Evolutionary Intelligence (2019) 12:113–129
https://doi.org/10.1007/s12065-019-00210-z

REVIEW ARTICLE

A survey on particle swarm optimization with emphasis on engineering and network applications

Mohammed Elbes[1] · Shadi Alzubi[1] · Tarek Kanan[1] · Ala Al-Fuqaha[2] · Bilal Hawashin[3]

Received: 4 August 2018 / Revised: 5 November 2018 / Accepted: 4 February 2019 / Published online: 21 February 2019

Abstract

Swarm intelligence is a kind of artificial intelligence that is based on the collective behavior of the decentralized and self-organized systems. This work focuses on reviewing a heuristic global optimization method called particle swarm optimization (PSO). This includes the mathematical representation of PSO in contentious and binary spaces, the evolution and modifications of PSO over the last two decades. We also present a comprehensive taxonomy of heuristic-based optimization algorithms such as genetic algorithms, tabu search, simulated annealing, cross entropy and illustrate the advantages and disadvantages of these algorithms. Furthermore, we present the application of PSO on graphics processing unit and show various applications of PSO in networks.

Keywords Heuristic-based optimization · Particle swarm optimization · Taxonomy · PSO network applications

1 Introduction

Swarm intelligence is a kind of artificial intelligence and is based on the collective behavior of the decentralized or self-organized systems. These systems are modeled by a population of agents that share information with each other and interact with their environment. Although there is no centralized control that governs how these agents will interact, the local, and somehow random, interaction between these agents leads to global system intelligence. Examples of these systems are ant colonies [1], bird flocking [2], and fish schooling [3].

Particle swarm optimization (PSO) is a stochastic population-based optimization method proposed by Kennedy and Eberhart [4]. It has been successfully applied to solve many problems such as artificial neural network training [5, 6], fuzzy logic control [7, 8], and pattern classification [9, 10].

PSO has been broadly considered by many researchers in the last two decades due its simplicity, accuracy, and fast convergence [10]. Different aspects of the original version of PSO in 1995 have been proposed and implemented. Also, many researches and review articles have been published in the recent years to overview the applicability of PSO in different applications [11–18]. Some comprehensive survey articles on PSO has been published, but most of them are quiet old and specific to a certain application [19–22]. The number of these articles is increasing since PSO was launched in 1995, Fig. 1 [4] shows the amount of publications related to PSO since 1995.

The following section defines the optimization problem in general. The rest of the paper is organized as follows, Sect. 3 describes the taxonomy of optimization techniques, standardization of PSO is discussed in Sect. 4, PSO and binary PSO with the modifications to these algorithms are discussed in Sects. 5 and 6 respectively, Sect. 7 illustrates parallel PSO using GPU implementation, Sect. 8 present

✉ Mohammed Elbes
m.elbes@zuj.edu.jo

Shadi Alzubi
smalzubi@zuj.edu.jo

Tarek Kanan
tarek.kanan@zuj.edu.jo

Ala Al-Fuqaha
ala@ieee.org

Bilal Hawashin
b.hawashin@zuj.edu.jo

1 Department of Computer Science, Alzaytoonah University of Jordan, Amman, Jordan

2 Department of Computer Science, Western Michigan University, Kalamazoo, MI, USA

3 Department of Computer Information Systems, Alzaytoonah University of Jordan, Amman, Jordan

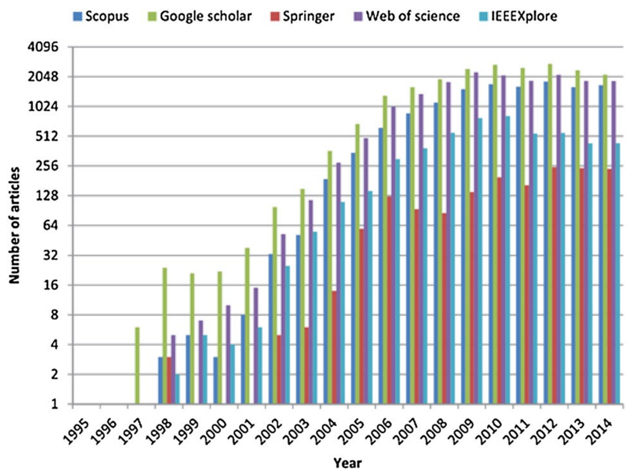

Fig. 1 Number of published articles regarding to PSO in the last two decades

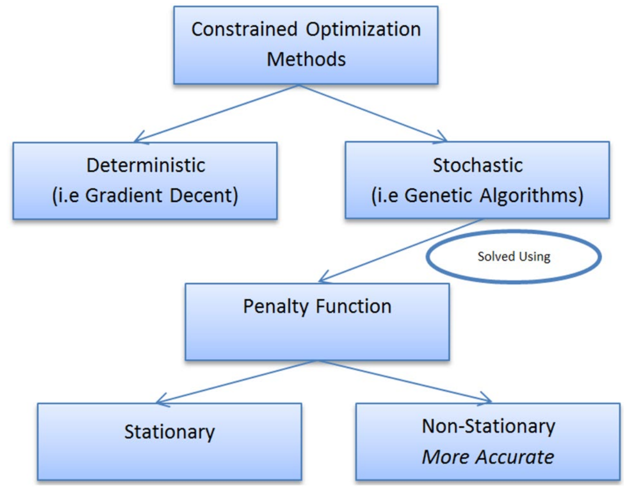

Fig. 2 Constrained optimization solution alternatives

examples of the PSO network and engineering applications, and the paper concludes in Sect. 9.

2 Optimization in mathematics and the penalty approach

Optimization refers to the study of minimizing or maximizing an objective function by finding the best values for its variables from within the permitted set of all values. Constrained optimization is the minimization of an objective function subject to the constraints on its variables. In general, an optimization problem can be represented as [23]:

$$\min f(x) \text{ such that } x \in S \subset R^n, \quad g_i(x) \le 0 \quad , i = \{1 \dots M\}, \tag{1}$$

According to the linear or nonlinear constraints $g_i(x) \le 0 \quad i = \{1 \dots M\}$

The formulation in Eq. (1) is not restrictive since the inequality can be represented as $-g_i(x) \ge 0$, and the constraint $g_i(x) = 0$ can be divided into two separate constraints $g_i(x) \le 0$ and $-g_i(x) \ge 0$.

Constrained optimization problems can be solved using two approaches as illustrated in Fig. 2. The deterministic approaches such as the gradient decent and feasible direction. Using such methods requires the objective function to be continuous and differentiable. Hence, the current research focuses on using stochastic methods such as genetic algorithms and other evolutionary programming algorithms.

The most popular way to solve constrained optimization problems is using a penalty function. The search space of the problem contains two types of points:

- Feasible points: are those who satisfy all the constraints of the problem
- Infeasible points: are those who violate at least one of the problem constraints.

The penalty approach solves an optimization problem by transforming it into a sequence of unconstrained problems by adding a penalty function of the constraints to the objective function. This approach penalizes those solutions that violate the problem constrains. Choosing high penalty values results in falling in a local minimum instead of a global minimum. On the other hand,if the penalty value is so small, the algorithm will hardly discover the feasible optimal solution [23].

The penalty approaches can be divided into two categories: stationary and non-stationary approaches. In the stationary approaches, a fixed value is added to the constraint when it violates the objective function. However, non-stationary approaches add a dynamically changing value for the penalty depending on how far the infeasible point from the constraint. The literature shows that the results obtained from non-stationary approaches are more accurate than those obtained using stationary approaches. The penalty function can be formulated as:

$$F(x) = f(x) + h(k)H(x) \quad x \in S \subset R^n, \tag{2}$$

where f (x) is the objective function in Eq. (1), h (k) is a dynamically changing penalty function, k is the current algorithm iteration and H (x) is a penalty factor defined as:

$$H(x) = \sum_{i=1}^{m} \Theta(q_i(x)) q_i(x)^{\gamma}(q_i(x)), \tag{3}$$

where $q_i(\mathrm{x}) = \max\{0, g_i(\mathrm{x})\}$, $\gamma(q_i(x))$ is the power of the penalty function, $\Theta(q_i(x))$ is a multi-stage assignment function and $g_i(x)$ are the constraints described in (1) (Table 1).

