:
    - *Global search stage:* The goal of this stage is to explore the search space globally. The particles move rapidly (*w* is high). The particle movement is more dependent on its self-recognition component rather than its social component. The size of the clusters is small. The particles' movements are influenced by its near neighborhoods. The possibility of crossover and mutation is relatively high.
    - *Local search stage:* The goal of this stage is to boost the exploitation. The particles are searching locally around the good solutions to find the best local solution in each area. The particles move slowly and more dependent on the social component rather than the self-recognition component. The possibility of crossover and mutation is relatively small.
    - *Final stage:* The goal of this stage is to find the global optimum by converging to the best-found solution by the entire swarm and refining the search around it. The particles move quickly then gradually slowdown. The cluster size increases gradually to involve the entire search space. The particles' movements depend equally on both self-recognition component and social component. The possibility of crossover and mutation is very small.
- *Optimization of the algorithm:* The same algorithm can be used to optimize the main algorithm parameters in high complex multimodal functions. The results show good performance improvement.

The flow chart depicts the ICO-PSO algorithm is shown in Fig. 1. It starts with the initialization process.

Initial values are assigned to positions and velocities of the particles. It is usually chosen as a uniformly distributed random numbers, which cover the whole range of the search space. In addition, the following parameters are defined in the initialization process:

- *glob* and *loc* that specifies the limits of the global, local and final stages. They are integers chosen such that 1< *glob* < *loc* <*N*. (*N* maximum number of iterations).
- *clas* maximum number of clusters.

Then the iterations begin. For the first iteration, the fitness is computed for all particles, and particles positions X_i is considered also as particles best positions P_i. When, the iteration number, z, is $1 < z \leq glob$, it denotes the *global stage* while $glob < z \leq loc$ denotes the *local stage*. The *final stage* is defined as $z > loc$. The following parameters are defined according to the current stage:

- *cross* and *mut* that specifies the crossover and mutation probabilities respectively.
- *clasNum* that defines the number of clusters.
- Inertia weight, coefficient of the self-recognition component and coefficient of the social component.

The iteration of the swarm, the right hand side of Fig. 1, is done as follows. For each particle i, the Euclidean Distances between the particle position X_i and the best known positions of all particles $P=[P_1\ P_2 \ldots P_j \ldots P_S]$ are calculated as follows.

$$Ed_i=[ed_{i1}\ ed_{i2} \ldots ed_{ij} \ldots ed_{iS}] \quad (5)$$

$$ed_{ij} = \sqrt{\sum_{l=1}^{D}(x_{il} - p_{jl})^2} \quad (6)$$

ed_{ij} given by (6) represents the Euclidean Distance between the position of particle i, X_i, and the best known position of particle j, P_j. As shown in Fig. 2, the maximum Euclidean Distance is calculated from the search space boundaries. If the boundaries are unknown, it is considered as the maximum computed Euclidean Distance so far. For the particle i, if $ed_{ij}<(ed_{max}/clasNum)$, the particle best position P_j is considered within the particle i local cluster. P_j that has the best fitness within the local cluster of particle i is called P_{local}. It is used to update the position and velocity of the particle i as follows:

$$v_{id}^{z+1} = w^{z+1}v_{id}^{z} + c_p r_1^z(p_{id}^z - x_{id}^z) + c_g r_2^z(p_{local\_d}^z - x_{id}^z) \quad (7)$$

$$x_{id}^{z+1} = x_{id}^{z} + v_{id}^{z+1} \quad (8)$$

It is clear that the term P_{local} adds more flexibility to the optimization algorithm. As the cluster size changes, we can manipulate the exploration-exploitation trade-off problem to achieve the goal of each stage. When the number of clusters, *clasNum*, decreases to one, P_{local} becomes P_{gbest}, the global best of the swarm. Furthermore, the values of w, c_p and c_g are changed to enhance the flexibility of the algorithm.

After updating the velocity and position of the particle, the fitness of the particle is computed. If the particle's fitness is better than the stored best fitness of the particle, the particle stored best fitness is updated by the new fitness and the new position. Then, crossover and mutation are done according to their predefined probabilities *cross* and *mut* respectively.

The crossover is done if a randomly chosen number in the range [0 1] is less than *cross.* A randomly chosen particle is used to crossover with the current particle at a randomly chosen position. The new position is used as the new particle location. The new particle's fitness is computed and stored if it is better than the stored best fitness. This operation helps escaping the local optimum area. The mutation is done in a particle if a randomly chosen number in the range [0 1] is less than *mut*. Then the mutation is done by changing a randomly chosen component x_{id} of the particle position X_i. Unlike the mutation probability in genetic algorithm literature in which it is the probability of every component of the gene to have mutation, *mut* is the probability of the entire particle to have mutation in one of its component. The mutation is done according to the following relation:

$$x_{id} = -x_{id}\,(r_3+0.5) \quad (9)$$

r_3 is a random number. Then the new fitness is computed. This operation helps escaping local optimum area too. The probabilities of crossover and mutation are changed in every stage of the optimization to add more flexibility to the algorithm to achieve its goal. As shown in Fig. 1, these operations are repeated until the maximum number of iterations is reached. The clever choosing of the algorithm parameters specifies its effectiveness. At the start of the *local stage,* we set the positions of the swarm, X, to its known good positions, P, ($X=P$) to start the local search. In the following section, full detailed data are presented and the results prove the effectiveness of the algorithm to deal with many multimodal multi-dimensional problems. More details are declared in Appendix A.

If it is needed, in case of too complex optimization problems, the algorithm can be used to optimize its main parameters, such as *glob*, *loc* and *clas*. *glob* and *loc* determine the range of the 3 stages, and *clas* determines the number of maximum clusters. The optimization of these parameters will effectively enhance the performance of the algorithm for higher complex functions as will be declared in the next section.

