Particle Swarm Optimization from Theory to Applications

Article *in* International Journal of Rough Sets and Data Analysis · January 2018

DOI: 10.4018/IJRSDA.2018040101

CITATIONS
19

READS
6,048

2 authors:

M. A. El-Shorbagy
Prince Sattam bin Abdulaziz University
52 PUBLICATIONS **595** CITATIONS

SEE PROFILE

Aboul Ella Hassanien
Cairo University
1,092 PUBLICATIONS **14,265** CITATIONS

SEE PROFILE

International Journal of Rough Sets and Data Analysis
Volume 5 • Issue 2 • April-June 2018

Particle Swarm Optimization from Theory to Applications

M.A. El-Shorbagy, Department of Basic Engineering Science, Faculty of Engineering, Menoufia University, Shebin El-Kom, Egypt

Aboul Ella Hassanien, Department of Information Technology, Faculty of Computers and Information, Cairo University, Giza, Egypt

ABSTRACT

Particle swarm optimization (PSO) is considered one of the most important methods in swarm intelligence. PSO is related to the study of swarms; where it is a simulation of bird flocks. It can be used to solve a wide variety of optimization problems such as unconstrained optimization problems, constrained optimization problems, nonlinear programming, multi-objective optimization, stochastic programming and combinatorial optimization problems. PSO has been presented in the literature and applied successfully in real life applications. In this paper, a comprehensive review of PSO as a well-known population-based optimization technique. The review starts by a brief introduction to the behavior of the PSO, then basic concepts and development of PSO are discussed, it's followed by the discussion of PSO inertia weight and constriction factor as well as issues related to parameter setting, selection and tuning, dynamic environments, and hybridization. Also, we introduced the other representation, convergence properties and the applications of PSO. Finally, conclusions and discussion are presented. Limitations to be addressed and the directions of research in the future are identified, and an extensive bibliography is also included.

KEYWORDS

Optimization Problems, Parameter Setting, Particle Swarm Optimization, PSO, Swarm Intelligence

INTRODUCTION

Particle swarm optimization (PSO) is considered one of the evolutionary computational algorithm which is depend on intelligence of the swarm. It is proposed by Kennedy and Eberhart (1995a; 1995b) where it has been simulated from the artificial livings research. Also, it is a population based optimizer. The PSO mechanism is started by randomly initializing a set of potential solutions, then the search for the optimum is performed repetitively. In PSO algorithm, the optimal position is found by follow the best particles. By Comparison with evolutionary algorithms (EAs), PSO has profound intelligent background which can be performed by more easily. Due to PSO advantages, it is very suitable for research of science, and also in real life problems such as: evolutionary computing, optimization and many other applications. PSO belongs to the swarm intelligence group (Kennedy, Ebherhart, & Shi, 2001) to solve global optimization problems. It was proposed originally as a model of social behavior; where it was firstly proposed as a method of optimization in 1995 (Kennedy & Ebherhart, 1995a). In other words, we can say that PSO is associated with artificial life especially to theories of swarming, simulation of social behaviors and also to EAs.

DOI: 10.4018/IJRSDA.2018040101

PSO can be implemented easily, inexpensive computationally, required low memory and low speed for CPU (Eberhart, Simpson, & Dobbins, 1996). Furthermore, it does not need any information about the objective function such as gradient, but only needs its value. It was proved that PSO is an effective method for solving many global optimization problems and, in some cases, it does not face the difficulties faced by other EA.

In this paper, a comprehensive review of PSO is presented. The review starts by a brief introduction to the behavior of the PSO, then basic concepts and development of PSO is discussed, it's followed by discussion of PSO inertia weight and constriction factor as well as issues related to parameter setting, selection and tuning, dynamic environments, and hybridization. In addition, we introduced the other representation, convergence properties and the applications of PSO. In addition, conclusions and discussion are presented. Finally, we identify the limitations to be addressed, the directions of research in the future and an extensive bibliography is also included.

THE BASIC OF PSO

To understand the PSO, we can imagine a swarm of bees passing through an open field that contains wildflowers. The swarm has a natural desire to locate the highest density position of the flowers in this field. The swarm hasn't any previous knowledge for the field. So, the bees start their search and spread out in random locations. Every bee can remember its locations that have most flowers and this information can be transmitted by communication to the rest of the swarm. With the passage of time, the bees are torn between returning to its previously successful position in finding flowers and heading toward the location (having the most flowers) that reported by the rest of the swarm. The hesitating bee moves in both directions, converting its path to fly somewhere between the two points based on whether social influence dominates its decision or not. Occasionally, a bee may fly over a place in the field that has more flowers than had been discovered by any bee in the swarm. Then, the whole swarm would be drawn as a part in the direction of that location (see Figure 1).

In the figure, dashed lines trace the paths of our imaginary bees, and the solid arrows show their two velocity vector components. The bee No.2 was found the global best position. While, the bee No.1 has found one component of his velocity vector (its personal best location), while the other component is the global best position. The bee No. 3 shows that an example of a particle that have not found a good personal best, it is still drawn to the same position of global best.

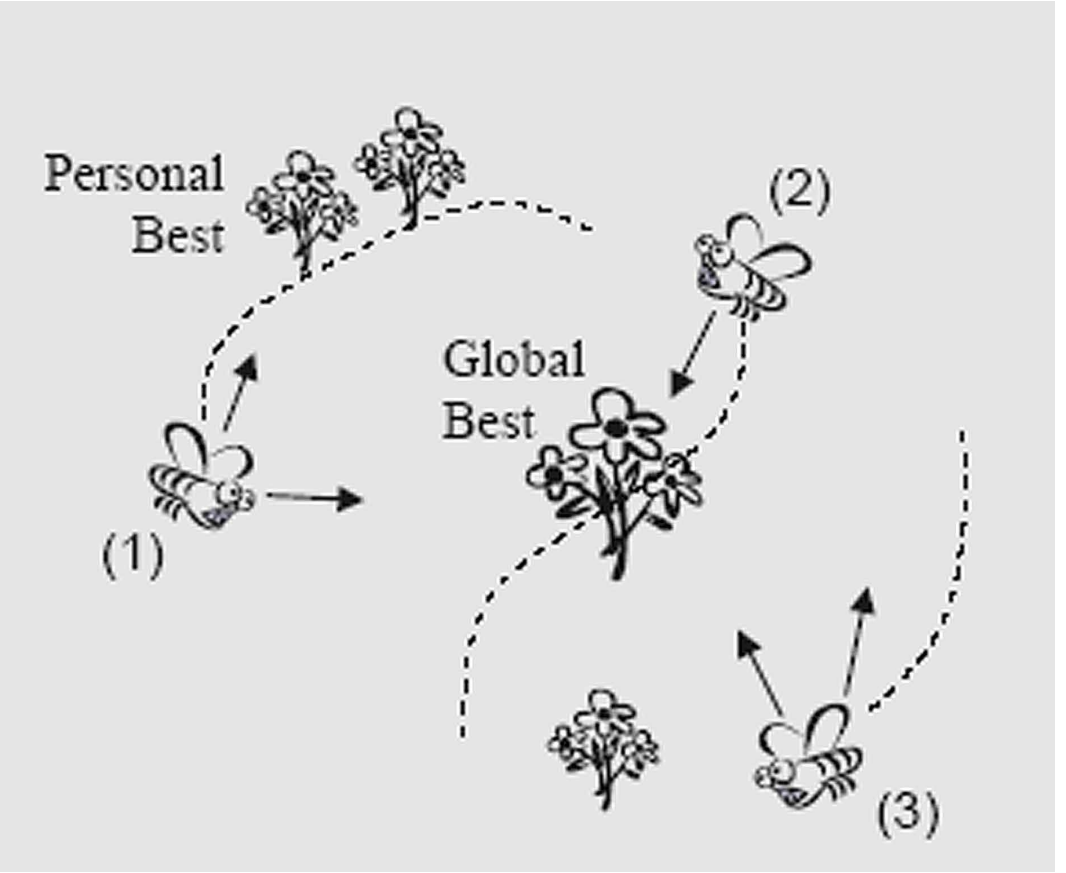
By this manner, the field is explored by the bees by updating its speed and direction of travel depending on its successful at finding flowers in comparison to the rest of the swarm. continuously, they are checked the places to move away from that give lowest concentration of flowers, expecting to find the places that give highest concentration. Finally, all places in the field will be explored by the bees and they will end up by swarming around the position which has the highest flowers concentration. Kennedy and Eberhart (1995a) gave a model describing the behavior of PSO in a computer program. After short time, they realized that their model can be developed as an optimization algorithm and they called it PSO. The term "particle" is used to represent fish, bees, birds, or any other type of natural agents that have the behavior of swarm.