3 Taxonomy of optimization algorithms

In this section, we provide a detailed taxonomy of optimization. In general, optimization can be classified into three major categories [24]:

- **Optimization algorithms**: The majority of these algorithms are designed for linear programming. Examples of optimization algorithms are simplex algorithm of George Dantiz, extensions of the simplex algorithms designed for quadratic programming and combinatorial algorithms.
- **Iterative methods:**These methods are used for non-linear programming problems. Examples of these methods are the Newton's method, conjugate gradient method, interior point methods and the gradient descent method.
- **Heuristics:**In addition to the *finitely* terminating optimization algorithms and the *convergent* iterative method, heuristic approaches provide an approximate solution to optimization problems. Examples of these approaches are the genetic algorithms, tabu search, harmony search and particle swarm optimization.

The aforementioned and more approaches are presented in Fig. 5 at the end of this paper. Since the main topic of this paper is a heuristic-based approach (PSO), in this section we provide a detailed description for the major heuristic approaches. We also compare the performance of PSO and other heuristic approaches such as genetic algorithms (GA), tabu search (TS) , simulated annealing (SA), harmony search (HS), stochastic tunneling (ST) and the cross entropy (CE) method. Table 2 contains a summary of the advantages and disadvantages of these algorithms.

- **Genetic algorithms (GA):** A search method that mimics the biological evolution. Given a target problem, the input to the GA is a set of candidate solutions generated randomly and evaluated according to a fitness function. In general, most of these solutions do not survive the evaluation process and are killed instantly. Multiple randomly modified copies of the survivors are generated and a pool of new generations of candidate solutions is constructed. This process of generating, evaluating and modifying the best solutions is repeated for several hundreds or thousands of rounds until the algorithm converges to an acceptable solution [25–28].
- **Tabu search (TS):** The goal of TS is to prevent the search procedure from falling in a local optimum by keeping track of the search paths that already visited by the search procedure. This can lead the algorithm to accept some inferior solutions to avoid revisiting previous paths to find the best solution through more globalized search. The visited search paths (forbidden solutions) are stored in a tabu list which is maintained by a forbidding strategy that decides which solutions are candidates and to be kept in the tabu list [29–32].
- **Simulated annealing (SA):** This method mimics the process of crystallization from a melt. The atoms of a melt are free to move at high temperatures and when cooling the melt sample, these atoms start to crystallize into a solid. If the melt is cooled quickly, the melt becomes amorphous and if it is annealed (cooled) slowly the melts becomes a perfect crystal which is considered the global minimum energy configuration of the system. The goal is to find the best annealing schedule that converts the melt into a perfect crystal [33–36].
- **Harmony search (HS):** Another meta-heuristic optimization approach that mimics the musical improvisation. Each musician corresponds to a variable in the fitness function and the pitch range of each musical device corresponds to the range of values a variable can have. A candidate solution is represented by an improvised harmony. The more the musicians practice, the better harmonies created and correspondingly better solution vectors will be produced [37–41]. The algorithm works by randomly generating solution vectors (harmonies) and then disturbing these vectors according to HS algorithm to get the global minimum (best harmony).
- **Stochastic tunneling (ST):** A global optimization technique was originally proposed for minimizing the energy function in complex rugged potential energy surfaces (PES). In this method, the dynamical process explores a transferred adaptively changing version of the PES not the original one. The idea of this algorithm is to flatten the energy surface in all areas that have a value for energy above a certain threshold [42–45].
- **The cross entropy method (CE):** This is a new generic method used in rare event simulation and combinatorial optimization. The CE is an iterative procedure in which each iteration has two phases. The first phase is generating random data samples using a specified mechanism followed by the second phase of updating the parameters of the mechanism to produce better results in the next iteration. The power of the CE method stems from the fact that it produces a precise mathematical framework for deriving fast and optimal learning rules from the simulation theory. In this method, the deterministic optimization problem is transformed into a stochastic optimization problem and then the CE method is used to solve the problem [41, 46, 47].

Table 1 Summary of the modifications and improvements to the original PSO algorithm

	Modification	Modification effect	Space
1	The introduction of the inertia weight w in the velocity equation vid = w*v_{id} + $c_1 rand_1()*(p_{id} - x_{id}) + c_2*rand_2()\ (p_{gd} - x_{id})$	Setting w initially for a value greater than 1 allows for more exploration of the search space and then decreasing to allow for more detail exploration around the optimal value	Cont./binary
2	The introduction of the Constriction Factor X which is defined by: $X = \frac{2}{\lvert 2-\phi-\sqrt{\phi^2-4\phi}\rvert}$	When $\phi > 4$ the convergence of PSO is fast and guaranteed	Cont./binary
3	Choice of the number of particles to be from 20 to 100	Guarantees sufficient explorers to avoid falling in local optima and avoids unnecessary processing due to high number of particles	Cont./binary
4	The three basic logic rules introduced in [5]	Substantial performance gains compared to the canonical PSO, higher ability to locate the feasible solutions and found the optimal solution for the problem in hand	Cont.
5	$\Delta X_{id} = \Delta X_{id} + c_1 rand_1()(x_{id} - p_{id}) + c_2 rand_2()\ (x_{id} - p_{gd})\ x_{id}$ where p_{id} and p_{gd} are the particle and global worst positions respectively	The optimal solution was reached by guiding each member of the swarm to move away from its previous worst position and the group's worst position. The approach is considered a variant of the PSO and sometimes it outperforms the original PSO	Cont.
6	Scaling the velocity with the term (1-(t/T)h)v_{id} = (1 - (t/T)h) V_{max} if v_{id} >(1 − (t/T)h) V_{max} V_{id} = − (1−(t/T)h) V_{max} if v_{id} < − (1 − (t/T)h) V_{max}	Considerably better convergence performance than PSO within the given generations. Also added more control on the PSO which is achieved by introducing the parameter h. This parameter controls the reducing speed of the searching scale	Cont.
7	Velocity is defined as the rate of change in the bits of the particle. The previously found direction of change to one or to zero is maintained through the introduction of new vectors of velocity $V_1^{\rightarrow}$ and $V_0^{\rightarrow}$ for each particle. After these vectors are updated, the velocity change is obtained as in Eq. (13)	Guarantees the algorithm convergence	Binary
8	Position update formula If $0.5 - \delta < s(v_{i,j}^{k+1}) < 0.5 + \delta$ then $x_{i,j}^{k+1} = x_{i,j}^{k}$ If $s(v_{i,j}^{k+1}) < 0.5 - \delta$ then $x_{i,j}^{k+1} = 0$ $s(v_{i,j}^{k+1}) < 0.5 + \delta$ then $x_{i,j}^{k+1} = 1$	Allows each particle to keep its own inertia and prevents particles from moving to the same position and get trapped in a local optimum. It also provides better performance and has quick convergence abilities	Binary
9	Sigmoid function was changed to the form: SI (x) = $\frac{2}{1+e^{-\lvert x\rvert}} - 1$	The algorithm outperformed the original PSO in all runs but sometimes was outperformed by the (? + λ) evolutionary algorithm	Binary
10	position update equations : If ($0 < v_{iD} \leq a$), then $x_{iD}(new) = x_{iD}(new)$ if ($a < v_{iD} \leq \frac{1+a}{2}$), then $x_{iD}(new) = p_{iD}(new)$ If($\frac{1+a}{2} < v_{iD}$? 1), then $x_{iD}(new) = pg_{iD}(new)$	The modification was to choose the value for the next position of the particle such that 10% of the particles are forced to fly away to avoid falling in a local optima it also increased the effectiveness of the search algorithm	Binary
11	Inertia weight changes:w (t + 1) = 4.0*w(t)*(1− w(t)) Where w (t) in (0, 1)	Prevents the original PSO from getting immaturely trapped in a local optimum	Binary