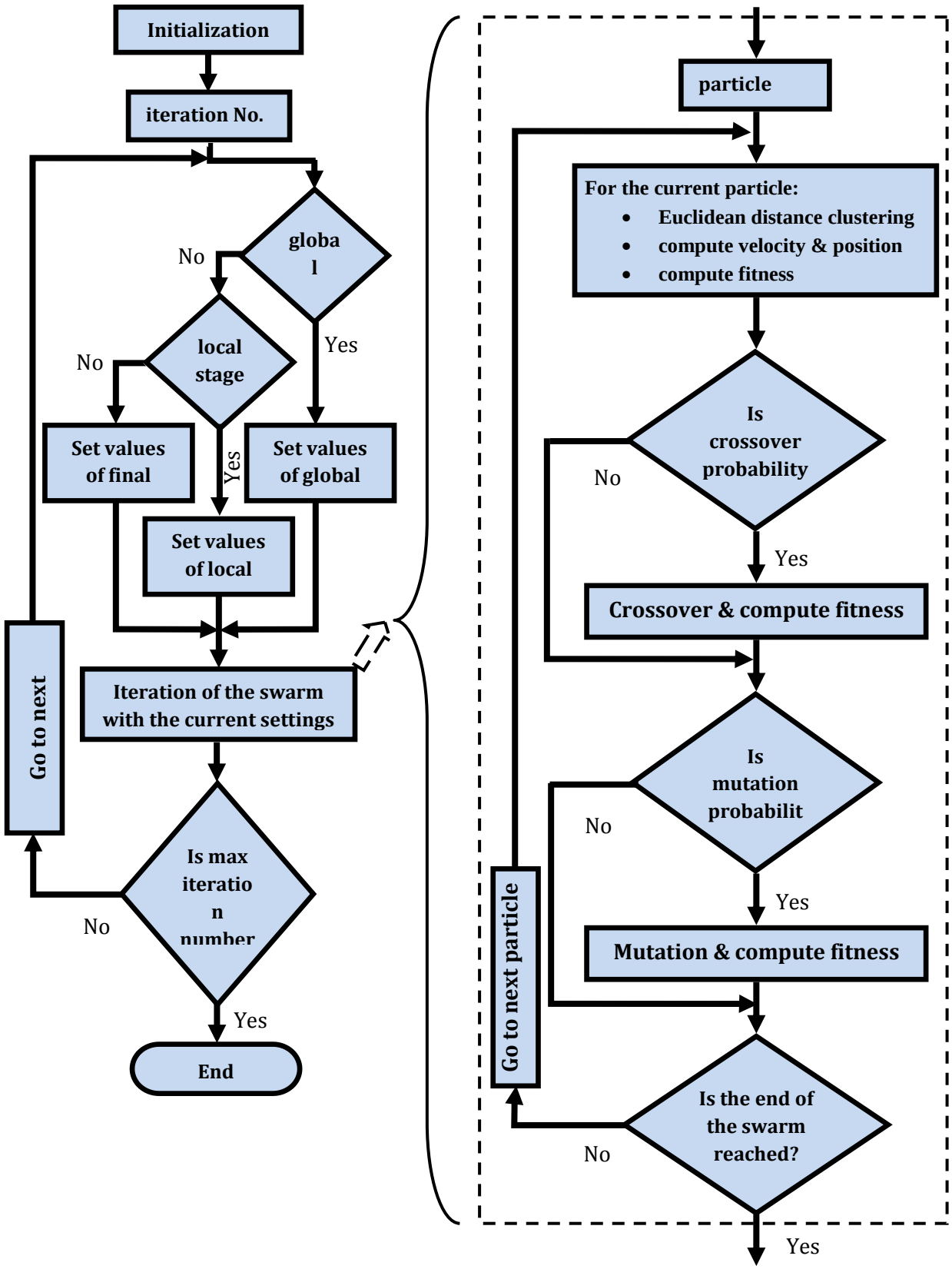

Fig. 1 The enhanced clustered PSO algorithm

The flowchart of the optimization algorithm is the same as the flowchart shown in Fig. 1. We call the two optimizations processes *outer optimization* and *inner optimization*. *Inner optimization* stands for the original optimization process of the multimodal functions and it will be optimized by the *outer optimization. Outer optimization* has the same flowchart but "compute fitness" term is done by running the *inner optimization* with the parameters given by the outer one for specific number of times and considering the fitness as how many times the *inner optimization* finds the global optimum. While the optimization of the ICO-PSO does

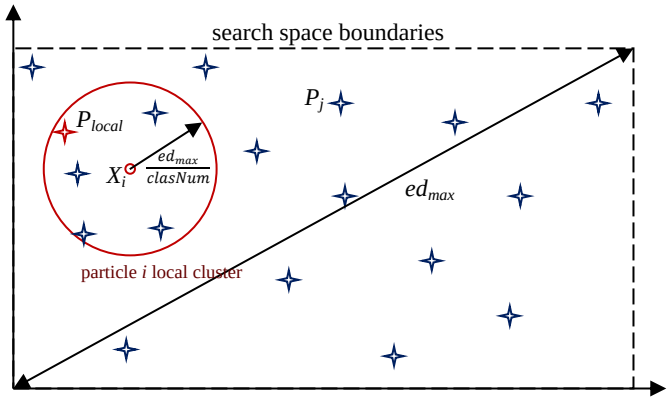

Fig. 2 The local cluster of the particle X_i containing some best positions P_j

not involve heavy computational work and it does not consume too much time, it is not a necessity as will be declared by the data given in the next section. General parameters sittings are appropriate for wide range of problems.

IV. SIMULATION AND RESULTS

ICO-PSO is designed to manipulate real world complex (multi-dimensional multimodal) problems via limited hardware resources. In other words, it increases the ability of finding global optimum in high dimensional optimization problems, especially in multimodal problems, without requiring huge population size or large number of iterations that necessitate high computation effort and large memory size. The results are discussed through three subsections. In subsection IV.A, we analyze, in details, the behavior of ICO-PSO through the three stages of the optimization process. Therefore, a 2-dimensional problem is utilized to simplify the detailed behavior analysis. In subsection IV.B, the effect of limited resources is studied. ICO-PSO and some other well-known algorithms are subjected to optimizations in which the dimensions are increased without enough afforded computations. The behavior of ICO-PSO dealing with these limitations is compared to other algorithms. In subsection IV.C, ICO-PSO is compared to other algorithms in the literature. The ability of ICO-PSO to achieve same results of other algorithms (or close enough) with limited computations is presented in this subsection.

A. Analysis of ICO-PSO Behavior Through the Optimization

In this subsection, we analyze the behavior of ICO-PSO compared to PSO through the optimization to check if it achieves the stages' goals. First, consider the data of the multimodal functions that are listed in Table 1. We choose multimodal functions that can be expanded to multi-dimensional forms. In this subsection, all functions' global minimums are at point (0, … 0) and the minimum values are zeros except F_4. Its minimum value is –1.

The optimization of the function F_1 (in 2-dimentional form) is done several times. In each time, we generate a random initial population and assign it to both PSO

Table 1 four multidimensional multimodal test functions data

Name	Equation	Range
Rastrigin's function	$F_1 = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5,5]^D$
Griewank's function	$F_2 = 1 + \sum_{i=1}^{D}(\frac{x_i^2}{4000}) - \prod_{i=1}^{D}(\cos(\frac{x_i}{\sqrt{i}}))$	$[-25,25]^D$
Ackley's function	$F_3 = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	$[-5,5]^D$
Drop-Wave function	$F_4 = -\frac{1+\cos(12\sqrt{\sum_{i=1}^{D}x_i^2})}{0.5(\sum_{i=1}^{D}x_i^2)+2}$	$[-5,5]^D$

and ICO-PSO. Fig. 3 declares one of the optimizations in which PSO failed to find the global optimum while ICO-PSO succeeded. The figure declares the best-found positions of the particles, P, plotted on top of the contour of F_1 at the end of the *global stage* (after 25 iterations). The contour circles in blue (dark colors at the center) represent local minima while the contour circles in green, yellow (light colors at the center) or red at the corners represent local maxima. The function has one global minimum at point (0, 0). PSO and ICO-PSO start with the same initial population, which is randomly distributed by a uniform distribution function. At the end of *global stage*, the best-found positions, *P*, of PSO is distributed on 5 local minima areas. Unfortunately, none of them is the area of the global minimum and finally the PSO is trapped at one of the local minima. While ICO-PSO best-found positions are distributed on 14 local minima areas including the global one. This is the primary goal of *global stage*, to explore more of the search space and locate more areas of interest. Therefore ICO-PSO is effectively exploring the search space 2.8 (14/5) times more than PSO in this example. This ability of exploring the entire search space, namely exploration, is attributed to the parameters' settings of the *global stage*. When the dimensions increase, the number of local minima increases and the ability of exploration becomes more vital.

At this moment, let's take another optimization of F_1 where PSO was more lucky to find the area of global minimum. Fig. 4 shows this optimization. At the end of the 25th iteration, both ICO-PSO and PSO found the area of global minimum, around point (0, 0). In the next iterations, PSO continues its routine according to (2), (3) and (4), but ICO-PSO starts the *local stage*. As declared after 50 iterations, PSO converged to a local minimum rather than the global minimum. The explanation is declared by the zoomed in figure at the

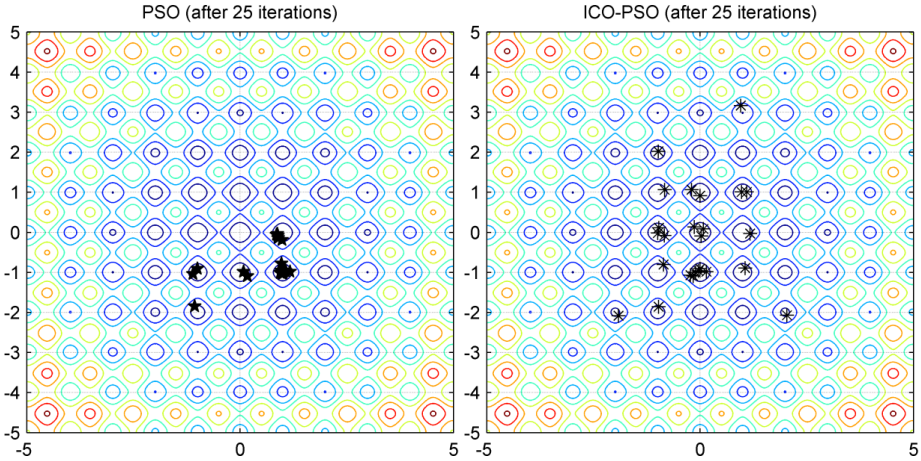

Fig. 3 graphical representation of best known positions, *P*, of PSO and ICO-PSO for the optimization of F_1 after *global stage*.