PSO ALGORITHM (PSOA)

The PSOA details are presented as follows: if there is a swarm of p birds or particles, every particle representing a possible point (solution) in the problem space S. Kennedy and Eberhart (2001) originally proposed that the position x^i, per each particle i, is evolved as:

$$x_{k+1}^i = x_k^i + v_{k+1}^i \tag{1}$$

Figure 1. Simulation of PSO by Swarm of Bees Searching for Flowers



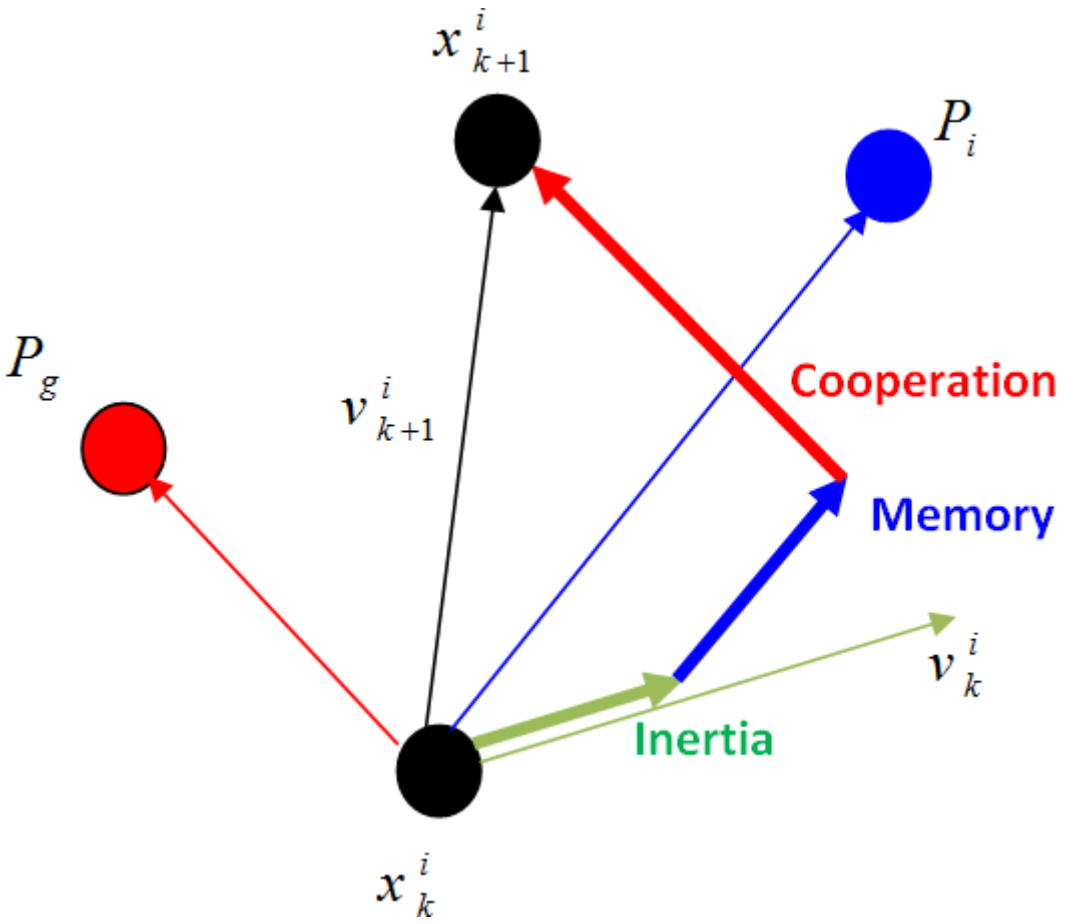
and its velocity v^i is calculated as:

$$v_{k+1}^i = v_k^i + c_1 \times r_1 \times \left(p_k^i - x_k^i\right) + c_2 \times r_2 \times \left(p_k^g - x_k^i\right) \quad (2)$$

The subscript *k*, indicates the increment of a time. p_k^i represents the best position of the particle *i* at time *k* so far, while p_k^g is the global best position in all particles in the swarm at time *k*. r_1 and r_2 are random numbers between 0 and 1. c_1 and c_2 represent the cognitive and social scaling parameters respectively which are selected such that $c_1 = c_2 = 2$ to give a mean equal 1 when they are multiplied by r_1 and r_2 (Kennedy, Eberhart, & Shi 2001). The using of these values make the particles overshooting the target in the half of time. Equation (2) is used to determine the new velocity v_{k+1}^i of the *i-th* particle's, at time *k*, while equation (1) gives the *i-th* particle's new position x_{k+1}^i by adding its v_{k+1}^i to its current position x_k^i. Figure 2 shows the updating of velocity and position of PSO particle's in two-dimensional space.

Let us denote the so far best fitness value of the particle *i* at p_k^i as f_{best}^i and the so far best fitness value of the particle *i* at p_k^g as f_{best}^g. The general PSOA pseudo code can be described in Figure 3.

Figure 2. Updates Velocity and Position of PSO in Two-Dimensional Space



EVOLUTION OF PSOA

The evolution of the original PSOA of Kennedy and Eberhart (2001; Suganthan, 1999) are detailed in this section.

Introduction of Constant Inertia Weight

The first significant variation on the original PSOA is introduced by Shi and Eberhart (1998a); where they added an inertia term *w* into the velocity Equation (2) as:

$$k_{\max} \tag{3}$$

The parameter *w* performs as scaling process on the particle velocity $x_0^i \in S \; in \; R^n$ for $i = 1,\ldots,p.$, i.e. give the particle a momentum. High values of w result in straight paths for particle with significant overflying at the target. This leads to global search with good characteristic. While, low values of *w* make the trajectories of particle erratic with a decreasing in overshoot. This leads to a good local search characteristic. These properties are desirable for refined local search.

Figure 3. The General PSOA Pseudo Code

Step 1. Initialization

(a) Set constants $k_{\max}$, c_1, c_2.

(b) Initialize randomly particles positions $x_0^i \in S \text{ in } R^n$ for $i = 1,\ldots, p.$

(c) Initialize randomly particles velocities $0 \le v_0^i \le v_{\max}^i$ for $i = 1,\ldots, p.$

(d) Set $k = 1$

Step 2. Optimization

(a) Evaluate the function value f_k^i .

(b) If $f_k^i \le f_{best}^i$ then $f_{best}^i = f_k^i$, $p_k^i = x_k^i$.

(c) If $f_k^i \le f_{best}^g$ then $f_{best}^g = f_k^i$, $p_k^g = x_k^i$.

(d) If stopping criterion is satisfied go to step 3.

(e) All velocities of particle are updated v_k^i for $i = 1,\ldots, p$ by equation (2).

(f) All positions of particle are updated x_k^i for $i = 1,\ldots, p$ by equation (1).

(g) $k=k+1$.

(h) Go to Step 2(a).

Step 3. Termination

Reduction of Linear Inertia

Also, Shi and Eberhart (1998a; 1998b) proposed reduction of linear inertia which is a new version of the constant inertia weight. This variation falls under the linear scaling of the inertia parameter *w* during the search process, usually between an initial value of 0.8 and a final value of 0.4 distributed on a specified number of iterations.

Limitation of Maximum Velocity

This variation is proposed by Shi and Eberhart (1998b; 2000); where the velocity of each particle is limited to a specified maximum velocity $0 \le v_0^i \le v_{\max}^i$ for $i = 1,\ldots, p.$. This situation is used to reduce the large step size in the position Equation (1). f_k^i is computed as a fixed fraction $f_k^i \le f_{best}^i$ of the distance between the boundaries of the search domain:

$$f_{best}^i = f_k^i \tag{4}$$

where $p_k^i = x_k^i$ and $f_k^i \le f_{best}^g$ respectively represent the upper and lower bounds of the domain *D*.

Constriction Factor

A constriction factor $f_{best}^{g} = f_{k}^{i}$ was introduced by Clerc (1999) into the velocity Equation (2); where it reduces the effect of the particles velocity as the search progresses. The constriction factor $p_{k}^{g} = x_{k}^{i}$ value is computed as a function of the two parameters c_1 and c_2:

$$v_{k}^{i} \text{ for } i = 1,\ldots,p \tag{5}$$

$$x_{k}^{i} \text{ for } i = 1,\ldots,p \tag{6}$$

Dynamic Inertia with Maximum Velocity Reduction

This evolution, introduced by Fourie and Groenwold (2000, 2002), aims to diminish the sensitivity dependence of the problem on the parameters, which is related to the previous implementations of inertia (Shi & Eberhart, 1998b; 2000). The approach is summarized as follows: Firstly, w_0 is determined and the initial $v_{k+1}^{i} = w \times v_{k}^{i} + c_1 \times r_1 \times \left(p_{k}^{i} - x_{k}^{i}\right) + c_2 \times r_2 \times \left(p_{k}^{g} - x_{k}^{i}\right)$ is calculated. If no improvement occurs after a fixed number of iterations *h* in the fitness values (v_k and $v^{\max}$) of the swarm *i.e.* $v^{\max}$, The domain of the swarm is decreased by reducing the inertia and maximum velocity by two fractions γ and $v^{\max} = \gamma\left(x_{UB} - x_{LB}\right)$; where x_{UB} as:

$$x_{LB} \tag{7}$$

PARAMETER SELECTION

There are a quite many of parameters that must be considered in the PSO algorithm: C_F and the population size (P). So, it might be a good idea to know how these parameters is controlled in the model of PSOA. The aim of setting the PSO parameters is to determine how them optimize the search domain.

The optimization of the search space is depending on setting of the PSO parameters. For instance, the general setting, that gives sensible results on most optimization problems, can be applied, but rarely is optimal. C_F and $v_{k+1}^{i} = C_F \times \left[v_{k}^{i} + c_1 \times r_1 \times \left(p_{k}^{i} - x_{k}^{i}\right) + c_2 \times r_2 \times \left(p_{k}^{g} - x_{k}^{i}\right)\right]$ is the useful setting for the general search. While, $C_F = \dfrac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}$; where $\varphi = c_1 + c_2, \quad \varphi > 4$ depends on the search domain size and properties of the optimization function. Practically, $v^{\max}$ is approximately set to 10% of the average dimension size of search domain.

By these settings, getting optimal is not guaranteed for many problems. So, we must know more information about different settings effects of PSO parameters, to select a favorable value from problem to the other. For example, if the optimization problem has a uni-modal fitness landscape, the PSO must be acted as a hill-climber (HC), and the parameters is setted in a specific way to make PSO an efficient as the behavior of HC. while, if the optimization function has many peaks, we can set the parameters in a different way to adapt the behavior of multimodal problems.