Table 2 Advantages and disadvantages of the common search algorithms

Advantages	Disadvantages
Particle swarm optimization	
A derivative-free technique	Lacking somewhat of a solid mathematical foundation for analysis
Easy in its concept and coding implementation	Still having the problems of dependency on initial conditions, parameter values, difficulty in finding the optimal design parameters, stochastic characteristics of the final outputs
Less sensitive to the nature of the objective function	
Limited number of parameters. Also, less sensitive to parameters and fast convergence	
Less dependent on initial points	
Genetic algorithms	
No derivatives needed and easy to parallelize	Convergence is not guaranteed even to a local minima
Can escape local minima	Parameter space should be discrete
Problem linearization is not needed	Initial convergence is fast but gets much slower with time
Models with high probability get sampled more than models with lower probability	As other models with many parameters to tune, genetic algorithms are computationally expensive
	Convergence is very difficult when noise is available
Tabu search	
Always escapes a local minima	Not common to be implemented in the continuous space due to the difficulties of performing neighborhood movements in continuous search spaces
Very general and simpler that other optimization algorithms	When extended to work in the continuous space, it becomes more computationally expensive due to the local search method requirements
Easy to implement with no any space requirements	Algorithm design gets complex when optimizing multiple objective functions
TS is flexible and open to any other future improvements	
Simulated annealing	
Can deal with arbitrary systems and cost functions	Annealing repeatedly is very slow, especially if the cost function is expensive to compute.
Finding an optimal solution is statistically guaranteed	SA doesn't work well when the energy function is smooth or there are few local optima for the objective function
Easy implementation even for complex problems	Heuristic methods works better when providing problem-specific solution although sa is comparable to heuristic methods some times
In general it gives a good acceptable solution	
Harmony search	
HS requires less mathematics as it utilizes only a single search memory to evolve	The parameters of the HS have to be chosen carefully otherwise it will have poor performance and huge number of iterations to find the optimal solution.
Initial values for the decision variables are not needed and no need for derivatives	Weak local search ability
HS is more flexible and produces better solutions than GA	
Cross entropy	
Ease and flexibility of implementation	It is a generic method
Low overhead and robustness	Performance function should be relatively cheap
Does not require decomposition of the problem into a master problem and operation sub problems, which reduces computational complexity	Tweaking (modifications) may be required
Speed of convergence	
CE parameters need not be tweaked for each run	

A summary of advantages and disadvantages of adapting any of these algorithms in a certain problem is provided in Table 2.

4 Standardization of PSO

The production of computational intelligence by exploiting analogues of social interaction instead of individual cognitive abilities was the initial aim behind PSO in 1995. The first versions of PSO [48] were inspired by the work of Heppner and Grenander having analogues for flocks of birds searching for corn [49] which was then transformed into a more powerful optimization method which is the PSO.

The algorithm of the PSO can be written as follows [50]: neighborhood. In this approach, also known as the *gbest* model, each particle has a global knowledge of all particles of the swarm [52]. Using one of these models is application dependent. In general, the *lbest* approach has slow convergence rate unlike the *gbest* model which has high convergence rate that can trap the algorithm into a local minimum.

```
For each particle
  {
    Do
      Initialize particle
      Calculate the corresponding fitness value
      If the fitness value is better than the particle's best fitness value
          Set the current P vector to the particle's current X vector
  }
    Choose the particle with the lowest fitness value and make it the global
best position
  For each particle
  {
  Calculate the particle's velocity according to equation 9
  Update the particle current position vector X according to equation 10
    }
  } while maximum iteration or minimum error criteria is not attained
```

The algorithm starts by creating the swarm of particles assigning each particle with its parameters like its initial position. Then, the position of each particle is updated according to Eqs. (9, 10). In each iteration, the particle compares its current position with its own best position. If the current position is better than that position, the current position becomes the particle position. The particle with the best value for the fitness function is chosen to be the swarms best particle and the particles in the swarm tend to move toward this particle. The flow chart in Fig. 3 shows the PSO mode of operation.

The advantage of the PSO is that it does not require tuning many parameters in order to get acceptable performance [51]. Furthermore, it is applicable for both constrained and unconstrained problems and is easy to implement. On the other hand, the PSO could have premature convergence if the penalty values were high and might get trapped in a local optima. Also, the parameters are problem dependent and it is not trivial to find the best values for parameters.

In the original PSO, Euclidean neighborhood was used for information sharing between particles. This approach, also known as the *lbest* model, has high computational complexity which was the reason for replacing it with topological

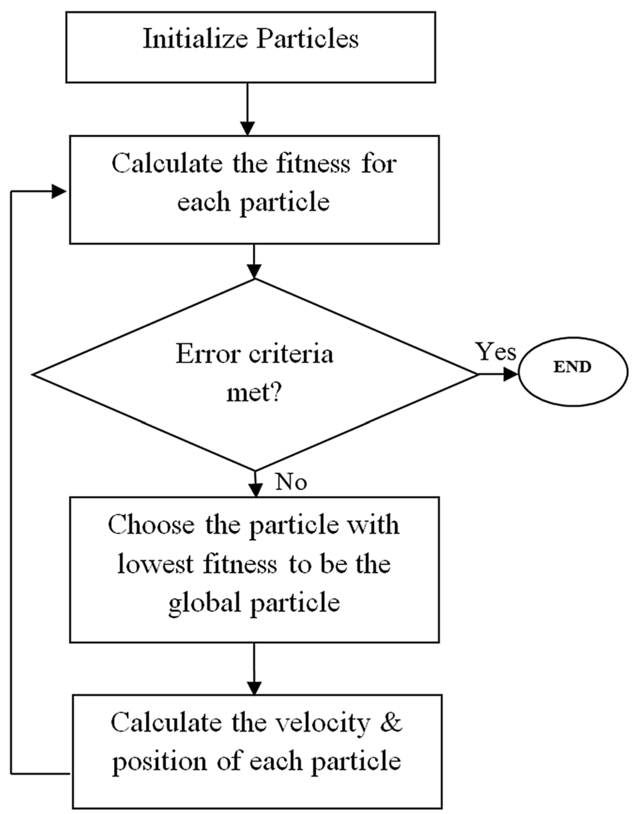

Fig. 3 PSO mode of operation

4.1 Inertia weight and constriction

In the original PSO, the velocity and position update equations are:

$$v_{id} = v_{id} + c_1 r_1 (p_{id} - x_{id)} + c_2 r_2 (p_{gd} - x_{id}), \tag{4}$$

where

$$\begin{aligned} &\mathrm{v}_{id} = \mathrm{V}_{max} \quad \text{if} \quad \mathrm{v}_{id} > \mathrm{V}_{max} \\ &\mathrm{v}_{id} = -\mathrm{V}_{ma}\mathrm{x} \quad \text{if} \quad v_{id} < -\mathrm{V}_{max}, \\ &x_{id} = x_{id} + v_{id} \end{aligned} \tag{5}$$

where c_1, c_2 are the cognitive and social parameters, r_1 and r_2 are random numbers uniformly distributed between 0 and 1. p_{id} is the particles best position and p_{gd} is the swarms best position.

In order to avoid large values for the velocities leading the swarm to explode after several iterations, the particles' velocities are clamped by setting a maximum velocity V_{max}. Having a fixed value for V_{max} is not applicable for all search spaces. Large search spaces require high values of V_{max} allowing for adequate exploration of the search space. This is not the case for small search spaces which require small values of V_{max} to avoid the swarm from exploding out of the feasible solution space. Incorrect choice of V_{max} can lead to poor performance. Hence, there is no precise method for choosing a value for V_{max} beyond trial and error. For this reason, the inertia weight w is introduced to replace V_{max} and Eq. (5) becomes:

$$v_{id} = w.v_{id} + c_1.r_1.(p_{id} - x_{id}) + c_2.r_2(p_{gd} - x_{id}). \tag{6}$$

The inertia weight has an initial value greater than 1 to allow for more exploration of the search space. The inertia weight then decreases reaching a value that allows for in-detail exploration around the global optimum. In [53], a study for the choice of the inertia weight value was carried out. The authors concluded that the use of the inertia weight for controlling the velocity results in a high efficiency of the PSO.

A carefully chosen value of the inertia weight provides a balance between the global exploration and local exploitation of the search space. Setting the inertia weight to a random value between [0, 1] is better than setting the inertia to a maximum value and linearly decrease this value until it reaches zero. The linear decrease of the inertia weight can trap the in local minima instead of a global one. Another method for this adaptive search in the search space was the introduction of the Constriction Factor X which is defined by [53]:

$$X = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}, \tag{7}$$

where $\phi = c_1 + c_2$.

When $\phi < 4$, the convergence around the best fitness value is slow but not guaranteed. However, when $\phi > 4$ the convergence is fast and is guaranteed. A typical value for x is around 0.7 which results when $c_1 = c_2 = 2$, and the velocity equation becomes:

$$V_{id} = X * [v_{id} + c_1.r_1(p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id})]. \tag{8}$$

4.2 Choice of the number of particles

The initial number of particles has a great impact on the performance of the optimization algorithm. Small number of particles can lead to insufficient explorers resulting in local optima. On the other hand, a high number of particles leads to unnecessary processing degrading the performance of the optimization algorithm. Empirical results show that a number of 50 particles provides good results with many objective functions. In general 20–100 particles are usually acceptable [52].

5 PSO in continuous space

PSO was first introduced in 1995 by Kennedy and Eberhart [54]. They developed methods for optimizing continuous nonlinear mathematical functions. The ideas of this algorithm were taken from artificial intelligence, social psychology and swarming theory [54]. The algorithm simulates swarms of animals searching for foods like fish schools and bird flocks. This algorithm represents an optimization problem by randomly created particles. These particles move in the solution space looking for the particle with best solution. The algorithm relies on the concept of information sharing between particles searching for the solution optima.