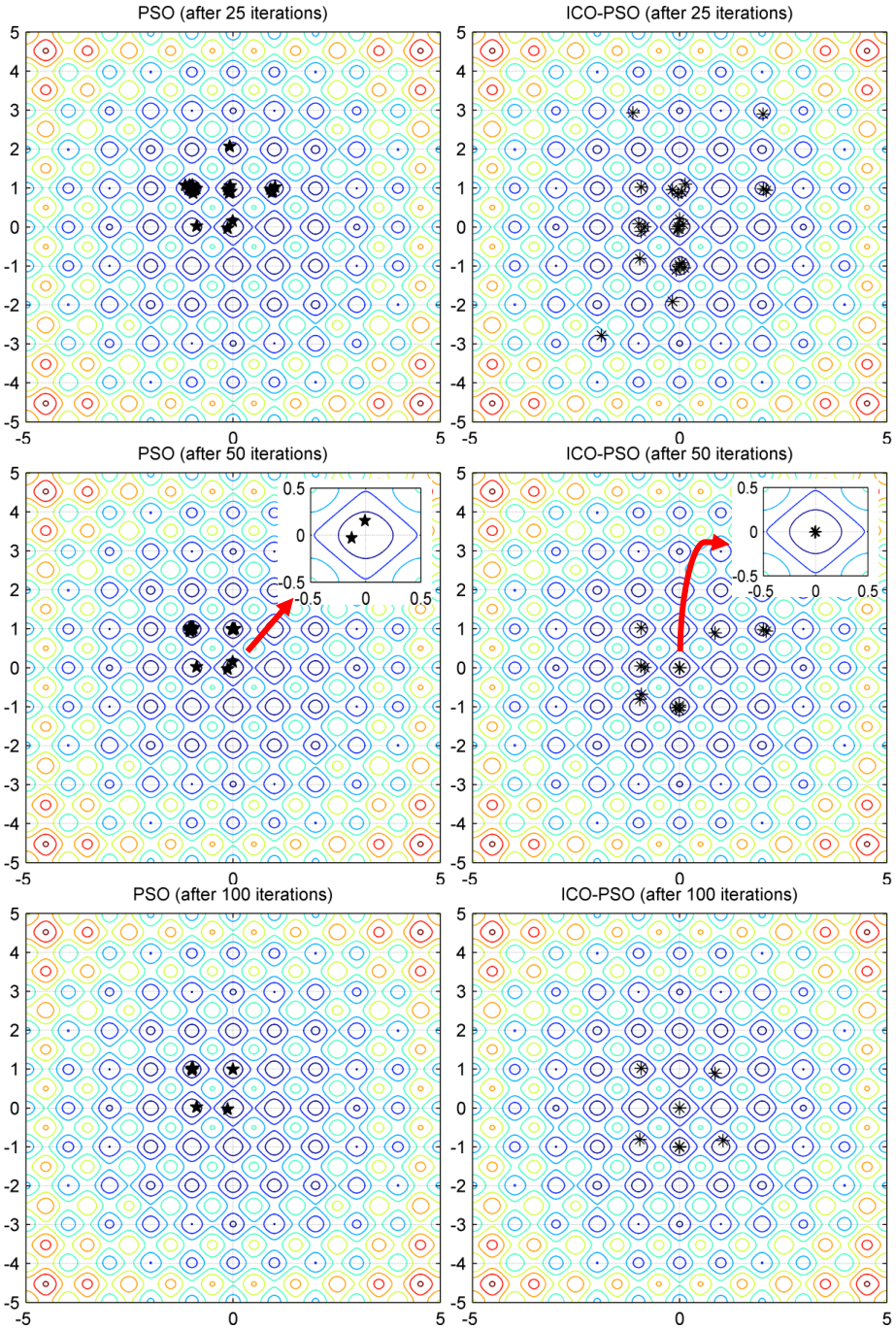

Fig. 4 Graphical representation of best-known positions, *P*, of PSO and ICO-PSO for the optimization of F_1 through the optimization stages (25 iterations: end of *global stage*, 50 iterations: end of *local stage*, 100 iterations end of *final stage*)

up right corner. The found best position at the global minimum area is not at the center that has the lowest minimum. So that, the value of the found best position at the centre of the local minimum is better than the value found in the global minimum area but not at the center. As a result, the global minimum of the swarm, that attracts all particles, is at a local minimum of the search space not at the global one. After 100 iterations, the PSO swarm converged to the local minimum, trapped, and was not able to find the global minimum. On the other hand, while ICO-PSO starts the *local search stage,* each particle slows down, searches the near area of its best position and its motion is affected mainly by its known best position then by its near neighborhood. As a result, after 50 iterations, at the end of *local search stage,* the swarm has found the best positions of many local minima including the global one. After the *final stage,* ICO-PSO found the global minimum, improved the search around it, and more particles (19 out of 25) are concentrated in the global minimum area.

In conclusion, because of PSO rapid convergence, it fails to find the global optimum in multimodal functions for 2 reasons; it has weak exploration of the search space and weak exploitation around the found good solutions except the best found one. As the dimensions increase, the complexity and number of local minima increase and these disadvantages become more apparent. ICO-PSO overcomes these disadvantages by its stage-structured algorithm and by its added features, namely; clustering, mutation, crossover and optimization.

B. The Effect of Limited Resources on ICO-PSO and Some Will-Known Algorithms

In This subsection, we discuss the effect of increasing the optimization dimensions with limited computing capabilities (small population size and low number of iterations) on the algorithm effectiveness. ICO-PSO is compared to classic PSO and genetic algorithm in addition to FGLT-PSO. In [6], FGLT-PSO was evaluated using several (20) unimodal and multimodal nonlinear benchmark functions and compared to several classic and improved PSO including QIPSO, LPSO, AFPSO and many others. FGLT-PSO outperforms other algorithms in terms of solution accuracy and convergence speed. Solution accuracy means FGLT-PSO successfully reaches the global optimum in multimodal functions. However, the optimizations are done with sufficient population size and number of iterations. In real world problems, as the dimensions of the search space increase, the optimization process become too expensive, or even intolerable, to be implemented for limited resources systems.

In order to evaluate ICO-PSO compared to other algorithms, at the start, the optimization is done in 2 dimensions. For a fair comparison, each optimization is done several times, 1000 times in this research. In each time, a new random initial population (positions and velocities) is generated and assigned to the algorithms. Each algorithm is evaluated by considering the number of times the algorithm reaches the global optimum, or considerably close to it, and not trapped in any local optimum. Note that the particle fitness is considered close enough to the global minimum when the fitness is much lower than the lowest local minimum. The values of acceptable fitness values of the functions listed in Table 1 are shown in Table 2.

The number of successes to find the global minimum is considered as the evaluation of the algorithms in 2 dimensions. Then the dimensions are increased and the whole process is repeated again for every dimension. However, we increase the dimensions without increasing the number of particles and the maximum number of iteration is chosen considerably low for each dimension. The chosen parameters of the algorithms are listed in Table 3. The parameters of ICO-PSO are chosen to improve the ability of each stage to achieve its goal. Fig. 5 represents these changes for N=100. The data listed in Table 3 are applied for all dimensions (2, 3, 4, 5 and 10).