The Control Parameters c_1 and c_2

When setting c_1 and c_2, there are two important facts must be considered. The first fact is that the relation between the two values c_1 and c_2 determines the attraction point p_k^g. If p_k^i, every particle attracted to the mean of $\left[f\left(p_k^g\right) \geq f\left(p_{k-h}^g\right)\right]$ and α. Most implementations of PSOA used a setting with c_1 almost equal to c_2.

In the extreme case β all particles are converted to independent HCs; where the social part $0 < \alpha,\ \beta < 1$ is 0. In the iterated HC, the best point in the neighborhood is obtained by changing the current point, if a better point is found. This step is repeated until access to the local optimal. furthermore, with new starting points, the process can be repeated as many times. On the other hand, with setting $w_{k+1} = \alpha w_k,\ v_k^{\max} = \beta v_k^{\max}$ PSO is flying around the point $w,\ C_F,\ c_1,\ c_2,\ v^{\max}$ and searches a neighborhood. If the better solution is found by the particle, $c_1 = c_2 = 2$ will be updated (if $w = 0.8$ is not the optimum). The particle will start flying around the new $v^{\max}$. After that, the particle is moved, by repeating this process, until the optimum is found, according to the HC behavior. But, the HC particle does not know exactly when finds the optimum solution, because it doesn't search systematically in the neighborhood.

In the other extreme case $v^{\max}$, the part $\dfrac{c_1 p_k^i + c_2 p_k^g}{c_1 + c_2}$ of the particle equals 0 and the particles are attracted to the single point $c_1 = c_2$.

Now, the swarm turn into big HC as previously described; where all particles fly around p_k^g, and change their positions with every update of p_k^i. The neighborhood is scanned by all particles at the same time in a parallel way, as is the case in parallel stochastic HC. If $c_2 = 0$, the particle *i* is very attracted to $c_2\left(p_k^g - x_k^i\right)$, rather than $c_2 = 0,$, and vice a versa if p_k^i. However, in (2001), Carlisle and Dozier obtained on good results with p_k^i and p_k^i

The second fact is the magnitudes of c_1 and c_2. The higher magnitudes of c_1 and c_2, the more acceleration. In fact, at any time *k* the acceleration of particle's is obtained by p_k^i with neglect the inertia and constriction. So, setting the variables (c_1 and c_2) high enables the swarm to change their positions rapidly. While, setting their magnitudes low makes movement and reaction of the swarm slow. Generally, they move far from the point of attraction and will not change their directions as in high magnitudes.

The Inertia Weight (*w*)

The momentum of the particle is controlled by the inertia weight *w*. If $\left(c_1 = 0\right)$, only a little momentum is maintained from the previous time-step. So, the direction is possible changed quickly. If $c_1\left(p_k^i - x_k^i\right)$, the velocity concept is fully lost and the particles move in every step without any information about the previous velocity. While, if *w* >1, we get the same effect of setting c_1 and c_2 are low: Particles hardly change their directions, which means that they explore a larger area with a low probability for convergence to optimum. putting p_k^g must be used with care, because the velocities are biased more to exponential growth. In short, settings *w* near 1 facilitates global search, and settings it in the range (0.2 to 0.5) facilitates rapid local search (LS).

The inertia weight *w* has been studied by Eberhart and Shi (1998b). They found that setting *w* equals 0.8 is a good choice when p_k^g. It is a good choice in many cases, although it is based on one test function (the Schaffer function). Also, Eberhart and Shi applied an annealing scheme for the setting of w; where w changes from 0.9 (start of run) to 0.4 (end of run) (1999). They compared their

model results with results obtained by Angeline at p_k^g (Angeline, 1998), and they concluded that the performance is improved on 4 tested functions. The strategy of decreasing *w* is an optimal setting for alot of problems; where the swarm explores the search domain in the starting of the run, and turns into LS in the ending of the run.

The Constriction Factor (C_F)

As mentioned in section 4, the constriction factor model has the same effect as *w*, except that it also scales the contributions from $c_1 >> c_2$ and p_k^i with p_k^g. Thus, any trying to direction (i.e. the relationship between $c_1 << c_2$ and $c_1 = 2.8$) must be done with c_1 and c_2 with this model. However, $c_2 = 1.3.$ works as *w*: Low values give little exploration and rapid convergence, while high values facilitate much exploration and slow convergence.

One of the few theoretical contributions of PSOA comes from the mathematician Clerc (2002), who has entered the $c_1\left(p_k - x_k\right) + c_2\left(p_k^g - x_k\right)$ on PSO. He has studied the system of PSO by means of second order differential equations. However, this analysis did not model the all particles in the swarm, but only one single deterministic particle in a one-dimensional space. Furthermore, the author assumed $w << 1$ and $w = 0$ are static values. Despite these simplifications, there is one outcome from the analysis. In the model of constriction, $w > 0$ is set as a function in c_1 and c_2, to guaranteed the convergence even without $v^{\max} >> 3$. An additional parameter *m* is introduced to control the convergence speed of the particles to the attraction point, instead of $w = 1$ introduced in Equation 6.

$$p_k^i \tag{8}$$

The main advantage of using from p_k^g instead of *m*, is that *m* clear and reliable to control the behavior of the swarm: If m is near to 0, a fast convergence is obtained (HC behavior). While, if *m* is close to 1, a slowest convergence is obtained but with a high exploration degree, which is useful in multimodal problems.

The Maximum Velocity (v^{max})

Primarily, C_F was introduced to avoid divergence and explosion. With $c_1\left(p_k^i - x_k^i\right) + c_2\left(p_k^g - x_k^i\right)$ or *w*, v_k^i has become unnecessary; where the convergence at least can be reached without it (Clerc & Kennedy, 2002). Thus, for simplifications, C_F do not use by some researchers. Despite this fact, C_F limitation can improve the search. For example, if p_k^i and p_k^g in one end of the search domain and the particle is positioned in the other end, the particle will be capable to obtain a velocity of four times. In other words, if the distance, *d*, from C_F to $v^{\max}$ and C_F is approximately the length of the search domain, and we assume $C_F = \dfrac{2m}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}$ where $\varphi = c_1 + c_2, \quad \varphi > 4, \ m \in \left[0,1\right]$, then from Equation (2) we find that C_F i.e. four times the size of the search domain, which clearly is nonsense.

Hence, after going to the other end of the search domain, the particle is spent a long time in evaluations and slowing its velocity before to turn around. In this time, the search is stopped, and the PSO still uses the CPU-time on evaluating the same boundary solutions again and again. Instead of that, it could have approached the attraction point $v^{\max}$ in a controlled and more efficient way with

suitable C_F and gaining useful information about the fitness landscape between the two corners by sampling various solutions on its way across the search domain.

The Neighborhood Topology

The neighborhood topology is one of the important factors in PSO. The PSO presented in section 3, uses a fully connected neighborhood topology (or *gbest*). In other words, each particle is a neighbor for all other particle. Other topologies are described by Kennedy (1999) as follows:

1. The *n*-best topology; where every particle is connected to its nearest particles *n* in the space of topological (see Figure 4). With $n = 2$, this becomes a circle topology and with $n =$ swarmsize-1 it becomes a *gbest*/neighborhood topology.
2. The wheel topology; where only one central particle is make a connection to the others. In (Kennedy, 1999), many of topologies were studied by Kennedy, and he concluded that the topology has an influence on results. However, it is difficult to find these effects, when the obtained results are analyzed.

Size of Population

The population size selection is problem dependent. The most common ranges of the population sizes are from 20 to 50 particles. It was observed that PSO not needed population greater than other EA to reach high quality solutions. Actually, in most optimization problems 10 particles is enough to obtain good results, while in some complicated or special optimization problems, 100 or 200 particles may be enough to get well results.

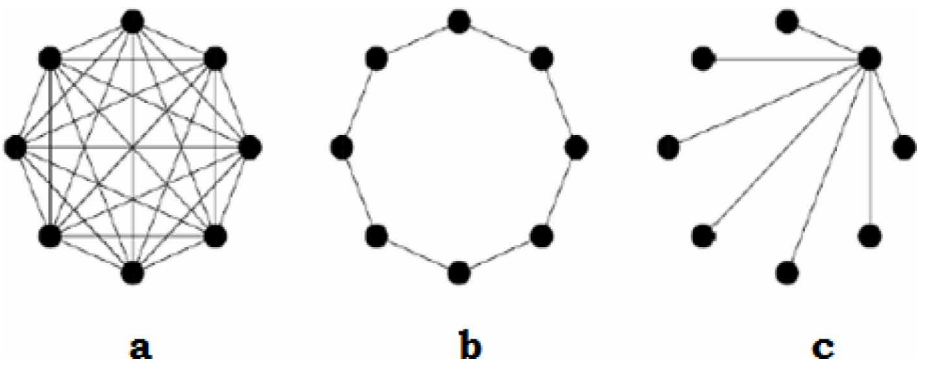
OTHER REPRESENTATION OF PSO

One of the limitations of the PSO algorithm is that it was designed just for real optimization. When PSO is compared with the family of other EA such as genetic algorithms (GAs), it loses some basics of its applicability. Many of researchers tried to introduce an extension of PSO to other representations. In the following, we introduce these extensions of PSO.

Binary/Discrete PSOA

PSOA have been extended to tackle one of the important problems, binary/discrete problem. Firstly, to extend the original version of PSO to binary/discrete problem, there is a critical part must be understood: the concepts of velocity and trajectory in the space of binary/discrete problem.