The ith particle in the swarm is represented by a D-dimensional vector $\mathrm{X}_i = (\mathrm{x}_{i1}, \mathrm{x}_{i2}, \ldots, \mathrm{x}_{iD})$, the particle's best position denoted as $\mathrm{P}_i = (\mathrm{p}_{i1}, \mathrm{p}_{i2}, \ldots, \mathrm{p}_{iD})$ and the particle' velocity $\mathrm{V}_i = (\mathrm{v}_{i1}, \mathrm{v}_{i2}, \ldots, \mathrm{v}_{iD})$. The movement of the particle in the search space is controlled by the following movement equations:

$$V_i^{k+1} = x[wV_i^k + c_1 r_{i1}^k (P_i^k - X_i^k) + c_2 r_{i2}^k (P_g^k - X_i^k)] \tag{9}$$

$$X_i^{k+1} = X_i^k + V_i^{k+1}, \tag{10}$$

where $i \in [1, N]$ and N is the size of the swarm.

The relative magnitude between $r_1.c_1$ and $r_2.c_2$ determines whether the particle moves towards p_{Best} or g_{Best}. If the upper bound of r1.c1 is greater than the upper bound of $r_2.c_2$, the particle tends to utilize the neighborhood experience more than its own experience.

5.1 Modifications to continuous PSO

Del valle et al. in [55] presented a modification to the original PSO in [54]. The modified PSO was applied in power system applications particularly in location and sizing of multiple STATCOM units in a power system. The goal was to find the best solution to improve the voltage profile of the power system at minimum cost. A comparison with the original PSO illustrated that the modification of velocity equations allowed the research process to be more efficient in finding the best feasible solutions. The enhancement was adding a basic logic to the particles to facilitate the search in the problem space. The logic was defined by the rules [55]:

- If the particle is not yet in the feasible space then its velocity is defined as:

$$v(t) = w_i v_i(t-1) + c.r.[pg - x_i(t-1)]. \tag{11}$$

This means that the particle should rely on its neighborhood to get into the feasible space rather than on its current position.

- If none of the particles in the swarm are in the feasible space, the maximum velocity of each particle is set to a random number so that the particles move erratically in the search space trying to find one feasible particle.
- If the particle local best and the swarm's global best are both feasible, then the original canonical PSO is applied.

Simulation results illustrated that the enhanced PSO suggested in [55] shown substantial performance gains compared to the canonical PSO, higher ability to locate the feasible solutions and it found the optimal solution for the problem in hand.

Swarm particles are urged to move away from their previous worst position and the group's worst position. The previous worst position of a particle is represented by $P = (p_{i1}, p_{i2}, ..., p_{iD})$, the swarm's worst position is g and the change in position is $\Delta X_i = (\Delta x_{i1}, \Delta x_{i2}, \dots, \Delta x_{iD})$. The velocity equations are similar to those of the original PSO:

$$\Delta X_{id} = \Delta X_{id} + c_1 r_1 (x_{id} - p_{id}) + c_2 r_2 (x_{id} - p_{gd}) \tag{12}$$

$$x_{id} = x_{id} + \Delta X_{id}. \tag{13}$$

This modification to the PSO was tested on benchmark functions (i.e. The sphere, Griewank, Rastrigrin and Rosebrock) that are used by many researchers. The results for specific settings of these functions were better than that of the PSO. The results also illustrated that this technique does not always outperforms the PSO and can be considered as a variation of the PSO instead of a better PSO algorithm.

An efficient speedup strategy based on introducing an adaptive scaling term into the original PSO was proposed in [56]. The velocity of the particle is equal to V_{max} multiplied by a scaling term $(1-(t/T)^h)$, where t is the number of current generation, T is the given maximum number of generations and h is a constant based on trial and error. The modified velocity equations become [56]:

$$v_{id} = (1-(t/T)^h)Vmax \quad if \quad vid > (1-(t/T)^h)Vmax \tag{14}$$

$$V_{id} = -(1-(t/T)^h)Vmax \quad if \quad vid < -(1-(t/T)^h)Vmax. \tag{15}$$

This scaling term $(1-(t/T)^h)$ allows the algorithm to evolve with a descending searching scale.

The modified PSO in [56] was tested on four well known benchmarks for PSO. The modification resulted in a considerably better convergence performance than the original PSO. The modified algorithm also added more control on the PSO by introducing the parameter h. This parameter controls the reducing speed of the searching scale.

The modified algorithm was applied to a five-hole aerodynamic probe calibration which is one of the typical modeling problems of aerodynamics and fluid engineering. PSO was applied to train multi-layered feed forward neural networks to calibrate the probe aerodynamic characters from the existing measured data [56]. The training process was repeated with ten independent runs for both the modified and the original PSO. The results illustrated that the modified PSO can get an accelerated convergence within the given generations when used to train the neural network.

6 Binary PSO

Many optimization problems that require a discrete ordering of discrete elements can be represented in the discrete or binary space. Scheduling and routing are examples of these problems. In the continuous space, the trajectories of particles are adjusted on the basis of information about each particle's previous best performance and the previous performance of its neighbors in the swarm. In the binary space, the trajectories are changed based on the probability that a certain coordinate will change to one or zero. In other words, a particle may be seen as moving nearer or father on the corners of a hypercube [48].

The velocity of the particle is defined as the hamming distance between the particle at time t and at time t + 1 in the next iteration. This is represented in terms of changes of probabilities that a certain bit will be one or zero. In general, v_{id}, will represent the probability that a bit will be equal to 1.

The PSO equations remain the same as Eq. (10, 11) except that p_{id} and x_{id} are integers in $\{0, 1\}$. Since the velocity is a probability in the [0, 1]. To achieve that, a logistic transformation $S(v_{id})$ is used to limit the velocity to be in [0, 1]. The transformation S can be the sigmoid function which is defined as:

$$S(v_{id}) = \frac{1}{1 + e^{-v_{id}}}. \tag{16}$$

The resulting position change is defined based on the rule

$$x_{id} = \begin{cases} 1 & R < S(v_{id}) \\ 0 & R \geq S(vid), \end{cases} \tag{17}$$

where R is uniformly distributed random number between [0, 1].

While a higher value of V_{max} in the continuous space allows for more exploration of the search space, a lower value of V_{max} is required initially to explore the search space. This value should be decreased gradually when the algorithm is about to converge to the optimal solution.

6.1 Runtime analysis of binary PSO

The analysis of the runtime for the evolutionary PSO algorithm is getting more attention in the recent years. This kind of analysis is usually hard because the underlying probabilistic model of the swarm algorithm usually depends on a history of past solutions [57]. In 1997, Kennedy and Eberhart [48] introduced a binary version of the classical PSO.

Carsten et al. in [57] performed a study of the runtime of the binary PSO. The study presented some lower bounds for a broad class of implementations for swarms that have a polynomial size. The upper bounds of the execution time were proved by transferring a fitness-level argument for evolutionary algorithms to PSO. This was then applied to find an estimate for the execution time on the class of unimodal functions [57].

The lower bounds analysis of v_{max} illustrated that setting v_{max} to a constant value leads to better results only for problems with bounded sizes. However, this leads to dramatic performance decrease for problems with variable size. The authors proved that the probability that PSO finds the global optimal value when having at most 2^K global optima is equal to 2^{-K} where K is a positive constant [57]. The upper bound of the binary PSO was found by setting the cognitive constant $c_1 = 0$. This means that each particle follows the leader of the swarm and ignores its best solution.

6.2 Modifications to binary PSO

Mojtaba et al. in [58] studied the shortcomings of the original binary PSO proposed in [48]. Their work illustrated that no clear choice was made for the value of the inertia weight in the original binary PSO.

The proposed algorithm in [58] interpreted the velocity in a different way than in [48]. The velocity is defined as the rate of change in the bits of the particle. The direction of change to one or to zero is maintained through the introduction of new vectors of velocity $V_1^{\rightarrow}$ and $V_0^{\rightarrow}$ for each particle. After these vectors are updated, the velocity change is obtained as in Eq. (18). The previous direction and previous state of each particle is also taken into account and provided better solutions.