Tables (4, 5 and 6) show the results of the optimization of functions F_1, F_2 and F_3. The optimization is done for 1000 time for each dimension and the tables list the number of times each algorithm reaches the global minimum or considerably close to it. It is clear that all algorithms are affected, but in different degree, by increasing the dimensions without increasing the population size and without sufficient iterations. In Table (4), at 10 dimensions, ICO-PSO finds the global minimum 522 times, while the nearest algorithm finds it only 13 times. The number between parentheses is the ratio of number of successes of ICO-PSO to the number of successes of other algorithms at the same dimensions. It is obvious that this ratio increases as the dimensions increase, which implies the ICO-PSO power in dealing with higher dimensional problems with minimal computing effort. Table (5) and Table (6) also show the superiority of ICO-PSO as the dimensions increase without sufficient number of iterations or population size. Note that the Ackley's function, F_3, is so easy to be optimized by population size of 25 particles. Therefore, we did the optimization with population size of 5 particles. It is the same reason for choosing range [-25, 25] for F_2 instead of [-5, 5]. Fig. 6 shows the functions in comparison to each other near the global minimum in one dimension. In F_3, there is a big difference in magnitude between the global minimum and the local minima, and in F_2, the area of global minimum is big. This is why we increase the challenge by decreasing the population size or increasing the range.

F_4 is more difficult with respect to other functions. As shown in Fig. 6, it is clear that in F_4 the local minima are very close, in distance and in magnitude,

Table 2 Acceptable accuracy of optimizations in subsection IV.A

Functions	F1	F2	F3	F4
Acceptable fitness	< 0.1	< 0.001	< 0.1	< –0.94

Table 3 Optimization algorithms' parameters

Algorithm	Parameter sittings[a]	
General settings	S=25 (for F_1,F_2,F_4) and S=5 for F_3	
PSO	w=0.9→0.4, c_p=c_g=0.5	
Genetic algorithm	crossover (function: Scattered, fraction = 0.8) mutation (function: Gaussian, mean = 0, variance = (0.5 × initial range)→0	
FGLT-PSO	w=0.9→0.4, C_1=0.5→2, C_2=1→2, C_3= 0.5→1.5	
ICO-PSO	General	*glob*=25, *loc*=50, *clas*=3,
	Global stage	w=0.9, c_p=0.7, c_g=0.3, *cross*=0.15, *mut*=0.15, *clasNum*=*clas*
	Local stage	w=0.3, c_p=0.6, c_g=0.4, *cross*=0.05, *mut*=0.05, *clasNum*=*clas*
	Final stage	w=0.9→0.4, c_p=0.5, c_g=0.5, *cross*=0.01, *mut*=0.01, *clasNum*=*clas*→1 (at half of the final stage then continues as 1).

[a]The arrow, →, means the parameter changes linearly from left value to right value during the optimization (or during the stage in ICO-PSO).

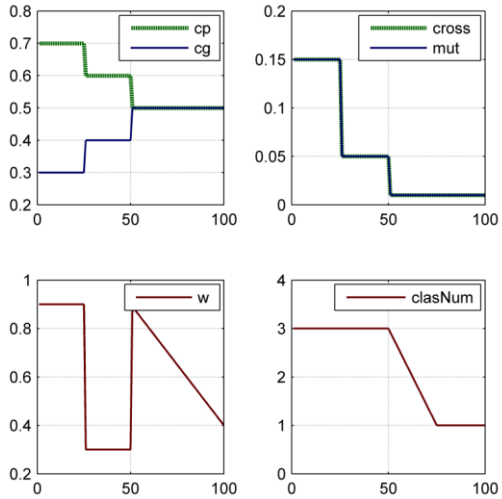

Fig. 5 ICO-PSO parameters' settings

Table 4 Algorithms' results (number of successes to reach global minimum out of 1000 try) for Rastrigin's function

	F_1 (Rastrigin's function) S=25 particles				
	D=2, *N*=100	*D*=3, *N*=200	*D*=4, *N*=400	*D*=5, *N*=500	*D*=10, *N*=1000
PSO	895 (1.12)[a]	356 (2.67)	96 (9.59)	17 (49.24)	0 (inf.)[b]
Genetic Algorithm	390 (2.56)	373 (2.54)	522 (1.76)	309 (2.71)	13 (40.15)
FGLT-PSO	913 (1.10)	529 (1.79)	193 (4.77)	43 (19.47)	0 (inf.)
ICO-PSO	**1000**	**948**	**921**	**837**	**522**

[a]() is the ratio of ICO-PSO to other algorithms at the same dimensions. [b]inf. refers to infinity (dividing by zero).

Table 5 Algorithms' results (number of successes to reach global minimum out of 1000 try) for Griewank's function

	F_2 (Griewank's function) S=25 particles				
	D=2, *N*=100	*D*=3, *N*=200	*D*=4, *N*=400	*D*=5, *N*=500	*D*=10, *N*=1000
PSO	223 (2.88)[a]	27 (8.22)	7 (16.71)	1 (49)	0 (inf.)[b]
Genetic Algorithm	57 (11.26)	12 (18.50)	15 (7.80)	6 (8.17)	0 (inf.)
FGLT-PSO	334 (1.92)	69 (3.22)	22 (5.32)	3 (16.33)	14 (2.93)
ICO-PSO	**642**	**222**	**117**	**49**	**41**

[a]() is the ratio of ICO-PSO to other algorithms at the same dimensions. [b]inf. refers to infinity (dividing by zero).

Table 6 Algorithms' results (number of successes to reach global minimum out of 1000 try) for Ackley's function

	F_2 (Ackley's function) S=5 particles				
	D=2, *N*=100	*D*=3, *N*=200	*D*=4, *N*=400	*D*=5, *N*=500	*D*=10, *N*=1000
PSO	**995** (0.997)[a]	916 (1.06)	748 (1.30)	521 (1.81)	8 (106)
Genetic Algorithm	50 (19.84)	16 (60.69)	25 (38.84)	7 (134.71)	0 (inf.)[b]
FGLT-PSO	975 (1.02)	823 (1.18)	602 (1.61)	361 (2.61)	17 (49.89)
ICO-PSO	992	**971**	**971**	**943**	**848**

[a]() is the ratio of ICO-PSO to other algorithms at the same dimensions. [b]inf. refers to infinity (dividing by zero).

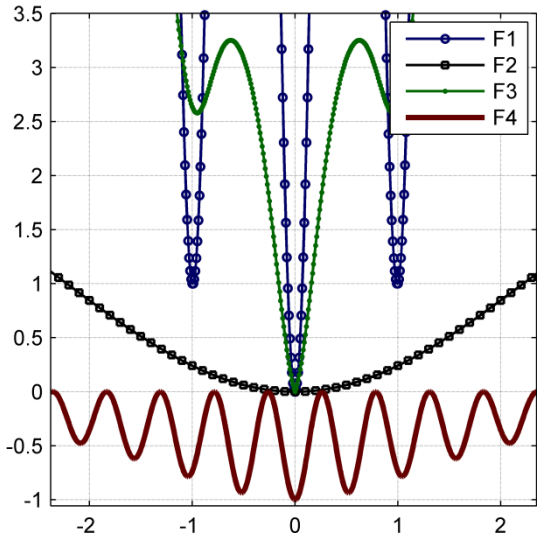

Fig. 6 Test functions behavior near the global minimum in 1-dimension

to the global minimum with respect to other functions. Although ICO-PSO performance is better than other algorithms, it can do better. ICO-PSO is used to

optimize its parameters as discussed in the last section. The optimization is done to optimize *glob* and *loc*, which determine the ranges of the stages, in addition to the maximum number of clusters, *clas.* The *outer optimization's* parameters are set as listed in Table (3) with population size of 25 and maximum number of iterations of 100. The optimized ICO-PSO results are showed in Table (7). At 5 dimensions, the number of successes of ICO-PSO was 2 and it is raised to 60 after optimization. This is an improvement ratio of 30 times. This ratio can be increased by adding more ICO-PSO parameters, such as the stages parameters, to the optimization.

In the end, the summary of these results is declared by Fig. 7. The average of the successes of the algorithms (for the four functions) at 2, 3, 4, 5 and 10 dimensions are showed in log scale. The number of successes of the algorithms at 10 dimensions for F_4 is considered zero for all algorithms. The figure shows how the algorithms are clearly affected by increasing the dimensions without enough computation effort. The superiority of ICO-PSO to overcome this difficulty with respect to other algorithms is obvious. At 10 dimensions, ICO-PSO, in average, finds the global optimum 352.75 times while the nearest one, FGLT-PSO, average is 7.75 times. This gap increases by increasing the dimensions.