Figure 4. a) Gbest Topology b) Circle Topology, and c) Wheel Topology



Kennedy and Eberhart (1997) are used the velocity value of the particle as a probability to know whether the bit $v^{\max}$ will be in one state or zero (0 or 1). Kennedy and Eberhart squashed the velocity $v^{\max}$ using a logistic map $v^{\max}$; where the velocity is determined by Equation (2). According to a generated random number within [0,1], if it is greater than p_k^i, then p_k^g is set to be 0, otherwise it is set to be 1. The binary version of PSO outperforms many versions of GAs in most tested problems (Kennedy & Spears, 1998).

Tracking Moving Extrema

Recently, a modification was proposed by Carlisle and Dozier (2000) and Eberhart and Shi (2001), whereby a moving extrema in a dynamic problem environment could be tracked. The dynamic tracking problems considered one of the difficult problems for all EAs, including PSO, in terms of dealing with it. In dynamic problem, the optimization functions change over time. So, the currently solutions which is good, may be worse in the near future. There are three approaches to handle the dynamic tracking problems by PSOA. The first approach to handle this problem is to apply PSO without any changing based on the belief that PSO have ability to fast convergence. The second approach, when a change occurs in the environment optimization, is to reset and reevaluate the previous best. But, this is a convenient for problems which has slow-changing environment. The third approach, is to apply some rules to discover and return to the dynamic functions, i.e. re-randomize particles when change is discovered.

Parallel PSO (PPSO)

Parallel PSO (Chen et al., 2014) aims to reduce the run time by using multiple groups of swarm, concurrently with produce the same results. The mathematical form of the parallel PSOA can be written as:

$$x_k^i \tag{9}$$

$$p_k^i \tag{10}$$

$$p_k^g \tag{11}$$

where $c_1 \approx c_2 \approx 2$, $v_{k+1}^i \approx 4d$, $\left(c_1 p_k + c_2 p_k^g\right) \Big/ \left(c_1 + c_2\right)$ is the groups number, *m* is a positive integer, $v^{\max}$ is the size of particles in the *j-th* group, x^i is the position of the *i-th* particle in the *j-th* group at the *k-th* iteration, v^i is the *i-th* particle velocity in the *j-th* group at the *k-th* iteration, $S\left(v^i\right) = 1 \Big/ \left(1 + \exp\left(-v^i\right)\right)$ is the position of the global best between all particles of the *j-th* group from the first iteration to the *k-th* iteration and $S\left(v^i\right)$ is the best position among x^i from the first iteration to the *k-th* iteration. There are three possible strategies, for parallel PSO, as follows:

- **Strategy 1**: The best particle is migrated among all particles $v_{k+1}^{i,j} = C_F \times \left[v_k^{i,j} + c_1 \times r_1 \times \left(p_k^{i,j} - x_k^{i,j}\right) + c_2 \times r_2 \times \left(p_k^{g,j} - x_k^{i,j}\right)\right]$ to each group and mutate $x_{k+1}^{i,j} = x_k^{i,j} + v_{k+1}^{i,j}$ to replace the worse particles in each group and update $\text{fitness}\left(p^g\right) \le \text{fitness}\left(p_k^{g,j}\right)$ with $i = 0,\ldots,p_j - 1$ for each group for every R1 iterations.

- **Strategy 2**: The best particle position $j = 0,...,S-1$ is migrated of the j-th group to the q-th groups to substitute some worse particles in each receiving group for every R2 iterations. Here $S\left(=2^m\right)$.
- **Strategy 3**: The groups are separated into two subgroups. The strategy No.1 is applied to the subgroup 1 after every R1 of iterations, while the strategy No.2 is applied to both subgroup No.1 and subgroup No.2 after every R2 of iterations.

Quantum PSO (QPSO)

The QPSO is presented in (Yang, Wang, & Jiao, 2004) as an algorithm with good performance-complexity trade-off. According to the biological and social principles of the swarms, QPSO is based on the behavior of group of various animals, associated to the quantum principle of mechanics physical. As in the quantum theory, a *qubit* is defined as the smaller unit that carries information. The *i-th* particle is defined with quantum energy by:

$$p_j \tag{12}$$

where *L* is the particle length, $x_k^{i,j}$ and $v_k^{i,j}$.

The term $p_k^{g,j}$ means the probability of a bit to be "0", and has to be converted to a discrete form. Therefore, the discrete particle p^g is obtained through the statement:

$$\text{If } p_k^{g,j} \tag{13}$$

where p^g is a random value modeled through p^g

As in the PSOA, QPSO algorithm has memory to store the best position values that found for every particle ($p_k^{g,j}$) and the best global position (p^g). From these positions, the best global and individual quantum energy values are determined to make changes in the positions of particle.

$$p_k^{g,j} \tag{14}$$

$$q = j \otimes 2^m,\ j = 0,...,S-1,\quad m = 0,...,n-1 \text{ and } \quad S = 2^n \tag{15}$$

where α and β are the parameters that control the step for the q function, with $q_i\left[k\right] = \left[q_{i1}\left[k\right] \quad q_{i2}\left[k\right] \quad ... \quad q_{iL}\left[k\right]\right]^T$. The *i-th* particle energy is updated as:

$$i = 1,\ 2,...,P \tag{16}$$

where $q_{iL}\left[k\right] \in \left(0,1\right)$ and $q_{iL}\left[k\right]$ are represent the weight for every component of the energy, with $x_i\left[t\right] = \left[x_{i1}\left[t\right] \quad x_{i2}\left[t\right] \quad \cdots \quad x_{iK}\left[t\right]\right]^T$

Hybridizing with Other Types of Algorithms

Hybrid PSO is a combination between the PSOA and other algorithms such as the EA techniques. EA operators, selection, crossover and mutation, have been merged inside the procedures of PSOA. The combination between the selection operation and PSO help to copy the best performance particles into the next offspring; thus, PSO keeps the particles that have best performance in its memory (Angeline, 1998). On the other hand, the combination between crossover operator and PSOA helps the particles to fly in a new area in the search domain as in GAs; where the information can be transferred between two particles (Løvbjerg, Rasmussen, & Krink, 2001). The mutation operator is considered one of the commonly evolutionary operators that used in PSO. The aim of merging mutation operator into PSO is to rise the diversity of the particles and give PSO the capability to go to the global minima (Miranda & Fonseca, 2002).

There are many of hybrid algorithms between PSO and other techniques such as: hybrid PSO and GA (Ali & Tawhid, 2016; Garg, 2016; Grosan, Abraham, Han, & Gelbukh, 2005; Premalatha, & Natarajan, 2010; Kao & Zahara, 2008), hybrid PSO and ant colony optimization (ACO) (Mandloi & Bhatia, 2016; Shelokar, Siarry, Jayaraman, & Kulkarni, 2007), hybrid PSO and firefly algorithm (FA) (Kora & Sri Rama Krishna, 2016), hybrid PSO and artificial bee colony (ABC) (Li, Wang, Yan & Li, 2015), hybrid PSO and bacterial foraging (Dhillon, Lather & Marwaha, 2016), hybrid PSO and Levy flight (Hakli & Uğuz, 2014), hybrid PSO and gravitational search algorithm (GSA) Das, Behera & Panigrahi, 2016; Mirjalili & Hashim, 2010, hybrid PSO and Simulated Annealing (SA) (Idoumghar, Melkemi, Schott & Aouad, 2011), hybrid PSO with chaos (Liu et al., 2005; Alatas, Akin & Ozer, 2009; Tan, 2012; Gandomi, Yun, Yang & Talatahari, 2013) and hybrid PSO with LS (Chen, Qin, Liu & Lu, 2005; Qin, Yin & Ban, 2010; Qu, Liang & Suganthan, 2012) etc.

PSO CONVERGENCE PROPERTIES

There are many theoretical studies have been published about the PSO convergence properties. These studies have concluded, as in EAs, that PSOA is very affected by the selection of parameters.

Most studies are applied to the simplified model of PSO; where the swarm is containing only one particle in one dimension. In addition, *pbest* and *gbest* particles are considered constants during the optimization process. While, the terms $\rho_{iL}\left[k\right] > q_{iL}\left[k\right],\ \text{then}\ x_{iL}\left[k+1\right]=1,\ \text{else}\ \ x_{iL}\left[k+1\right]=0$ are set as constant values. Under these assumptions, the trajectories of particle and its convergence have been analyzed. The PSO convergence is defined as in the following definition:

Definition 1: According to the sequences of global best as $\rho_{iL}\left[k\right]$, the PSOA is converged iff

$$U\left(0,\ 1\right). \tag{17}$$

where *p* is any position in the space domain. So, Definition No.1 don't determine whether the point *p* is global or local.

Ozcan and Mohan (1998) are proposed the first study about the PSO convergence properties. But, the model of Ozcan and Mohan did not take into account the inertia weight. They showed that if $x_{best}^{i}\left[k\right]$, the trajectory shape of particle is a sine wave; where the frequency and amplitude of the wave is determined by parameter choices and initial conditions. Also, they concluded that the nature of trajectory may be make the particle searching repeatedly in a region it visited before, unless other particle in the neighborhood finds a good solution.

On the other hand, Van den Bergh (2002) proposed the same model of Ozcan and Mohan (1998) but take into account the inertia weight of particle. He proved that, if $x^g_{best}\left[k\right]$, the particle converges and go to the point $q^g_{best}\left[k\right]=\alpha\cdot x^g_{best}\left[k\right]+\beta\cdot\left(1-x^g_{best}\left[k\right]\right)$. While, if $q^i_{best}\left[k\right]=\alpha\cdot x^i_{best}\left[k\right]+\beta\cdot\left(1-x^i_{best}\left[k\right]\right)$ *i.e.* $\alpha+\beta=1$, the particle converges and go to the point $q_i\left[k+1\right]=c_1\cdot q_i\left[k\right]+c_2\cdot q^i_{best}\left[k\right]+c_3\cdot q^g_{best}\left[k\right]$.