Another modification to the original binary PSO was introduced in [59]. In this work, the authors provided a better solution to the partner selection problem which is a critical issue in the research of the virtual enterprise. After presenting the optimization model, an improved version of the binary PSO was designed to decrease the probability of the algorithm falling into a local optimum and to enhance the search ability. The modification targeted the particle position equation such that the particle velocity is divided into three regions. The state of the particle being one, zero or unchanged is determined by the particle's current region. The ranges of these regions change adaptively as the algorithm iterates to achieve global convergence. The velocity equation remains the same as in the original binary PSO, but the position equations became [59]:

$$IF \quad 0.5 - \delta \leq s(v_{i,j}^{k+1}) < 0.5 + \delta \quad then \quad x_{i,j}^{k+1} = x_{i,j}^{k} \tag{18}$$

$$IF \quad s(v_{i,j}^{k+1}) \leq 0.5 - \delta \quad then \quad x_{i,j}^{k+1} \tag{19}$$

$$IF \quad (v_{i,j}^{k+1}) \leq 0.5 + \delta \quad then \quad x_{i,j}^{k+1}. \tag{20}$$

The value of δ is initially 0.5 and decreases gradually in each iteration. Equation (18) allows each particle to keep its own inertia and prevents particles from moving to the same position falling in a local optimum. The simulation results in [59] illustrated that the proposed algorithm provides better performance and has quick convergence abilities.

Hereford et al. in [60] addressed a special type of optimization problems when the solution is a set of integers in the discrete space. The target problem was the Sudoku puzzle and the modified PSO is called Integer PSO (IPSO). This algorithm uses the same velocity update equation as in the original binary PSO. The only difference is that instead of having one velocity in the N-dimensional search space, a separate velocity value for each variable is utilized. This means that each particle has N-dimensional velocity vector and each of the variables' velocity vectors is updated separately. The velocity vector is scaled by a modified version of the sigmoid function to a get a value between 0 and 1. The goal of this modification was to change the sigmoid function to get high probabilities for large positive and negative velocities. Hence, the sigmoid function was changed to [60]:

$$SI(x) = \frac{2}{1 + e^{-|x|}} - 1. \tag{21}$$

The proposed IPSO in [60] was compared with two other algorithms, the ($\mu + \lambda$) evolutionary algorithm, and the original PSO algorithm. The IPSO was tested for 50 Sudoku puzzles of different levels of challenge. The algorithm outperformed the original PSO in all runs but was outperformed by the (μ+λ) Evolutionary algorithm in some cases.

Another modification to the original PSO was carried out in [61]. The goal was to design an algorithm for gene selection and tumor classification. The modification was updating the next position such that 10% of the particles are forced to move away from the g_{best} to avoid falling in local optima. The new suggested position update equations were:

$$IF \quad (0 < v_{iD} \leq a) \quad then \quad x_{iD}(new) = x_{iD}(new) \tag{22}$$

$$IF \quad \left(a < v_{iD} \leq \frac{1+a}{2}\right) \quad then \quad x_{iD}(new) = p_{iD}(new) \tag{23}$$

$$IF \quad \left(\frac{1+a}{2} < v_{iD} \leq 1\right) \quad then \quad x_{iD}(new) = p_{gD}(new). \tag{24}$$

The experimental results in [62] illustrated that this modification increased the effectiveness of the search algorithm and presented a useful tool selecting marker gene subset and mining high dimensional data.

The authors in [57] suggested an inertia weight equation that prevents the original PSO from getting immaturely trapped in a local optimum. The approach followed the evolutionary approaches to find a near optimal solution like genetic algorithms [63], Ant colony optimization [64] and tabu search [65]. The goal was to find a combined approach of the binary PSO with the K-nearest neighbor algorithm for feature selection using logistic map. The suggested algorithm was called the chaotic binary PSO (CBPSO). Chaos is a deterministic dynamic system which is sensitive to initial values. A chaotic map is used to determine the value of the inertia weight in ach iteration. The suggested inertia weight equation that guarantees the global optimum results is:

$$w(t+1) = 4.0 \times w(t) \times (1 - w(t)) \quad Where \quad w(t) \in (0, 1). \tag{25}$$

The experimental results in [57] illustrated that the new suggested inertia weight saves processing time compared to other methods in the literature. The results also revealed that the CBPSO with chaotic sequences reduces the number of features and achieves higher classification accuracy.

7 Parallel PSO using GPU implementation

The tremendous power of graphics processing unit (GPU) computing relative to prior CPU-only architectures presents new opportunities for efficient solutions of previously intractable large-scale optimization problems. To move beyond non-graphical applications and into general purpose parallel programming, NVIDIA, a computer games company, introduced the CUDA model which enables programmers to write their own code using a standard programming language like C with NVIDIA extensions. We refer the reader to [66] for more details on CUDA-based GPU.

Several limitations were identified in standard PSO in recent years. To overcome these limitations, PSO is implemented on GPU in many researches [67–71]. Addressing these limitations is very important because as long as PSO runs in real time, the performance of PSO will be transferable to a wide variety of optimization problems [72]. Significant amount of research utilized the usage of GPU for swarm intelligence inspired optimization algorithms taking advantage of the parallel processing power provided by the GPU [14, 67, 72–81]. Jaspreet Kaur et al claimed in [67] that simulation results shown that the parallel implementation of PSO overcomes the serial PSO algorithm considering the processing time.

In this section, we will present a generic PSO GPU/CUDA solution using Single Instruction Multiple Threading (SIMT) to solve large-scale optimization problems with time efficient compared with CPU implementation. To be able to achieve high performance, our mapping of PSO to GPU architectures, carefully follows these steps:

- Map the tasks into multiple threads.
- Manage access to global memory to guarantee coalesced memory access.
- Manage access to shared memory.
- Manage thread synchronization.

7.1 Swarm structure in CUDA

Since there are several types of memory available on the GPU device, it is important to choose the appropriate memory type to use for each of the models constituent elements. For PSO, positions and velocity variables are stored using a

```
typedef struct
{
  int X [N*popSize];
  float V[N*popSize];
  int p[N*popSize];
  int g[N];
  float  bestSwarmFitness;
}CUDA_SWARM;
//X: Particle position Vector
//V: Velocity vector for variable X
//N: Number of variables
//p: Best local Vector
//g: Best global Vector
```

Fig. 4 Swarm structure in CUDA

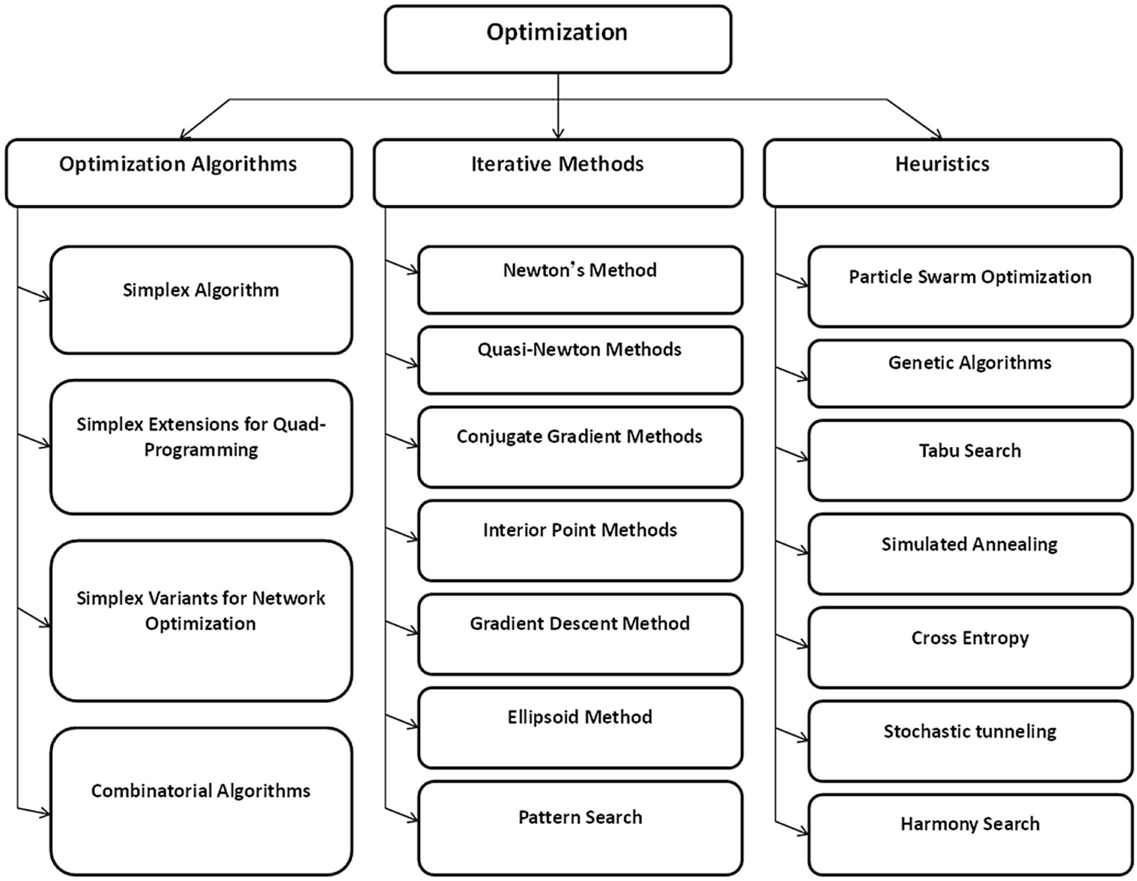

Fig. 5 Taxonomy of optimization

struct in CUDA. As depicted in Fig. 4, a single array stores all variables of the same type for all particles. In the memory coalescing next subsection , we describe the reason for such organization. Because shared memory is fast and can store modest amounts of data it is used it to store the fitness value, best local fitness value, and random number generator seed for each particle. In addition the design utilizes the efficient instruction sequences that ensure concurrent tasks among all threads (Fig. 5).