C. ICO-PSO Compared to Other Algorithms in The Literature

In order to evaluate the contribution to computational effort saving of ICO-PSO algorithm compared to other algorithms, let's consider other researches that represent results of well-known algorithms. In this subsection, we consider the results given in other researches and investigate the ability of ICO-PSO to achieve the same results, or considerably close to it, with reduced computation effort. We implemented FGLT-PSO [6] in the last subsection and compared it to ICO-PSO at dimensions 2, 3, 4, 5, and 10. Let's compare its results and other algorithms results obtained in [6] at higher dimensions (30 and 50) with the ICO-PSO. We choose a variety of test functions; unimodal, multimodal, shifted and rotated, to verify the performance of ICO-PSO at different situations (F_3 Table 1 and F_5-F_{11} in Table 8). The test functions properties are listed in Table 8. F_5 and F_6 are unimodal while other functions are multimodal. The average results (of 30 run) are listed in Table 9, 10 and 11. Note that the Ackley's function (F_3) used in this experiment is the same given in Table 1 but the range is $[-32, 32]^D$. in Table 11, more algorithms are tested at 30 dimensions.

Tables 9, 10 and 11 list the average best solutions, standard deviations and median best solutions of the 30 run of the algorithms according to [6], in addition to population size and maximum number of iterations. Table 9 is 30 dimensions test while Table 10 is 50 dimensions test. Table 11 introduces extra algorithms and extra functions at 30 dimensions. The ICO-PSO is tested for these functions but with reduced population size and maximum number of iterations. Although ICO-PSO results has the first rank in 4 functions (F_7, F_8, F_9, and F_{11}) and the second rank in 4 functions (F_5, F_6, F_3, and F_{10}), it is not the purpose of this subsection to achieve best results. The purpose is to find out how much ICO-PSO can achieve reasonable results with respect to other algorithms with reduced computation effort. The reduction percentage of population size varies between 20% and 83.3% while the reduction percentage of maximum number of iterations varies between 30% and 90%. For example, as shown in results of F_{11} (Table 11), ICO-PSO has rank number one of the 9 algorithms although it utilizes population size of 5 particles (instead of 30) and maximum number of iterations of 1000 iterations (instead of 10000). The reduced computation effort and memory size are significant. While ICO-PSO performs well in unimodal functions, its big contribution is in the multimodal functions. All these functions utilize the same ICO-PSO parameters introduced in Table 3 without the need of parameter optimization.

Table 7 Algorithms' results (number of successes to reach global minimum out of 1000 try) for Drop-Wave function

	F_4 (Drop-Wave function) S=25 particles			
	D=2, N=100	D=3, N=200	D=4, N=400	D=5, N=500
PSO	874 (1.14)[a]	247 (2.77)	42 (6.67)	2 (30)
Genetic Algorithm	180 (5.53)	22 (31.14)	3 (93.33)	1 (60)
FGLT-PSO	799 (1.25)	171 (4.01)	20 (14)	0 (inf.)[b]
ICO-PSO	**995**	**685**	**280**	**60**

[a]() is the ratio of ICO-PSO to other algorithms at the same dimensions. [b]inf. refers to infinity (dividing by zero).

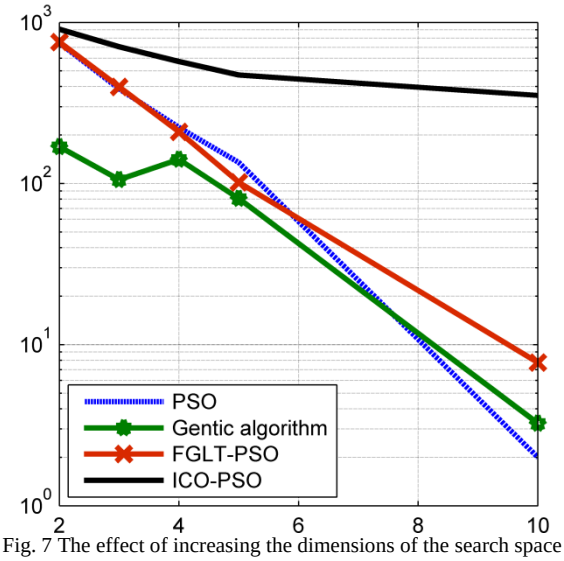

Fig. 7 The effect of increasing the dimensions of the search space without sufficient computation effort

Table 8 Extra test functions used in the experiments of subsection IV.C according to [6].

Name	Equation	Range	x^*	$f(x^*)$
Sphere Function (unimodal)	$F_5 = \sum_{i=1}^{D} x_i^2$	$[-100,100]^D$	$(0,0)^D$	0
Schwefel's Problem 1.2 (unimodal)	$F_6 = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	$[-100,100]^D$	$(0,0)^D$	0
Non-continuous Rastrigin's Function	$F_7 = \sum_{i=1}^{D} (y_i^2 - 10\cos(2\pi y_i) + 10)$ where $y_i = \begin{cases} x_i & \lvert x_i\rvert \le 0.5 \\ \frac{round(2x_i)}{2} & \lvert x_i\rvert \ge 0.5 \end{cases}$	$[-5.12,5.12]^D$	$(0,0)^D$	0
Shifted Generalized Griewank's Function	$F_8 = 1 + \sum_{i=1}^{D} (\frac{z_i^2}{4000}) - \prod_{i=1}^{D} (\cos(\frac{z_i}{\sqrt{i}})) + fbias_8$ where $fbias_8$=-180, z=x-o	$[-600,600]^D$	$(0,0)^D$	–180
Shifted Rosenbrock's Function	$F_9 = \sum_{i=1}^{D-1} [100(z_i^2 - z_{i+1}^2)^2 + (z_i - 1)^2] + fbias_9$ where $fbias_9$=390, z=x-o+1	$[-100,100]^D$	$(1,1)^D$	390
Rotated Salomon's Function	$F_{10} = 1 - \cos\left(2\pi \sqrt{\sum_{i=1}^{D} y_i^2}\right) + 0.1 \sqrt{\sum_{i=1}^{D} y_i^2}$ where $y = M \times x$, M rotation matrix [29].	$[-100,100]^D$	$(0,0)^D$	0
Rotated Rastrigin's Function	$F_{11} = \sum_{i=1}^{D} (y_i^2 - 10\cos(2\pi y_i) + 10)$ where y=$M \times x$, M rotation matrix [29].	$[-5.12,5.12]^D$	$(0,0)^D$	0

$f(x^*)$ denotes the optimum value of the function at the optimum position x^*.

Another point of interest is the convergence rate. FGLT-ICO showed very fast convergence rate at many functions such as Shifted Rastrigin's Function at 30 dimensions. It is the same as F_1 but shifted and a bias of -330 is added so that the best fitness is -330. We assign the same reduced population size (15) and maximum number of iterations (5000) to FGLT-PSO, PSO and ICO-PSO. The average best solution is shown in Fig. 8. While FGLT-PSO and PSO are faster, they are trapped far away from the global minimum. ICO-PSO is slightly slower because of the *global search stage* but it provides the algorithm more capability to find global minimum with the limited computations. The fast convergence loses its value when the algorithm is trapped far away from the global optimum. Therefore, ICO-PSO is more capable of optimizing multimodal function with limited resources.