These results were proved under the condition of c_1, c_2 and c_3 being constant values. In addition, Van den Bergh circulated his model using the stochastic nature of $c_1+c_2+c_3=1.$ and $\phi_1=c_1r_1, \phi_2=c_2r_2$; where he assumed uniform distributions. In this case, he concluded that the particle converges to the point $\left\{gbest_k\right\}_{k=0}^{\infty}$; where $\lim_{k\to\infty} gbest_k=p$ i.e. the particle converges to a point between its neighborhood best and its personal best position.

To guarantee convergence, the condition $0<\phi<4,\ \left(\phi=\phi_1+\phi_2\right)$ must be hold. Maybe, we can choose any values of c_1, c_2 and *w* such that this condition is violated and the swarm still converges (Van den Bergh, 2002): The swarm has convergent behavior if $w>\frac{1}{2}\left(c_1+c_2\right)-1$ is near to 1.0; where $\frac{\phi_1 pbest+\phi_2 gbest}{\phi_1+\phi_2}$. This means that the particle trajectory will converge all time with taking divergent steps occasionally.

The studies in Ozcsn and Mohan (1998) and Van den Bergh (2002) assumed that the trajectories are not constricted. Clerc and Kennedy, in (2002), were gave theoretical analysis of particle behavior; where they were introduced the constriction factor which aims to prevent the velocity and maintain the particle inside the search domain.

In (2006), Sierra and Carlos Coello introduced two necessary conditions to guarantee convergence to the global or local optimum:

1. Monotonic condition: the gbestk+1 position can be better than the gbestk position.
2. PSOA must be able to generate a solution in the optimum neighborhood with probability (nonzero), from any solution x in the search domain.

In (2002), Van den Bergh gives a proof that the standard PSO is not a local (neither global) optimizer. Because, although PSO achieves the monotonic condition, once the algorithm reaches to this state: *x* = *pbest* = *gbest* for all particles, no further progress will be achieved. The main problem is the swarm may be reached to this state before gbest goes to a minimum (global or local). Therefore, the PSOA is said to be converge prematurely. Finally, we can say that the standard PSOA is not a local (global) search technique; where it has no ensured convergence to the global (local) minimum from an arbitrary initial point.

Also, van den Bergh offers two methods of extending PSOA to be a global search technique. The first one, is to the generate new random solutions to force PSOA to perform a random search in an area around the global best position to prevent stagnation. While, the second method is to propose a multi-start PSO.

From the discussion previously provided, Sierra and Carlos Coello (2006), proved that it is possible to guarantee convergence, by determining the flight formula parameters correctly. But, in multi-objective optimization (MOO), this property does not guarantee the convergence to the true curve of Pareto front, as in single optimization. In MOO case, they concluded that to ensure convergence, we still need the two conditions (1) and (2) with change the condition (1) to:

- The solutions added to the archive of solutions at iteration $c_1 = c_2 = c$ should be nondominated with the solutions obtained in all iterations so far.

The use of $w > c-1$-dominance archiving technique (Sierra & Coello, 2006) achieves this condition, while the normal dominance strategies do not. To ensure this condition with the normal dominance-based strategies (Sierra & Coello, 2006), we must sure that any solution leave the archive is dominated by the added solution. By this way, any multi-objective particle swarm optimization (MOPSO) approach may be satisfy the condition No.1, while we must prove that it achieves the condition No.2, to guarantee the global convergence to the true curve of Pareto front.

APPLICATIONS OF PSO

PSO has many advantages such as: easy to implement, simple in concept and few parameters to adjust. So, PSO is applied to solve too large number of applications. PSO, like most techniques of EAs, can be used for solving most problems of optimization. At this time, PSOA is applied to all types problems whether its search space variables are real or binary. There are a major number of problems that its results obtained by PSO are very competitive.

Constrained Optimization Problems (COPs)

Constrained optimization problems (COPs) is considered one of the very important applications areas for PSO. The major troubles which we face when solving COPs is that how the constraints is handled. There are three approaches is used to handle the constraints in COPs. The first approach is to convert the COP to a non-COP by adding the penalty function to violate the constraints (Parsopoulos & Vrahatis, 2002). While, the second approach is to repair the infeasible solutions and keep the feasible solutions (Hu & Eberhart, 2002). In the third algorithm (called hybrid algorithms) some information decoding strategies is employed. Ray and Liew (2001) used a constraint matrix to handle the constraints. By this matrix, better performer list is generated; where it used to determine the search direction for the residual particles. In the following, we will review some of algorithms that solving COP.

Yadav and Deep (2014) proposed a new co-swarm PSO, which is obtained by combining shrinking hypersphere PSO (SHPSO) with the differential evolution (DE). The swarm was divided to two sub swarms; where the first sub swarms uses SHPSO and the second sub swarms uses DE. The Tests were performed on one of the state-of-the art problems proposed: IEEE CEC 2006. The results of the CSHPSO are compared with SHPSO and DE in a variety of fashions. Experimental results prove that CSHPSO is considered a good new algorithm; where it can be used to solve any real COP.

Sun, Zeng, and Pan (2011) proposed an improved vector particle swarm optimization (IVPSO) algorithm; where the constraint is handling by a technique depending on the simple constraint preserving method. Position and velocity of all particles and its changes are expressed by vectors in order to present the procedures of optimization by a more understanding method. The authors proposed multi-dimensional search methodology to search in the local region and obtain new feasible positions. By This manner (multi-dimensional search) the authors avoided neglecting any good positions in the feasible region and improve optimization efficiency. The performance of IVPSO is evaluated by a well-known 13 benchmark problems. Results show that IVPSO is competitive, stable and simple stable.

Elsayed, Sarker, and Mezura-Montes (2014) proposed a new self-adaptive PSOA which can be solve a variety of COPs efficiently. This approach is used different mix of PSO variants; where each of which has different number of particles from the current population that used for evolving. In every iteration, the algorithm sets more particles to the better-performing variants and fewer to the worse-performing ones. In addition, new methodology for adapting PSO parameters is proposed. The performance of self-adaptive PSOA was analyzed and tested by two sets problems, CEC2006 and CEC2010. The result showed a significantly better performance than the other comparison algorithms.

Abd-El-Wahed, Mousa, and El-Shorbagy (2011) presented a hybrid approach combining PSO with GA for solving COPs; where it integrates the advantages of both techniques. At first the algorithm is begun by initializing random particles which travel in the search region. During this, integrating PSO and GA is used in the evolution process of these particles. Secondly, a modified constriction factor is introduced to control the velocity of the particles and restrict it. The results of various tests have showed the efficiency of the proposed algorithm for solving COPs

Min/Max Problems

One of the important problems that was solved by PSOA is min/max problems (Laskari, Parsopoulos, & Vrahatis, 2002; Shi & Krohling, 2002). There are 2 approaches used for solving this type of problems. The first approach is handle the min-max problem by the same method in the minimization problem and the maximum part is embedded in the calculation of the objective (fitness values). By this approximation, the obtained solutions by this approach can meet the requirements of the min/max problem (Laskari, Parsopoulos, & Vrahatis, 2002). The second approach transforms the min/max problem to two optimization problems (minimum problem and maximum problem). Two swarms of PSO are used for solving the two problems, respectively, but with independent run. The swarm of minimization problem is used as a changing environment of the other swarm of maximization problem and vice versa. In addition, the two swarms cooperate through the calculation of fitness (Shi & Krohling, 2002).

Multi-Objective Optimization Problems (MOOPs)

In MOOPs, there are multiple objectives need to be optimize at the same time. In most MOOPs, no one solution is found to be optimal to all objectives. But, there are a huge of alternative solutions which are known as a Pareto front. In addition, these solutions are equal in their preference. So, there is a difficulty in choosing between them.

MOOP is one of the important application areas which has studied by PSOA. Many of algorithms have been designed to handle MOOPs by using PSO. In the first algorithm, that was used to solve MOOP, is converted MOOP to a single objective optimization problem (SOOP).

The first study used to solve MOOPs by PSOA has been proposed by Parsopoulos and Vrahatis (2002); where the weighted aggregation method was merged in PSO to find the Pareto front solutions. Many PSOAs were proposed to solve MOOPs. A comprehensive survey for the PSOAs that are solving MOOPs can be found in (Sierra & Coello, 2006); where different techniques have been discussed and categorized. In the following, we will review some of algorithms that solving MOOPs.

In (Hwang, Koo, & Lee 2006), Homogenous PSO (HPSO) for MOOP is proposed; where single global archive concept is used for determining *gbest* and *pbest*. This means that the particles have been giving up its special identity and dealt as a member of social set. To avoid premature convergence, HPSO needs, in the first iteration, at least two or more Pareto solutions. HPSO have been tested to illustrate its ability to find the Pareto optimal set for MOOPs.

Cagnina, Esquivel, and Coello (2005) proposed simple multi-objective particle swarm optimizer (SMOPSO); where it combines an elitist policy, Pareto dominance, and two schemes to keep diversity. The major modifications that used to handle MOOPs are uniform mutation operator, an elitist policy, and new technicality to select the *gbest* and *pbest* particles.