7.2 PSO coalesced memory access

The manner and frequency with which global memory is accessed has major performance implications for CUDA GPU implementations. Because of this, PSO velocity and position variables are carefully structured in global memory in order to guarantee coalesced memory access. More specifically, instead of storing the variables associated with each particle one after another, the values of each variable are grouped together for all particles. If the size of each variable is 4 bytes (float or integer), this structure ensures global memory coalescing 16 threads (a half-warp) access 64 bytes of memory simultaneously in one transaction [66].

7.3 PSO shared memory

In CUDA-enabled GPU devices, accessing to shared memory by threads in the same warp is as fast as accessing a register, provided that we guarantee there is no bank conflict between the threads [66]. Despite of its size limitations (16 kb) on each multiprocessor, the authors in [66] were able to use shared memory to store the values of particle fitness, best local particle fitness, and seeds used in random number generator. With this approach, it is guaranteed that no shared memory bank conflict can arise, since each thread is accessing its own variable that is located in a unique shared memory address. The schematic depicting the software/memory architecture is illustrated in Fig. 4. Notice that every particle has one unique initial seed to generate its random numbers.

8 PSO network and wireless network applications

PSO is applied in almost all disciplines of engineering. In this section we present the usage of PSO in network related applications. Sun et al. in [82] used PSO to solve the problem of selecting the neighboring peers in a P2P network. In this kind of problems, performance and search efficiency are highly influenced by the network topology. Many P2P systems were built not only to share multimedia files between users, but also for the public welfare such as supplying a processing power to fight cancer. Any P2P system consists of peers and some connection between them. The key point for having efficient and high performance system is to define how these peers are connected.

Finding such a definition for these connections is challenging because of the dynamic membership of the peers in the network. Hence, a continuing reorganization of the network topology is required all the time [82]. An efficient strategy for neighbor selection was proposed in [83].

Another application of PSO in networks was performed by Papagianni et al. [84]. The goal was to study the usage of PSO in the design of a network infrastructure including decisions concerning the locations and sizes of the links. A secondary goal was to address the quality of service issue in the design process. The optimization aimed to minimize the network layout cost and the average packet delay in the network [84].

The network design problem is considered NP-Hard so it was targeted using meta-heuristic techniques such as simulated annealing, tabu search and evolutionary computing [85, 86]. The solution to this problem is usually not optimal since it involves the optimization of several contradicting objectives such as network deployment, average delay or throughput subject to constraints like bandwidth and reliability [84].

An efficient solution to the shortest path problem (SPP) in networks was introduced by Ammar and Nirod in [87]. The shortest path computation is one of the major problems in graph theory and finding a polynomial solution for such problem is known to be impossible. Finding a feasible solution for this problem is considered the basis for many applications ranging from routing in communications networks to robot motion planning, sequence alignment in molecular biology to length limited Huffman coding and many other applications [87].

The SPP was targeted by many approaches like artificial neural networks (ANN) [88], tabu search [89] and genetic algorithms [90]. ANN weren't used on large scale for this kind of problem because the hardware complexity increases with the size of the network and ANNs do not provide suboptimal path like the meta-heuristic approaches. This was the reason for moving toward the evolutionary approaches like GA and TS. These two approaches gave better results than the ANN but when compared to PSO, the later outperformed them in terms of computation complexity, success rate and solution quality [87].

Along with PSO which was used in [87] to solve the SPP, additional noising mechanisms [91] were used to improve the local search quality around any local solution by using a diversification mechanism to discover near optimal points. The main issue in using PSO in SPP is the way a particle is encoded. A representation scheme called cost-priority-based was used in [49] to encode the particle based on node priorities. The goal was to compare the performance of PSO with two variants of GA based approaches. The results illustrated that PSO outperforms the GA for all configurations of the network in terms of speed and solution quality [87].

A modified version of PSO called trained PSO (TPSO) was presented by Shahin et al. in [92]. This approach distributed particles to reduce traffic and computational overhead in the optimization process and was applied in an ad-hoc network. The goal of the optimization was to find the node with the highest processing load in the network. To reduce the overhead of particles moving across the ad-hoc network, the original PSO was improved by changing the parameters of PSO (w and P) according to the requirements of the system using a training system [92]. The simulation results in [92] illustrated that the convergence time of TPSO is almost constant while the traffic and computational complexity over a node is reduced in comparison to PSO.

PSO was used by Alfawair et al. in [93] to design an algorithm for power consumption minimization in ad-hoc networks. In this type of networks, the power supply for the mobile node is limited by the capacity of the batteries. Hence it is necessary to develop a power minimization algorithm for throughput enhancement to determine neighbor selection and power level assignment [93]. A solution to this problem is achieved by scheduling the node to enter into a sleep mode which decreases the total power consumption. Another approach called power control [94, 95] reduces the power consumption by adjusting the power level for each frame to be sent based on the perceived ad-hoc network status. The simulation results carried out in [93] illustrated that the suggested power saving algorithm based on PSO outperform all other existing algorithms.

Diptam et al. in [96] proposed an anomaly detection system based on data mining to detect volume anomalies by monitoring Simple Network Management Protocol (SNMP). They combined Digital Signature of Network Segments (DSNS) with the particle swarm optimization heuristic and neural network training on real datasets. Their proposed approach uses SVM to cluster traffic collected by SNMP and DSNS. The authors then combine SVM with PSO to improve the performance of finding the centroids of the clusters.

Shing-Han et al. in [97] combined both K-mean and PSO to remedy the Botnet problem in which attackers launch denial of services attacks paralyzing large-scale websites and steal confidential data from infected computers. Their approach starts by discovering some kinds of network behaviors like long connections, connection failure and network scanning. Features from these behaviors are then extracted by scanning network traffic in the network and transport layers. PSO and K-means are then used to detect Botnet hosts in the organizational network.

Pryiad et al. [98] studied the problem of high mobility in MANETs which drastically affects the performance of the routing protocol and the network lifetime. Their approach utilized network parameters like energy drain rate and relative mobility estimation to predict network lifetime. To further improve the performance, the authors implemented a new algorithm by integrating network lifetime with PSO. This integration resulted in reduction of both network delay and congestion due to the obviation of the centralized control.

A soft fault diagnoses model for wireless sensor networks using PSO was proposed Rakesh et al. [99]. The proposed approach uses three phases for fault diagnosis; namely, initialization, fault identification and fault classification. The approach then uses the analysis of variance method to identify faulty nodes in the network. Furthermore, the feed-forward neural network along with PSO are used to classify the faulty node in the network.

In cloud data center OpenStack proved to not achieving the control of the physical network. Kai et al. proposed a multi-tenant virtual network to provide the user flexible control of the data center network. They developed a Virtual Software Defined Networking VSDN in the cloud. Their results proved that using PSO algorithm improved the utilization rate of the network bandwidth compared with the SP algorithm [100].

Many algorithms existed to help finding optimal solutions for sending data from node to node in wireless sensor networks. Lakshmanan and Tomar have combined PSO with genetic algorithm. They optimized the localization by applying their proposed methods then their simulation showed great achievement in evaluating and validating their methods. They provided better results than the general PSO and GA algorithms when working separately [101].

Vimalarani et al. in [102] presented an approach to improve PSO-Based Clustering Energy Optimization (EPSO-CEO) algorithm for wireless sensor networks by using PSO algorithm. They built a simulator to test their outcomes. The simulation showed that the (ECPSO-CEO) technique improved the consumed energy and increase the lifetime of WSN.