Another research example is the paper introduces CLPSO [21]. CLPSO is compared to 8 algorithms through 16 function. These algorithms are PSO with inertia weight (PSO-w) [23], PSO with constriction factor (PSO-cf) [24], Local version of PSO with inertia weight (PSO-w-local), Local version of PSO with constriction factor (PSO-cf-local) [25], Unified Particle Swarm Optimizer (UPSO) [26], FIPS [19], Fitness-Distance-Ratio-based PSO (FDR-PSO) [27] and Cooperative Particle Swarm Optimizer (CPSO-H) [28]. A sample of these functions, listed in Table (12), is used to test ICO-PSO. In these functions (F_{12}-F_{15}), the initial population is not distributed over all the search space. There are parts of the search space, out of the initial swarm distribution, that need to be explored. Therefore, there are two parameter changes of the ICO-PSO algorithm are made. The initial velocity should be larger than usual, and the global and local search need more time to accomplish its missions. Appendix A declares these changes. The results are listed in Table 13. ICO-PSO results have ranks 1 and 2 while it utilizes reduction in population size and maximum fitness evaluations (FEs) reaches 85%. For example, ICO-PSO achieves rank number 1 for the function F_{15} while it utilizes 75% and 85% reduction in population size and maximum fitness evaluations respectively. The superiority of manipulating the limited computation resources is obvious. Detailed notes about ICO-PSO algorithm are listed in Appendix A.

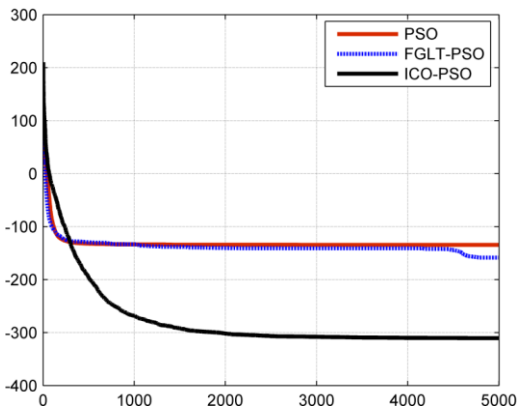

Fig. 8 convergence rate for F_9 with the reduced population size and number of iterations (optimum -330).

Table 9 ICO-PSO results compared to 4 algorithms results including FGLT-PSO introduced in [6] (at 30 dimensions).

F	Algorithm	Average best solution	Standard deviation	Median best solution	Population size	Max. iteration number	Reduction percentage of population size and iterations
F_5	**PSO**	5.397e –69	1.161e –68	2.321e –70	50	10000	30%, 40% (Results rank 2)
	LPSO	2.747e –28	4.078e –28	1.047e –28	50	10000	
	QIPSO	3.502e –69	1.664e –68	2.727e –72	50	10000	
	FGLT–PSO	**0.00e +00**	**0.00e +00**	**0.00e +00**	50	10000	
	ICO–PSO	6.592e –79	3.603e –78	7.993e –86	**35**	**6000**	
F_6	**PSO**	1.189e +04	6.605e +03	1.083e +04	50	10000	20%, 30% (Results rank 2)
	LPSO	1.399e +02	9.380e +01	1.254e +02	50	10000	
	QIPSO	5.733e +03	5.484e +03	5.000e +03	50	10000	
	FGLT–PSO	**5.959e –20**	**1.762e –19**	**5.483e –21**	50	10000	
	ICO–PSO	2.842e –10	8.532e –10	4.282e –11	**40**	**7000**	
F_3	**PSO**	1.013e –14	3.312e –15	7.994e –15	50	10000	60%, 50% (Results rank 2)
	LPSO	2.931e –14	1.433e –14	2.576e –14	50	10000	
	QIPSO	**8.941e –15**	**2.457e –15**	**7.994e –15**	50	10000	
	FGLT–PSO	7.771e –02	2.015e –01	7.994e –15	50	10000	
	ICO–PSO	6.057e –14	9.624e –15	5.770e –14	**20**	**5000**	
F_7	**PSO**	5.827e +01	3.080e +01	5.450e +01	50	10000	80%, 75% (Results rank 1)
	LPSO	4.281e +01	2.064e +01	3.753e +01	50	10000	
	QIPSO	4.310e +01	2.901e +01	3.500e +01	50	10000	
	FGLT–PSO	2.136e +01	8.612e +00	2.212e +01	50	10000	
	ICO–PSO	**1.313e +01**	**6.350e +00**	**1.200e +01**	**10**	**2500**	
F_8	**PSO**	–8.340e +01	4.560e +01	–9.269e +01	50	10000	70%, 75% (Results rank 1)
	LPSO	–1.705e +02	4.344e +00	–1.704e +02	50	10000	
	QIPSO	–9.806e +01	3.839e +01	–1.053e +02	50	10000	
	FGLT–PSO	–1.974e +02	1.112e +00	–1.798e +02	50	10000	
	ICO–PSO	**–1.799e +02**	**1.921e –01**	**–1.799e +02**	**15**	**2500**	

Table 10 ICO–PSO results compared to 4 algorithms results including FGLT–PSO introduced in [6] (at 50 dimensions).

F	Algorithm	Average best solution	Standard deviation	Median best solution	Population size	Max. iteration number	Reduction percentage of population size and iterations
F_5	**PSO**	7.586e –49	2.124e –48	4.542e –50	50	15000	30%, 53.3% (Results rank 2)
	LPSO	7.297e –20	7.695e –20	4.843e –20	50	15000	
	QIPSO	2.446e –49	8.478e –49	1.395e –50	50	15000	
	FGLT–PSO	**5.239e –232**	**0.00e +00**	**2.541e –251**	50	15000	
	ICO–PSO	1.055e –52	4.85e –52	5.623e –57	**35**	**7000**	
F_6	**PSO**	4.167e +04	1.404e +04	4.002e +04	50	15000	20%, 30% (Results rank 2)
	LPSO	2.775e +04	8.532e +03	2.859e +04	50	15000	
	QIPSO	3.585e +04	1.967e +04	3.434e +04	50	15000	
	FGLT–PSO	**1.098e –08**	**2.405e –08**	**5.928e –09**	50	15000	
	ICO–PSO	2.134e –04	2.922e –04	9.954e –05	**40**	**10000**	
F_3	**PSO**	2.049e +00	4.661e +00	2.220e–14	50	15000	50%, 60% (Results rank 2)
	LPSO	1.145e –05	6.223e –05	1.782e –09	50	15000	
	QIPSO	**1.782e –14**	**3.695e –15**	**1.510e –14**	50	15000	
	FGLT–PSO	1.299e +00	5.560e –01	1.282e +00	50	15000	
	ICO–PSO	3.681e –11	1.284e –10	1.127e –12	**25**	**5000**	
F_7	**PSO**	1.980e +02	4.888e +01	1.890e +02	50	15000	70%, 83.3% (Results rank 1)
	LPSO	1.872e +02	3.958e +01	1.850e +02	50	15000	
	QIPSO	1.546e +02	5.192e +01	1.475e +02	50	15000	
	FGLT–PSO	4.759e +01	2.294e +01	4.636e +01	50	15000	
	ICO–PSO	**3.496e +01**	**1.036e +01**	**3.350e +01**	**15**	**2500**	
F_8	**PSO**	8.313e +01	9.121e +01	7.911e +01	50	15000	70%, 73.3% (Results rank 1)
	LPSO	–1.551e +02	1.216e +01	–1.565e +02	50	15000	
	QIPSO	6.894e +01	7.896e +01	7.481e +01	50	15000	
	FGLT–PSO	–1.791e +02	1.212e +00	–1.797e +02	50	15000	
	ICO–PSO	**–1.797e +02**	**5.821e –01**	**–1.799e +02**	**15**	**4000**	

V. CONCLUSION

The optimization process is becoming more vital in many aspects of modern industries, sciences and life. Many portable devices (such as smart phones, multi–copters, GPS devices, ... etc) in addition to IoT devices faces the challenge of being smaller while increasing its ability to deal with much more complex real world

data. This challenge necessitates algorithms that provide acceptable performance with minimal computation efforts that consumes the limited resources. Optimization is one of the processes that may be needed in many devices (e.g. path planning optimization in self-driving cars, multi-copters controllers, mobile robots ... etc). ICO-PSO is improved algorithms that overcome the difficulty of global optimization of multimodal problems with very low computational effort. ICO-PSO is tested by fifteen benchmark multi-dimensional test functions and the results proved its superiority with respect to many other algorithms in case of low available computation capabilities.