Hu and Eberhart (2002) proposed a modified PSO; where they used a strategy of dynamic neighborhood, new updating for particle memory and deal with multiple objectives by one-dimension optimization. The dynamic neighborhood PSO was applied only on optimization problems that have two objectives. The best position *pbest* is updated only when the current solution is dominated by a new solution. Based on these setting, PSOA found multiple optimal solutions successfully and determined the Pareto front.

Parsopulos, Tasoulis, and Vrahatis (2004) proposed a new parallel version: vector evaluated particle swarm optimization (VEPSO) for solving MOOPs; where it is inspired from the parallel

version of GA. In VEPSO, each of the objective function is evaluated by using one swarm, and the information is transferred between the swarms to exchange their best experience.

In (Mousa, El-Shorbagy, & Abd-El-Wahed, 2012), a hybrid multi-objective EA integrates the merits of both PSO and GA is proposed. The algorithm is started by group of random particles that fly in the problem domain, searching for optimization. Then GA is applied to evolve these particles. In addition, LS technique is used to improve the quality of solution; where it aims to decrease the distance between the obtained solutions of PSO/GA by generating more nondominated solutions between them. Many benchmark problems including the set of benchmark functions provided for CEC09 have been reported to show the importance of this technique in solving MOOPs.

El-Shorbagy (2015) proposed hybrid approach combining trust region algorithm (TRA) and PSOA. Firstly, the weighted method is used to convert the MOOP to a SOOP. Secondly, the SOOP is solved by TRA in order to get some points on the Pareto frontier which are used as positions of particles for PSOA. Finally, homogeneous PSO is applied to obtain the all nondominated solutions. Many kinds of benchmark problems have been tested and the results demonstrated that TRA/PSOA has the ability to solve MOOPs and generate the Pareto optimal solutions.

Meza, Espitia, Montenegro, Giménez, and González-Crespo (2017) presented multi-objective vortex PSO (MOVPSO) as a strategy based on the behavior of a particle swarm using rotational and translational motions. The MOVPSO strategy is based upon the emulation of the emerging property performed by a swarm (flock), achieving a successful motion with diversity control, via collaborative, using linear and circular movements. MOVPSO is tested through several MOO functions and is compared with standard multi-objective MOPSO. The qualitative results show that MOVPSO behave as expected.

Other Applications

In addition to the previous three main applications of PSO, it has been implemented successfully for solving many of others problem that containing a lot of real life problems. In 1995, PSO was applied to the first real life application in the field of neural networks. In this application, PSOA was capable of train and set the weights in a feed-forward neural network as effectively (Kennedy & Eberhart, 1995a). Since then, PSO has been applied to a lot of real life applications in many fields due to its efficiency, simplicity and nature of fast convergence. Generally, we can say that PSO can be applied for solving most of optimization problems and real-life applications which can be described as optimization problems. In the following, we will review some of algorithms.

In (Li, Jiao, Zhao, Shang, & Gong, 2017), a quantum-behaved discrete multi-objective PSO algorithm is proposed for solving complex network clustering as a MOOP. In addition, the non-dominant sorting selection operation is employed for individual replacement. The experimental results showed that the proposed algorithm performs effectively and achieves competitive performance with the latest methods.

In (Sethanan & Neungmatcha, 2016), a new structure of PSO is proposed; where gbest, lbest and nbest (neighbor best position) are combined for solving sugarcane mechanical harvester route planning (MHRP) is presented. A new encoding/decoding scheme of the particle is designed for merging the path planning with the accessibility and split harvesting constraints. Numerical results on many networks with sugarcane field topologies, comparing with other algorithms, showed the efficiency of the new structure of PSO for computation of MHRP.

In (Kesharaju & Nagarajah, 2016), a binary coded PSO technique is proposed to defect detection in armour ceramics. It is investigated in the implementation of feature subset selection and to optimize the classification error rate. In addition, the population data is used as input to an artificial neural network (ANN) based classification system to get the error rate, as ANN serves as an evaluator of PSO fitness function.

In (Mousa & El-Shorbagy, 2012), an enhanced PSOA was presented for solving the problem of reactive power compensation; where it is a combination between GA and PSO. This algorithm

begins by initialize a set of random particles which roving in the search domain, then these particles are evolved by PSO/GA to get approximate nondominated solution. In addition, the solution quality is improved by dynamic search technique. This enhanced PSOA was evaluated by By applying it to one of the standard power systems (IEEE 30—bus/6—generator). The numerical results demonstrated ability of this algorithm to get Satisfactory solutions of reactive power compensation problem.

Das, Abraham, and Sarkar (2006) presented a hybrid rough-PSO technique for Image Pixel Classification; where them dealt with the image segmentation as a clustering problem. Every cluster was modeled as a rough set. PSOA was used to set the threshold and relative importance of lower and upper approximations of the rough sets. Davies–Bouldin clustering validity index was used as the fitness function, which is minimized while arriving at an optimal partitioning.

In (Chandramouli & Izquierdo, 2006), the chaotic modeling of PSO is presented with image classification application. The performance of this model is compared with standard PSO. Numerical results are performed of this comparative study on images binary classes of from the Corel dataset. The simulation result shows a considerable increase in the image classification accuracy with the proposed chaotic PSO algorithm.

CONCLUSION

In this paper, we presented the main concepts of the population based optimization technique PSO. Firstly, the main concepts of PSO with its basic definitions are proposed. Secondly, evolutions of PSO, rules of parameter selection, other representation are described with convergence properties. Finally, different applications of PSO are introduced.

From the previous survey, we can learn that PSO has many common points with EAs; where it starts with a randomly generated particles (population) and has fitness values to evaluate it. In addition, the particles (population) is updated and searched for the optimum with a random technique. Finally, PSO has many strengths and weakness which is described as follows:

Strengths

PSOA gained their popularity from its straightforward concept and simplicity in computation (Kennedy, Eberhart, & Shi, 2001). Particle swarms fundamentally used two simple formulae to search about the optimum effectively. Also, many researchers have compared between PSO and other techniques on a variety set of problems (Boeringer & Werner, 2004; Salman, Ahmad & Al-Madani, 2002; Zhao & Cao, 2005; Ghoshal, 2004). It proved that PSO has better performance in some problems and able to competitive in other. PSO is a good search technique. So, many of researches is aiming to improve the standard PSO to solve various problems. In addition, PSO has great potential which can do further. For example, due to its similarity with EAs, PSO is integrated with many ideas of EAs to improve it. Like many EAs, PSO has a number of parameter to adjust. On one hand, this is beneficial for implementing adaptive systems (Kennedy & Eberhart, 1995a) and also shows the extensibility of PSO to other specifically designed algorithms although it may not perform as well as those algorithms. On the other hand, tuning parameters for solving a particular problem or a range of problems can be time-consuming and non-trivial. Compared with EAs, PSO has a few parameters that adjust in order to get acceptable performance (Hu & Eberhart, 2002). In addition, PSO is applicable for both unconstrained and constrained problems even without transforming any of the constraints or the objectives of a problem (Hu & Eberhart, 2002).

Weaknesses

Researchers have found many limitations that avoid the generic PSO from effectively solving certain types of optimization problems. Although, improvements have been working on to handle these limitations, these improvements may be not useful when used it to solve other problems. Thus, we should store these limitations in mind when developing new PSOA to solve other problems. For

example, although PSO has the ability to converge quickly, it tends to wander and slow down as it approaches an optimum (Vesterstrom & Riget, 2002). Owing to the premature convergence, it gets stuck quite easily and cannot be explore the domain of search enough. This problem appears when solving multimodal problems where the optimal solutions is multiple. Particularly, if many of these optimal solutions are local, PSOA may be trapping into local optimal. In addition, while there are little parameters as previously mentioned (Hu & Eberhart, 2002), these parameters open up a potential for evolving PSOA, some of these parameters depend on the problem. Some suggested values and experimental settings are still at trial-and-error stage (Eberhart & Shi, 2001), and it can be non-trivial to find the right settings for individual problems.

Future Directions of Research

There are many directions of research for PSOA which can be proposed in the following points:

- Deeply statistical analysis for PSO parameters must be studied to show how it affect the convergence of swarm to optimal solution of the problem.
- Introducing new terms to the velocity equation for the PSOA which simulating the wind speed and the vision factor in the search domain
- Testing different chaotic maps for controlling the PSO parameters.
- Hybridizing PSO with new optimization techniques such as: Glowworm Swarm Optimization (Eberhart & Shi, 2001), Monkey Algorithm (MA) (Zhou, Chen & Zhou, 2016), Krill Herd Algorithm (Bolaji, Al-Betar, Awadallah, Khader & Abualigah, 2016) and A Sine Cosine Algorithm (Mirjalili, 2016).
- Solving real life problems that more complex/larger scale to discover new limitations in PSOA.

REFERENCES

Abd-El-Wahed, W. F., Mousa, A. A., & El-Shorbagy, M. A. (2011). Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems. *Journal of Computational and Applied Mathematics*, *235*(5), 1446–1453. doi:10.1016/j.cam.2010.08.030

Alatas, B., Akin, E., & Ozer, A. B. (2009). Chaos embedded particle swarm optimization algorithms. *Chaos, Solitons, and Fractals*, *40*(4), 1715–1734. doi:10.1016/j.chaos.2007.09.063

Ali A.F., Tawhid M.A. (2016). A hybrid particle swarm optimization and genetic algorithm with population partitioning for large scale optimization problems. *Ain Shams Engineering Journal*.

Angeline, P. J. (1998). *Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Differences. In Evolutionary Programming VII,* LNCS (Vol. 1447, pp. 601–610). Springer.