Long et, al developed a new approach to maintain Wireless Sensor networks by using particle swarm optimization. Their approach built over three aspects: coverage rate, node energy consumption, and node residual energy. To moderate the computational density they used weight particle swarm optimization. Their simulation showed that using the PSO-based maintenance strategy (PSOMS) is superior the random and uniform redeploy scheme [103].

Recently, the localization techniques have had more thoughtfulness by researcher of wireless sensor networks. Wuling and Cuiwen have presented in their research a new localization technique that depend on Shuffled Frog Leaping Algorithm (SFLA) and particle swarm optimization (PSO). Their simulation results showed that their proposed algorithm has very good accuracy results [104].

Keun-Chang has focused on the theme of optimizing and designing the GN (Granlar networks) depending on PSO and information granulation [105]. The linguistic model using context-based fuzzy c-means clustering algorithm has been employed by Kwak to design the GN [106]. To find the number of clusters in each context, the author took into account two-sided Gaussian membership functions and weighting factor using PSO in optimization stage to compute the localization information of the particles.

Harkirat and Shivani in [107] proposed an algorithm for path re-routing based on hybrid ACO–PSO routing algorithm for MANETs. This technique was proposed to improve fault tolerance in mobile ad-hoc networks to enhance network performance. The proposed algorithm in [107] implemented using MATLAB toolbox with the assistant of data analysis toolbox. A contrast was done between the existing and the proposed technique resulting that the performance of the mobile ad-hoc network was enhanced using the proposed algorithm by throttling the power consumption and minimizing end-to-end delay [107].

Israa et al. proposed an approach in [108] that enhanced PSO for optimal network reconfiguration in smart networks. The goal in hand was to provision restoration for a maximum number of customers with minimum power loss, while in the other hand satisfying operational constraints. The authors pre-defined all the possible fault locations in branches. The proposed networks were tested simultaneously on IEEE 33-bus in various configurations under different branch fault locations.Their results shown that the Network configuration supplies both a satisfaction level of reliability in most fault cases and vast capabilities to further improve the process efficiency.

Michal et al. in [109] proposed a population reduction method using knowledge collected from shortest path analysis in communication networks.Their work scouts the attributes of the shortest path in communication network produced by the PSO algorithm which by recording the inner communication of PSO. According to the collected answers, the results demonstrated firstly that the shortest path in the communication network of PSO did not contain all particles that follows a normal-like distribution around the center at different values for different benchmark functions (fitness landscapes). Secondly, results presented that there was no clear correlation between particles number on the shortest path and final solution quality [109].

Rui et al. in [110] proposed PSO-Forwarding Information Base (PSO-FIB) algorithm that forwards experiences of particles to find the forwarding probability for each entry in the FIB and demonstrated the interaction of PSO-FIB with the routing layer. The authors used the improved Salma algorithm in [111] to create a network topology that have similar nodes to represent the link metrics. Their PSO approach was compared with ant colony optimization strategies [110]. Their results proven that the average cost of PSO-FIB was also less than those of ACO and PSO which means that the paths average performance searched by PSO-FIB was better than ACO paths average performance.

9 Conclusion

In this paper, the evolution of the particle swarm optimization (PSO) technique was presented in detail. The changes to the PSO in both the continuous and discrete spaces were discussed. A summary of the modifications in both the continuous and binary space is presented in Table 1. This work also presented the methods and procedures that aimed to find a standard for the PSO that can be adopted for many engineering and other applications.

The paper also presented the upper and lower bounds for the execution time of the PSO algorithm. Furthermore, the changes to the binary and continuous PSO is detailed listing the advantages and disadvantages of these changes to the original PSO which was presented in [10]. By considering these changes, it can be obliviously seen that the parameters of the PSO are application dependent and PSO was always need to be changed to be applied for a specific application.

References

1. Dorigo M, Blum C (2005) Ant colony optimization theory: a survey. Theor Comput Sci 344(2):243–278
2. Virágh C, Vásárhelyi G, Tarcai N, Szörényi T, Somorjai G, Nepusz T, Vicsek T (2014) Flocking algorithm for autonomous flying robots. Bioinspir Biomim 9(2):025012
3. Zhang Y, Agarwal P, Bhatnagar V, Balochian S, Yan J (2013) Swarm intelligence and its applications. Sci World J. https://doi.org/10.1155/2013/528069
4. Bonyadi MR, Michalewicz Z (2017) Particle swarm optimization for single objective continuous space problems: a review. Evol Comput 25(1):1–54. https://doi.org/10.1162/EVCO_r_00180
5. Garro BA, Vázquez RA (2015) Designing artificial neural networks using particle swarm optimization algorithms. Comput Intell Neurosci 2015:61
6. Chen X, Li Y (2006) Neural network training using stochastic PSO. In: International conference on neural information processing. Springer, pp 1051–1060
7. Borni A, Abdelkrim T, Zaghba L, Bouchakour A, Lakhdari A, Zarour L (2017) Fuzzy logic, PSO based fuzzy logic algorithm and current controls comparative for grid-connected hybrid system. In: AIP conference proceedings, vol 1814, AIP Publishing, p 020006
8. Bachache NK, Wen J (2013) Design fuzzy logic controller by particle swarm optimization for wind turbine. In: Ying T, Yuhui S, Hongwei M (eds) Advances in swarm intelligence. Springer, Berlin, pp 152–159
9. Engelbrecht AP (2013) Particle swarm optimization: global best or local best? In: Proceedings of the 2013 BRICS congress on computational intelligence and 11th Brazilian congress on computational intelligence, IEEE computer society, pp 124–135
10. Poli R (2008) Analysis of the publications on the applications of particle swarm optimisation. J Artif Evol Appl. https://doi.org/10.1155/2008/685175
11. Banks A, Vincent J, Anyakoha C (2007) A review of particle swarm optimization. Part I: background and development. Nat Comput 6(4):467–484
12. Elbes M, Al-Fuqaha A, Rayes A (2012) Gyroscope drift correction based on TDoA technology in support of pedestrian dead reckoning. In: Globecom workshops (GC Wkshps), 2012 IEEE, pp 314–319
13. Banks A, Vincent J, Anyakoha C (2008) A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. Nat Comput 7(1):109–124
14. AlFuqaha A, Elbes M, Rayes A (2013) An intelligent data fusion technique based on the particle filter to perform precise outdoor localization. Int J Pervasive Comput Commun 9(2):163–183. https://doi.org/10.1108/IJPCC-02-2013-0001
15. Al-Fuqaha A, Kountanis D, Cooke S, Elbes M, Zhang J (2010) A genetic approach for trajectory planning in non-autonomous mobile ad-hoc networks with QOS requirements. In: GLOBECOM workshops (GC Wkshps), 2010 IEEE, pp 1097–1102
16. Temür R, Sait TY, Toklu YC (2015) Geometrically nonlinear analysis of trusses using particle swarm optimization. Recent advances in swarm intelligence and evolutionary computation. Springer, Berlin, pp 283–300
17. Elbes M, Al-Fuqaha A (2013) Design of a social collaboration and precise localization services for the blind and visually impaired. Proced Comput Sci 21:282–291
18. Li Y, Zhan Z-H, Lin S, Zhang J, Luo X (2015) Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems. Inf Sci 293:370–382
19. Zhang Y, Wang S, Ji G (2015) A comprehensive survey on particle swarm optimization algorithm and its applications. Math Probl Eng. https://doi.org/10.1155/2015/931256
20. Pi Q ,Ye H (2015) Survey of particle swarm optimization algorithm and its applications in antenna circuit. In: 2015 IEEE international conference on communication problem-solving (ICCP), pp 492–495
21. Yang B, Chen Y, Zhao Z (2007) Survey on applications of particle swarm optimization in electric power systems. In: 2007 IEEE international conference on control and automation, pp 481–486
22. Keisuke K (2009) Particle swarm optimization—a survey. IEICE Trans Inf Syst 92(7):1354–1361
23. Vrahatis M, Parsopoulos K (2002) Particle swarm optimization method for constrained optimization problems. Front Artif Intell Appl 76:215–20
24. Carlos E, Alexander M, Roberto S, Lozano Jose A (2013) On the taxonomy of optimization problems under estimation of distribution algorithms. Evolut Comput 21(3):471–495
25. Jacobson L, Kanber B (2015) Genetic algorithms in Java basics. Springer, Berlin
26. Moorkamp M (2005) Genetic algorithms: a step by step tutorial. Dublin Institute for Advanced Studies, Barcelona