Table 11 ICO-PSO results compared to results of other algorithms including FGLT-PSO introduced in [6] (at 30 dimensions).

F	Algorithm	Average best solution	Standard deviation	Population size	Max. iteration number	Reduction percentage of population size and iterations
F_9	**PSO**	3.217e +09	3.880e +09	30	10000	33.3%, 50% (Results rank 1)
	QIPSO	2.347e +09	1.872e +09	30	10000	
	FIPS	1.340e +03	2.044e +03	30	10000	
	DMS–PSO	3.362e +08	3.089e +08	30	10000	
	CLPSO	5.943e +02	5.069e +01	30	10000	
	AFPSO	9.700e +07	1.197e +08	30	10000	
	AFPSO–QI	8.832e +07	9.793e +07	30	10000	
	FGLT–PSO	5.204e +02	1.328e +02	30	10000	
	ICO–PSO	**4.439e +02**	**4.050e +01**	**20**	**5000**	
F_{10}	**PSO**	1.703e +01	2.554e +00	30	10000	50%, 60% (Results rank 2)
	QIPSO	1.520e +01	1.319e +00	30	10000	
	FIPS	2.660e +01	1.417e +00	30	10000	
	DMS–PSO	1.292e +01	1.328e +00	30	10000	
	CLPSO	1.194e +01	1.356e +00	30	10000	
	AFPSO	1.038e +01	1.379e +00	30	10000	
	AFPSO–QI	8.462e +00	9.477e –01	30	10000	
	FGLT–PSO	**5.899e –01**	**2.280e –01**	30	10000	
	ICO–PSO	1.397e +00	2.356e –01	**15**	**4000**	
F_{11}	**PSO**	3.202e +02	1.470e +01	30	10000	83.3%, 90% (Results rank 1)
	QIPSO	3.175e +02	2.324e +01	30	10000	
	FIPS	4.341e +02	3.499e +01	30	10000	
	DMS–PSO	2.837e +02	1.606e +01	30	10000	
	CLPSO	2.633e +02	1.196e +01	30	10000	
	AFPSO	2.663e +02	1.200e +01	30	10000	
	AFPSO–QI	2.533e +02	1.263e +01	30	10000	
	FGLT–PSO	1.580e +02	**1.065e +01**	30	10000	
	ICO–PSO	**1.415e +02**	4.566e +02	**5**	**1000**	

Table 12 Extra test functions according to [21].

Name	Equation	Search rang	Initial range	x^*	$F(x^*)$
Weierstrass Function	$f_{12} = \sum_{i=1}^{D}\left(\sum_{k=0}^{kmax} a^k \cos(2\pi b^k (x_i + 0.5))\right) - D\sum_{k=0}^{kmax} a^k \cos(2\pi b^k \times 0.5)$ Where a=0.5, b=3, $kmax$=20.	$[-0.5, 0.5]^D$	$[-0.5, 0.2]^D$	$(0,0)^D$	0
Rotated noncontinuous Rastrigin's Function	$F_{13} = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10)$ where $z_i = \begin{cases} y_i & \lvert y_i\rvert \le 0.5 \\ \frac{round(2y_i)}{2} & \lvert y_i\rvert \ge 0.5 \end{cases}$, $y = M \times x$, M is rotation matrix [29]	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$	$(0,0)^D$	0
Rotated Weierstrass Function	$f_{14} = \sum_{i=1}^{D}\left(\sum_{k=0}^{kmax} a^k \cos(2\pi b^k (y_i + 0.5))\right) - D\sum_{k=0}^{kmax} a^k \cos(2\pi b^k \times 0.5)$ Where a=0.5, b=3, $kmax$=20, $y = M \times x$, M is rotation matrix [29]	$[-0.5, 0.5]^D$	$[-0.5, 0.2]^D$	$(0,0)^D$	0
Rotated Ackley's function	$F_{15} = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi y_i)\right) + 20 + e$ where $y = M \times x$, M is rotation matrix [29]	$[-32.738, 32.768]^D$	$[-32.768, 16]^D$	$(0,0)^D$	0

Table 13 ICO-PSO results compared to 8 algorithms results including CLPSO introduced in [21] (at 30 dimensions).

Algorithm	Mean ± std	Pop. size	Max FEs	reduced pop. size and FEs (%)	Mean ± std	Pop. size	Max FEs	reduced pop. size and FEs (%)
		F_{12}				F_{13}		
PSO–w	1.30e–04±3.30e–04	40	200000		6.32e+01±1.79e+01	40	200000	
PSO–cf	4.10e+00±2.20e+00	40	200000		7.88e+01±1.88e+01	40	200000	
PSO–w–local	4.94e–03±1.40e–02	40	200000		5.67e+01±1.36e+01	40	200000	
PSO–cf–local	1.16e–01±2.79e–01	40	200000	75%,	4.93e+01±1.11e+01	40	200000	50%,
UPSO	9.60e+00±3.78e+00	40	200000	64.6%	7.74e+01±1.40e+01	40	200000	17.5%
FDR	7.49e–03±1.14e–02	40	200000	(Results	4.36e+01±8.96e+01	40	200000	(Results
FIPS	1.54e–01±1.48e–01	40	200000	rank 1)	7.58e+01±1.92e+01	40	200000	rank 2)
CPSO–H	7.82e–15±8.50e–15	40	200000		8.80e+01±2.59e+01	40	200000	
CLPSO	3.45e–07±1.94e–07	40	200000		**3.77e+01±5.56e+00**	40	200000	
ICO–PSO	**3.79e–15±6.39e–15**	**10**	**70900**		4.30e+01±8.67e00	**20**	**165000**	
		F_{14}				F_{15}		
PSO–w	7.00e+00±1.98e+00	40	200000		1.71e+00±4.38e–01	40	200000	
PSO–cf	8.48e+00±2.54e+00	40	200000		1.66e+00±1.10e+00	40	200000	
PSO–w–local	5.96e+00±2.09e+00	40	200000		5.70e–01±7.60e–01	40	200000	
PSO–cf–local	5.95e+00±2.95e+00	40	200000	62.5%,	1.78e–01±5.62e–01	40	200000	75%,
UPSO	1.85e+01±3.37e+00	40	200000	64.6%	2.94e–01±6.71e–01	40	200000	85%
FDR	2.50e+00±1.46e+00	40	200000	(Results	3.59e–01±5.93e–01	40	200000	(Results
FIPS	**9.52e–02±9.53e–02**	40	200000	rank 2)	5.23e–07±**1.24e–07**	40	200000	rank 1)
CPSO–H	1.43e+01±3.53e+00	40	200000		2.10e+00±3.84e–01	40	200000	
CLPSO	3.07e+00±1.61e+00	40	200000		3.43e–04±1.91e–04	40	200000	
ICO–PSO	1.93e+00±1.78e+00	**15**	**70900**		**1.03e–07**±4.02e–07	**10**	**30100**	

REFERENCES

[1] S. U. Khan, S. Yang, L. Wang, and L. Liu, "A Modified Particle Swarm Optimization Algorithm for Global Optimizations of Inverse Problems", IEEE Transaction on Magnetics, vol. 52(3), 2016.

[2] M. A. Karamuftuoglu, S. Demirhan, Y. Komura, M. E. Çelik, M. Tanaka, A. Bozbey, and A. Fujimaki, "Development of an Optimizer for Vortex Transitional Memory Using Particle Swarm Optimization", IEEE Transactions on Applied Superconductivity, vol. 26(8), 2016.

[3] Y.-J. Chen, Q. Zhang, Y. Luo, and Y.-A. Chen, "Measurement Matrix Optimization for ISAR Sparse Imaging Based on Genetic Algorithm", IEEE Geoscience and Remote Sensing Letters, vol. 13(12), pp. 1875-1879, 2016.

[4] J. Rayno, M. F. Iskander, and M. H. Kobayashi, "Hybrid Genetic Programming With Accelerating Genetic Algorithm Optimizer for 3-D Metamaterial Design", IEEE Antennas and Wireless Propagation Letters, vol. 15, pp. 1743-1746, 2016.