Angeline, P. J. (1998). Using selection to improve particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998)*, Anchorage, USA. doi:10.1109/ICEC.1998.699327

Boeringer, D. W., & Werner, D. H. (2004). Particle swarm optimization versus genetic algorithms for phased array synthesis. *IEEE Transactions on Antennas and Propagation*, *52*(3), 771–779. doi:10.1109/TAP.2004.825102

Bolaji, A. L., Al-Betar, M. A., Awadallah, M. A., Khader, A. T., & Abualigah, L. M. (2016). A comprehensive review: Krill Herd algorithm (KH) and its applications. *Applied Soft Computing*, *49*, 437–446. doi:10.1016/j.asoc.2016.08.041

Cagnina L., Esquivel S., Coello C.C. (2005). A Particle Swarm Optimizer for Multi-Objective Optimization. *J. Comput. Sci. Technol.*, *5*(4), 204-210.

Carlisle, A., & Dozier, G. (2000). Adapting particle swarm optimization to dynamic environments. In *Proceedings of the International Conference on Artificial Intelligence*, Las Vegas (Vol. 1, pp. 429–434).

Carlisle, A., & Dozier, G. (2001). An off-the-shelf PSO. In Proceedings of the workshop on particle swarm optimization, Purdue School of engineering and technology, Indianapolis, USA.

Chandramouli, K., & Izquierdo, E. (2006). Image Classification using Chaotic Particle Swarm Optimization. *2006 International Conference on Image Processing*, Atlanta, GA, 3001-3004. doi:10.1109/ICIP.2006.312968

Chen, H. L., Yang, B., Wang, S. J., Wang, G., Liu, D. Y., Li, H. Z., & Liu, W. B. (2014). Towards an optimal support vector machine classifier using a parallel particle swarm optimization strategy. *Applied Mathematics and Computation*, *239*, 180–197. doi:10.1016/j.amc.2014.04.039

Chen, J., Qin, Z., Liu, Y., & Lu, J. (2005). Particle Swarm Optimization with Local Search. In *Proceedings of the 2005 International Conference on Neural Networks and Brain*, Beijing (pp. 481-484). doi:10.1109/ICNNB.2005.1614658

Clerc, M. (1999). The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In P.J. Angeline, Z. Michalewicz, & M. Schoenauer et al. (Eds.), *Proceedings of the Congress of Evolutionary Computation, Mayflower Hotel*, Washington D.C. (Vol. 3, pp. 1951–1957). IEEE Press. doi:10.1109/CEC.1999.785513

Clerc, M., & Kennedy, J. (2002). The Particle Swarm Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation*, *6*(1), 58–73. doi:10.1109/4235.985692

Das, P. K., Behera, H. S., & Panigrahi, B. K. (2016). A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. *Swarm and Evolutionary Computation*, *28*, 14–28. doi:10.1016/j.swevo.2015.10.011

Das, S., Abraham, A., & Sarkar, S. K. (2006). A Hybrid Rough Set--Particle Swarm Algorithm for Image Pixel Classification. In *Proceedings of the 2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06)*, Rio de Janeiro, Brazil, 26-26. doi:10.1109/HIS.2006.264909

Dhillon, S. S., Lather, J. S., & Marwaha, S. (2016). Multi objective load frequency control using hybrid bacterial foraging and particle swarm optimized PI controller. *International Journal of Electrical Power & Energy Systems*, *79*, 196–209. doi:10.1016/j.ijepes.2016.01.012

Eberhart, R., Simpson, P., & Dobbins, R. (1996). *Computational Intelligence PC Tools*. Boston, MA: Academic Press Professional.

Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the 1995 6th International Symposium on Micro Machine and Human Science* (Vol. 6, pp. 39–43). doi:10.1109/MHS.1995.494215

Eberhart, R. C., & Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. In *Proc. of the 2000 Congress on Evolutionary Computation*, Piscataway, NJ (pp. 84–88). IEEE. doi:10.1109/CEC.2000.870279

Eberhart, R. C., & Shi, Y. (2001). Tracking and optimizing dynamic systems with particle swarms. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001* (pp. 94–100). IEEE Press. doi:10.1109/CEC.2001.934376

Eberhart, R. C., & Shi, Y. (2001). Particle Swarm Optimization: Developments, Applications and Resources. In *Proceeding of the 2001 congress on evolutionary computation (CEC2001)*, Piscataway, NJ (Vol. 1, pp. 81-86). doi:10.1109/CEC.2001.934374

El-Shorbagy, M. A. (2015). Weighted Method Based Trust Region-Particle Swarm Optimization for Multi-Objective Optimization. *American Journal of Applied Mathematics*, *3*(3), 81–89. doi:10.11648/j.ajam.20150303.11

Elsayed, S. M., Sarker, R. A., & Mezura-Montes, E. (2014). Self-adaptive mix of particle swarm methodologies for constrained optimization. *Information Sciences*, *277*, 216–233. doi:10.1016/j.ins.2014.01.051

Fourie, P. C., & Groenwold, A. A. (2000). Particle swarms in size and shape optimization. In J.A. Snyman, & K. Craig (Eds.), *Proc. Workshop on Multidisciplinary Design Optimization*, Pretoria, South Africa (pp. 97–106).

Fourie, P. C., & Groenwold, A. A. (2002). The particle swarm optimization algorithm in size and shape optimization. *Structural and Multidisciplinary Optimization*, *23*(4), 259–267. doi:10.1007/s00158-002-0188-0

Gandomi, A. H., Yun, G. J., Yang, X.-S., & Talatahari, S. (2013). Chaos-enhanced accelerated particle swarm optimization. *Communications in Nonlinear Science and Numerical Simulation*, *18*(2), 327–340. doi:10.1016/j.cnsns.2012.07.017

Garg, H. (2016). A hybrid PSO-GA algorithm for constrained optimization problems. *Applied Mathematics and Computation*, *274*, 292–305. doi:10.1016/j.amc.2015.11.001

Ghoshal, S. P. (2004). Optimization of PID Gains by Particle Swarm Optimizations in Fuzzy Based Automatic Generation Control. *Electric Power Systems Research*, *72*(3), 203–212. doi:10.1016/j.epsr.2004.04.004

Grosan, C., Abraham, A., Han, S., & Gelbukh, A. (2005). Hybrid Particle Swarm – Evolutionary Algorithm for Search and Optimization. In A. Gelbukh, Á. de Albornoz, & H. Terashima-Marín (Eds.), *MICAI 2005: Advances in Artificial Intelligence. MICAI 2005* (p. 3789). Berlin, Heidelberg: Springer. doi:10.1007/11579427_63

Haklı, H., & Uğuz, H. (2014). A novel particle swarm optimization algorithm with Levy flight. *Applied Soft Computing*, *23*, 333–345. doi:10.1016/j.asoc.2014.06.034

Hu, X., & Eberhart, R. C. (2002). Solving constrained nonlinear optimization problems with particle swarm optimization. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*, Orlando, FL.

Hu, X., & Eberhart, R. C. (2002). Multiobjective optimization using dynamic neighborhood particle swarm optimization. In Proceedings of the 2002 Congress on Evolutionary Computation, Honolulu, Hawaii (pp. 12-17).

Hwang, S. K., Koo, K., & Lee, J. S. (2006). *Homogeneous Particle Swarm Optimizer for Multi-objective Optimization Problem*. ICGST Journal of Artificial Intelligence and Machine Learning.

Idoumghar, L., Melkemi, M., Schott, R., & Aouad, M. I. (2011). Hybrid PSO-SA Type Algorithms for Multimodal Function Optimization and Reducing Energy Consumption in Embedded Systems. Applied Computational Intelligence and Soft Computing.

Kao, Y., & Zahara, E. (2008). A hybrid genetic algorithm and particle swarm optimization for multimodal functions. *Applied Soft Computing*, *8*(2), 849–857. doi:10.1016/j.asoc.2007.07.002

Kennedy, J. (1999). Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance. In Proceedings of the 1999 Congress of Evolutionary Computation (Vol. 3, pp. 1931-1938). IEEE Press.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks* (Vol. 4, pp. 1942–1948). IEEE Press. doi:10.1109/ICNN.1995.488968

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In *Proc. 1997 Conf. on Systems, Man, and Cybernetics* (pp. 4104-4109). doi:10.1109/ICSMC.1997.637339

Kennedy, J., Eberhart, R. C., & Shi, Y. (2001). *Swarm Intelligence*. Morgan Kaufmann.

Kennedy, J., & Spears, W. M. (1998). Matching algorithms to problems: an experimental test of the particle algorithms on the multimodal problem generator. In *Proc. Int. Conf. on Evolutionary Computation* (pp. 78-83).

Kesharaju, M., & Nagarajah, R. (2017). Particle Swarm Optimization Approach to Defect Detection in Armour Ceramics. *Ultrasonics*, *75*, 124–131. doi:10.1016/j.ultras.2016.07.008 PMID:27951501

Kora, P., & Sri Rama Krishna, K. (2016). Hybrid firefly and Particle Swarm Optimization algorithm for the detection of Bundle Branch Block. *International Journal of the Cardiovascular Academy*, *2*(1), 44–48. doi:10.1016/j.ijcac.2015.12.001

Laskari, E. C., Parsopoulos, K. E., & Vrahatis, M. N. (2002). Particle swarm optimization for minimax problems. In *Proceedings of the 2002 Congress on* Evolutionary Computation CEC '02, Honolulu, HI (pp. 1576-1581). doi:10.1109/CEC.2002.1004477

Li, L., Jiao, L., Zhao, J., Shang, R., & Gong, M. (2017). Quantum-behaved Discrete Multi-objective Particle Swarm Optimization for Complex Network Clustering. *Pattern Recognition*, *63*, 1–14. doi:10.1016/j.patcog.2016.09.013

Li, Z., Wang, W., Yan, Y., & Li, Z. (2015). PS–ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems. *Expert Systems with Applications*, *42*(22), 8881–8895. doi:10.1016/j.eswa.2015.07.043

Liu, B., Wang, L., Jin, Y.-H., Tang, F., & Huang, D.-X. (2005). Improved particle swarm optimization combined with chaos. *Chaos, Solitons, and Fractals*, *25*(5), 1261–1271. doi:10.1016/j.chaos.2004.11.095

Løvbjerg, M., Rasmussen, T., & Krink, T. (2001). Hybrid particle swarm optimizer with breeding and subpopulation. In *Proceedings of the third Genetic and Evolutionary Computation Conference* (Vol. 1, pp. 469-476).