[5] Y. Li, Z.-H. Zhan, S. Lin, J. Zhang, and X. Luo, "Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems", Information Science, vol. 293, pp. 370-382, 2015.

[6] Z. Beheshti, S. M. Shamsuddin, and S. Sulaiman, "Fusion Global-Local-Topology Particle Swarm Optimization for Global Optimization Problems", Mathematical Problems in Engineering, vol. 2014, 2014.

[7] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems", IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, vol. 42(3), pp. 627-645, 2012.

[8] Y.-J. Gong, J.-J. Li, Y. Zhou, Y. Li, H. S.-H. Chung, Y.-H. Shi, and J. Zhang, "Genetic Learning Particle Swarm Optimization", IEEE Transactions on Cybernetics, vol. 46(10), pp. 2277-2290, 2016.

[9] S. Li, M. Tan, I. W. Tsang, and J. T.-Y. Kwok, "A Hybrid PSO-BFGS Strategy for Global Optimization of Multimodal Functions", IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, vol. 41(4), pp. 1003-1014, 2011.

[10] K. Premalatha, and A. M. Natarajan, "Hybrid PSO and GA for Global Maximization", International Journal of Open Problems in Computer Science and Mathematics, vol. 2(4), pp. 597-608, 2009.

[11] N. Abdelkarim, A. E. Mohamed, A. M. El-Garhy, and H. T. Dorrah, "A New Hybrid BFOA-PSO Optimization Technique for Decoupling and Robust Control of Two-Coupled Distillation Column Process", Computational Intelligence and Neuroscience, vol. 2016, (2016).

[12] J. Kennedy, and R. C. Eberhart, "Particle swarm optimization", In: Proceedings IEEE Int. Conf. on Neural Networks (ICNN), Perth, Australia, vol. IV, pp. 1942–1948, 1995.

[13] R. C. Eberhart, Y. Shi, and J. Kennedy, "Swarm intelligence (The Morgan Kaufmann series in evolutionary computation)", 1st ed. San Francisco: Morgan Kaufmann, 2001.

[14] S. Gheitanchi, F. Ali, and E. Stipidis, "Particle Swarm Optimization for Adaptive Resource Allocation in Communication Networks", EURASIP Journal on Wireless Communications and Networking, vol. 2010:465632, 2010.

[15] N. Chauhan, A. Mittal, D. Wagner, M. V. Kartikeyan, and M. K. Thumm, "Design and Optimization of Nonlinear Tapers using Particle Swarm Optimization", International journal of Infrared and Millimeter Waves, vol. 29(8), pp. 792-798, 2008.

[16] H. T. Dorrah, A. M. El-Garhy, and M. E. El-Shimy, "PSO-BELBIC scheme for two-coupled distillation column", Journal of Advanced Research, vol. 2(1), pp. 73-83, 2011.

[17] J. Kennedy and R. Mendes, "Population structure and particle swarm performance", in Proceedings of IEEE international conference on Evolutionary Computation, vol. 2, pp. 1671–1676, 2002.

[18] M. Pant, T. Radha, and V. P. Singh, "A new particle swarm optimization with quadratic interpolation", in Proceedings of The International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007), vol. 1, pp. 55–60, 2007.

[19] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better", IEEE Transactions on Evolutionary Computation, vol. 8(3), pp. 204–210, 2004.

[20] J. J. Liang, and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer", in Proceedings of the IEEE Swarm Intelligence Symposium (SIS '05), pp. 127–132, 2005.

[21] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions", IEEE Transactions on Evolutionary Computation, vol. 10(3), pp. 281–295, 2006.

[22] Y.-T. Juang, S.-L. Tung, and H.-C. Chiu, "Adaptive fuzzy particle swarm optimization for global optimization of multimodal functions", Information Sciences, vol. 181(20), pp. 4539–4549, 2011.

[23] Y. Shi, and R. C. Eberhart, "A modified particle swarm optimizer", in Proceedings of The IEEE International Conference on Evolutionary Computation, pp. 69–73, 1998.

[24] M. Clerc, and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space", IEEE Transactions on Evolutionary Computation, vol. 6(1), pp. 58–73, 2002.

[25] J. Kennedy, and R. Mendes, "Population structure and particle swarm performance", in Proceedings of The IEEE Congress on Evolutionary Computation, Honolulu, HI, pp. 1671–1676, 2002.

[26] K. E. Parsopoulos, and M. N. Vrahatis, "UPSO: A unified particle swarm optimization scheme", in Lecture Series on Computational Sciences, pp. 868–873, 2004.

[27] T. Peram, K. Veeramachaneni, and C. K. Mohan, "Fitness-distance-ratio based particle swarm optimization", in Proceedings of the IEEE Swarm Intelligence Symposium, pp. 174–181, 2003.

[28] F. van den Bergh, and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization", IEEE Transactions on Evolutionary Computation, vol. 8, pp. 225–239, 2004.

[29] R. Salomon, "Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions", BioSysems, vol. 39(3), pp. 263–278, 1996.

APPENDIX A: TECHNICAL NOTES

This appendix devoted to declare the details of the ICO-PSO that will help the reader to fully understand the algorithm and reconstruct it if needed. The appendix is constructed in the form of separate notes.

- The velocity of the particles is crucial because of the mutation and crossover operations. Crossover and mutation may cause sudden change in particle position, which leads to sudden increase of the particle velocity in the next iteration according to (7). Therefore, the particle velocity should be treated wisely. Regarding this problem, the following 2 steps are added to the algorithm:
  - The particles' velocities should be bounded. We choose velocity limits that are proportional to the search space limits. These limits are defined according to the following equation:

$$\max|v_i| = \frac{\mathrm{Max}_{x_i} - \mathrm{Min}_{x_i}}{4} \qquad i = 1, 2, ..., D \tag{4}$$

    where Max_x_i and Min_x_i are the limits of the search space at dimension i. The maximum velocity v_i (positive or negative) in the i^{th} dimension are bounded to quarter (1/4) of the search range at this dimension.
  - Usually, when the calculated particle position exceeds the search space limits (Max_x_i or Min_x_i) at the i^{th} dimension, its position is constrained to the limits, not allowed to be beyond it. In ICO-PSO, another step is added as follows:

if x_i > Max_x_i Then:
x_i = Max_x_i
v_i = –0.9 v_i
end

if x_i < Min_x_i Then:
x_i = Min_x_i
v_i = –0.9 v_i
end

Inverting the velocity direction and reducing its value slightly enhances the performance of the algorithm. Without this step, if the velocity was big, the particle position well be stuck at the limits many iterations until the velocity is inverted by the regular movement (7). Reducing the velocity is done to dampen down the oscillations that may occur to the particles near the boundaries. This process imitates a ball hitting a wall; it inverts its direction and loses some energy.

- Two cases of the initial position of the swarm are tested in the paper; when the initial swarm position distributed in the entire search space and when it is limited to a part of it. In the first case (F_1-F_{11}), no need for large velocities, even if the search space is large, moderate velocities are adequate to the swarm to start the global search. While in the other case (F_{12}-F_{15}), when it is not distributed in the entire search space, larger velocities are needed to allow the global search to explore the uncovered parts of the search space. These velocities are chosen as follows:
    - In case 1 (F_1-F_{11}): v_i in the initial population is uniformly random distributed on the interval [–2, 2].
    - In case 2 (F_{12}-F_{15}): v_i in the initial population is uniformly random distributed on the interval [– (Max_x_i– Min_x_i)/6, (Max_x_i–Min_x_i)/6] where Max_x_i and Min_x_i are the limits of the search space at dimension i.
- While functions (F_1-F_{11}) needs no change of parameters sittings listed in Table 3, the functions (F_{12}-F_{15}) needs more time to global and local stages because its initial distribution doesn't cover the entire search space. The values of glob and loc are set as follows:

  $glob$ = 0.5 N, loc = 0.75 N, where N is the number of iterations.

  Table 13 lists FEs instead of N to cope with the reference research, so N is chosen as 6000, 7000, 4000 and 2600 for F_{12}, F_{13}, F_{14} and F_{15} respectively.