Mandloi, M., & Bhatia, V. (2016). A low-complexity hybrid algorithm based on particle swarm and ant colony optimization for large-MIMO detection. *Expert Systems with Applications*, *50*, 66–74. doi:10.1016/j.eswa.2015.12.008

Marinaki, M., & Marinakis, Y. (2016). A Glowworm Swarm Optimization algorithm for the Vehicle Routing Problem with Stochastic Demands. *Expert Systems with Applications*, *46*, 145–163. doi:10.1016/j.eswa.2015.10.012

Meza, J., Espitia, H., Montenegro, C., Giménez, E., & González-Crespo, R. (2017). MOVPSO: Vortex Multi-Objective Particle Swarm Optimization. *Applied Soft Computing*, *52*, 1042–1057. doi:10.1016/j.asoc.2016.09.026

Miranda, V., & Fonseca, N. (2002). New evolutionary particle swarm algorithm (EPSO) applied to voltage/VAR control. In *Proceedings of the 14th Power Systems Computation Conference (PSCC'02)*, Seville, Spain.

Mirjalili, S. (2016). SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowledge-Based Systems*, *96*, 120–133. doi:10.1016/j.knosys.2015.12.022

Mirjalili, S., & Hashim, S. Z. M. (2010). A new hybrid PSOGSA algorithm for function optimization. In *Proceedings of the 2010 International Conference on Computer and Information Application*, Tianjin (pp. 374-377). doi:10.1109/ICCIA.2010.6141614

Mousa, A. A., & El-Shorbagy, M. A. (2012). Enhanced particle swarm optimization based local search for reactive power compensation problem. *Applications of Mathematics*, *3*, 1276–1284.

Mousa, A. A., El-Shorbagy, M. A., & Abd-El-Wahed, W. F. (2012). Local search based hybrid particle swarm optimization algorithm for multiobjective optimization. *Swarm and Evolutionary Computation*, *3*, 1–13. doi:10.1016/j.swevo.2011.11.005

Ozcan, E., & Mohan, C. K. (1998). Analysis of a simple particle swarm optimization system, In Intelligent Engineering Systems Through Artificial. *Neural Networks*, *8*, 253–258.

Parsopoulos, K. E., Tasoulis, D. K., & Vrahatis, M. N. (2004). Multiobjective optimization using parallel vector evaluated particle swarm optimization. *In Proceedings of the IASTED International Conference on Artificial Intelligence and Applications*, Innsbruck, Austria (Vol. 2, pp. 823–828).

Parsopoulos, K. E., & Vrahatis, M. N. (2002). Particle swarm optimization method for constrained optimization problems. Intelligent Technologies–Theory and Application. *New Trends in Intelligent Technologies*, *76*(1), 214–220.

Parsopoulos, K. E., & Vrahatis, M. N. (2002). Particle Swarm Optimization Method in Multiobjective Problems. In *Proceedings of the ACM 2002 Symposium on Applied Computing*, Madrid, Spain (pp. 603-607). doi:10.1145/508791.508907

Premalatha, K., & Natarajan, A. M. (2010). Hybrid PSO and GA models for Document Clustering. *International Journal of Advances in Soft Computing and Its Applications*, *2*(3), 302–320.

Qin, J., Yin, Y., & Ban, X. (2010). A Hybrid of Particle Swarm Optimization and Local Search for Multimodal Functions. In Y. Tan, Y. Shi, & K. C. Tan (Eds.), *Advances in Swarm Intelligence. ICSI 2010*. Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-13495-1_72

Qu, B. Y., Liang, J. J., & Suganthan, P. N. (2012). Niching particle swarm optimization with local search for multi-modal optimization. *Information Sciences*, *197*, 131–143. doi:10.1016/j.ins.2012.02.011

Ray, T., & Liew, K. M. (2001). A swarm with an effective information sharing mechanism for unconstrained and constrained single objective optimization problem. In *Proc. congress on evolutionary computation 2001*, Seoul, Korea (pp. 75-80).

Salman, A., Ahmad, I., & Al-Madani, S. (2002). Particle Swarm Optimization for Task Assignment Problem. *Journal of Microprocessors and Microsystems*, *26*(8), 363–371. doi:10.1016/S0141-9331(02)00053-4

Sethanan, K., & Neungmatcha, W. (2016). Multi-objective particle swarm optimization for mechanical harvester route planning of sugarcane field operations. *European Journal of Operational Research*, *252*(3), 969–984. doi:10.1016/j.ejor.2016.01.043

Shelokar, P. S., Siarry, P., Jayaraman, V. K., & Kulkarni, B. D. (2007). Particle swarm and ant colony algorithms hybridized for improved continuous optimization. *Applied Mathematics and Computation*, *188*(1), 129–142. doi:10.1016/j.amc.2006.09.098

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, Anchorage, AK (pp. 69-73). doi:10.1109/ICEC.1998.699146

Shi, Y., & Eberhart, R. C. (1998). Parameter Selection in Particle Swarm Optimization. In *Evolutionary Programming VII. LNCS*, *1447*, 591–600. doi:10.1007/BFb0040810

Shi, Y., & Eberhart, R. C. (1999). Empirical Study of Particle Swarm Optimization. In Proceedings of the 1999 Congress of Evolutionary Computation. IEEE Press. doi:10.1109/CEC.1999.785511

Shi, Y., & Krohling, R. A. (2002). Co-evolutionary particle swarm optimization to solve min-max problems. In *Proceedings of the 2002 Congress on* Evolutionary Computation CEC '02, Honolulu, HI (pp. 1682-1687).

Sierra, M. A., & Coello, C. C. (2006). Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the- Art. *International Journal of Computational Intelligence Research*, *2*(3), 287–308.

Suganthan, P. N. (1999). Particle swarm optimiser with neighbourhood operator. In P.J. Angeline, Z. Michalewicz, M. Schoenauer et al. (Eds.), *Proceedings of the Congress of Evolutionary Computation*, Mayflower Hotel, Washington D.C. (Vol. 3, pp. 1958-1962). doi:10.1109/CEC.1999.785514

Sun, C.-L., Zeng, J.-C., & Pan, J.-S. (2011). An improved vector particle swarm optimization for constrained optimization problems. *Information Sciences*, *181*(6), 1153–1163. doi:10.1016/j.ins.2010.11.033

Tan, D. (2012). Chaos Particle Swarm Optimization Algorithm for Multi-Objective Constrained Optimization Problems. In Y. Zhang (Ed.), *Future Wireless Networks and Information Systems. Lecture Notes in Electrical Engineering*. Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-27326-1_60

Van den Bergh, F. (2002). An Analysis of Particle Swarm Optimizers [Ph.D. thesis]. Department of Computer Science, University of Pretoria, Pretoria, South Africa.

Vesterstrom, J., & Riget, J. (2002). particle swarms – extensions for improved local, multi-modal, and dynamic search in numerical optimization [Master's thesis]. Department of computer science, University of Aarhus.

Yadav, A., & Deep, K. (2014). An efficient co-swarm particle swarm optimization for non-linear constrained optimization. *Journal of Computational Science*, *5*(2), 258–268. doi:10.1016/j.jocs.2013.05.011

Yang, S., Wang, M., & Jiao, L. (2004). A Quantum Particle Swarm Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC'04* (Vol. 1, pp. 320-324).

Zhao, B., & Cao, Y. (2005). Multiple objective particle swarm optimization technique for economic load dispatch. *Journal of Zhejiang University-SCIENCE A*, *6*(5), 420–427. doi:10.1631/jzus.2005.A0420

Zhou, Y., Chen, X., & Zhou, G. (2016). An improved monkey algorithm for a 01 knapsack problem. *Applied Soft Computing*, *38*, 817–830. doi:10.1016/j.asoc.2015.10.043

APPENDIX

Basic Definitions of PSO

In the following, some definitions of many terms that was used in PSO:

- **Swarm:** The algorithm population.
- **Particle:** The swarm member; where it represents the solution of the problem of optimization which we want to solve.
- **Local Best (lbest):** For a given particle, the best particle position in the neighborhood is called lbest.
- **Personal Best (pbest):** Personal best position of a given particle so far (i.e. the particle position that give the greatest success).
- **Global Best (gbest):** For all swarm, the best particle position is called gbest.
- **Leader:** The particle that guide other particles in the direction of better regions in the problem search space.
- **Velocity:** It is a vector used to drive the process of optimization; where It give the path of movement of the particle to evolve its current position.
- **Inertia Weight (w):** This parameter is used to decrease the effect of prior velocities on the current velocity and position of particle.
- **Learning Factor:** There are two learning factors c1 and c2 (they are defined as constants). c1 is the cognitive factor; where it represents the particle attraction in the direction of its own success, while c2 is the social factor; where it represents the particle attraction toward the success of its neighbors.
- **Neighborhood Topology:** Determine the particle's set that used to calculate the value of lbest for a given particle.