27. Parker PB (1999) Genetic algorithms and their use in geophysical problems. Technical report, Lawrence Berkeley National Lab. (LBNL), Berkeley, CA (United States)
28. Wang Y (2018) Improved OTSU and adaptive genetic algorithm for infrared image segmentation. In: 2018 Chinese control and decision conference (CCDC), IEEE, 2018
29. Pham D, Karaboga D (2012) Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks. Springer Science & Business Media, New York
30. Ke Q, Jiang T, De MS (1997) A tabu search method for geometric primitive extraction 1. Pattern Recognit Lett 18(14):1443–1451
31. Lamont G, Coello C, Van Veldhuizen D (2002) Evolutionary algorithms for solving multi-objective problems. Springer, New York
32. Siarry P, Berthiau G (1997) Fitting of tabu search to optimize functions of continuous variables. Int J Numer Methods Eng 40(13):2449–2457
33. Kirkpatrick S, Gelatt C, Vecchi MP (1993) Optimization by simulated annealing. Science 220:671
34. Koziel S, Rojas AL, Moskwa S (2018) Power loss reduction through distribution network reconfiguration using feasibility-preserving simulated annealing. In: 2018 19th International scientific conference on electric power engineering (EPE). IEEE
35. Breno de ARA, Niraldo RF (2018) Simulated annealing and tabu search applied on network reconfiguration in distribution systems. In: 2018 Simposio Brasileiro de Sistemas Eletricos (SBSE). IEEE, 2018
36. Ma R, Wang Y, Hu W, Zhu X, Zhang K (2018) Optimum design of multistage half-band fir filter for audio conversion using a simulated annealing algorithm. In: 2018 13th IEEE conference on industrial electronics and applications (ICIEA). IEEE
37. Geem Z, Hwangbo H (2006) Application of harmony search to multi-objective optimization for satellite heat pipe design. Master's thesis
38. Woo GZ, Hoon KJ, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68
39. Lee K, Geem Z (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Methods Appl Mech Eng 194:3902–3933
40. Mahdavi M, Fesangharyb M, Damangirb E (2007) An improved harmony search algorithm for solving optimization problems. Appl Math Comput 188(2):1567–1579
41. Heegaard P, Wittner O, Helvik B, Nicola V (2004) Distributed asynchronous algorithm for cross-entropy-based combinatorial optimization. Rare event simulation and combinatorial optimization (RESIM/COP), Budapest, Hungary, 2004
42. Schug A, Herges T, Wenzel W (2003) Reproducible protein folding with the stochastic tunneling method. Phys Rev Lett 91(15):2–10
43. Mayer BE, Hamacher K (2014) Stochastic tunneling transformation during selection in genetic algorithm. In: Proceedings of the 2014 annual conference on genetic and evolutionary computation, GECCO '14, New York, NY, USA, 2014, ACM, pp 801–806
44. Hamacher K (2013) A new hybrid metaheuristic—combining stochastic tunneling and energy landscape paving. In: María JB, Christian B, Paola F, Andrea R, Michael S (eds) Hybrid metaheuristics. Springer, Berlin, pp 107–117
45. Wenzel W, Hamacher K (1999) Stochastic tunneling approach for global minimization of complex potential energy landscapes. Phys Rev Lett 82(15):3003
46. De Boer P, Kroese P, Mannor S, Rubinstein R (2004) A tutorial on the cross-entropy method. Ann Oper Res 134(1):254–5330
47. Rubinstein RY, Kroese DP (2004) The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning. Springer, New York. https://doi.org/10.1007/978-1-4757-4321-0 **(ISBN: 978-1-4757-4321-0)**
48. Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: 1997 IEEE international conference on systems, man, and cybernetics. Computational cybernetics and simulation, vol 5, pp 4104–4108
49. Heppner F, Grenander U (1990) A stochastic nonlinear model for coordinate bird flocks. Ubiquity Chaos 233:238
50. Hu X, Eberhart RC (2006) Solving constrained nonlinear optimization problems with particle swarm optimization. In: Cybernetics and intelligent systems IEEE conference
51. Lee K, Park J (2006) Application of particle swarm optimization to economic dispatch problem: advantages and disadvantages? In: IEEE PSCE
52. Bratton D, Kennedy J (2007) Defining a standard for particle swarm optimization. In: SIS 2007. IEEE swarm intelligence symposium, 2007, April
53. Zhang L, Hu S, Yu H (2003) A new approach to improve particle swarm optimization, volume 2723/2003. Genet Evolut Comput. ISBN 978-3-540-40602-0
54. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proc. IEEE Int'l, pp 1942–1948, vol 4 conference on neural networks
55. del Valle Y, Digman M, Gray A, Perkel J, Venayagamoorthy GK, Harley RG (2008) Enhanced particle swarm optimizer for power system applications. In: 2008 IEEE swarm intelligence symposium, pp 1–7
56. Fan HY (2002) A modification to particle swarm optimization algorithm? Eng Comput 19(7–8):970–989
57. Witt C, Sudholt D (2008) Runtime analysis of binary PSO. In: Proceedings of the 10th annual conference on genetic and evolutionary computation, Atlanta, GA, USA, pp 135–142
58. Khanesar MA, Teshnehlab M, Shoorehdeli MA (2007) A novel binary particle swarm optimization. In: 2007 Mediterranean conference on control automation, pp 1–6, June
59. Gao F, Gui G, Zhao Q (2006) Application of improved discrete particle swarm algorithm in partner selection of virtual enterprise. IJCSNS Int J Comput Sci Netw Secur 6:208–212
60. Hereford J, Gerlach H (2008) Integer-valued particle swarm optimization applied to Sudoku puzzles. SIS IEEE intelligence symposium, 2008
61. Shi WM, Shen Q, Ye BX, Kong W (2007) A combination of modified particle swarm optimization algorithm and support vector machine for gene selection and tumor classification? Talanta 71:1679–1683
62. Yu H, Gu G, Liu H, Shen J, Zhu C (2008) A novel discrete particle swarm optimization algorithm for microarray data-based tumor marker gene selection. In: 2008 International conference on computer science and software engineering, vol 1, pp 1057–1060
63. Droste S, Jansen T, Wegener I (2002) On the analysis of the (1+1) evolutionary algorithm? Theor Comput Sci 276:51
64. Hoeffding W (1994) Probability inequalities for sums of bounded random variables. Springer, New York, pp 409–426
65. Doerr B, Neumann F, Sudholt D, Witt C (2007) On the runtime analysis of the 1-ANT ACO algorithm. In: Proc. of GECCO 07, ACM, pp 33–40
66. Nvidia Corp Website (2012) NVIDIA CUDA C Programming Guide, version 4.2. https://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf
67. Kaur J, Singh S, Singh S (2016) Parallel implementation of PSO algorithm using GPGPU. In: Computational intelligence and communication technology (CICT), 2016 second international conference on IEEE, pp 155–159

68. Zhou Y, Tan Y (2009) GPU-based parallel particle swarm optimization. In: Evolutionary computation, 2009. CEC'09. IEEE Congress on IEEE, pp 1493–1500
69. Hung Y, Wang W (2012) Accelerating parallel particle swarm optimization via GPU. Optim Methods Softw 27(1):33–51
70. Wu Q, Xiong F, Wang F, Xiong Y (2016) Parallel particle swarm optimization on a graphics processing unit with application to trajectory optimization. Eng Optim 48(10):1679–1692
71. Nobile MS, Besozzi D, Cazzaniga P, Mauri G, Pescini D (2012) A GPU-based multi-swarm PSO method for parameter estimation in stochastic biological systems exploiting discrete-time target series. In: Mario G, Leonardo V, William SB (eds) Evolutionary computation, machine learning and data mining in bioinformatics. Springer, Berlin, pp 74–85
72. Kintsakis AM, Chrysopoulos A, Mitkas PA (2015) Agent-based short-term load and price forecasting using a parallel implementation of an adaptive PSO-trained local linear wavelet neural network. In: European Energy Market (EEM), 2015 12th international conference on the IEEE, pp 1–5
73. Ouyang A, Zhuo Tang X, Zhou YX, Pan G, Li K (2015) Parallel hybrid PSO with cuda for lD heat conduction equation. Comput Fluids 110:198–210
74. Tan Y (2016) GPU-based parallel implementation of swarm intelligence algorithms. Morgan Kaufmann, Burlington
75. Maruf HM, Hattori H, Fujimoto N (2016) A CUDA implementation of the standard particle swarm optimization. In: Symbolic and numeric algorithms for scientific computing (SYNASC), 2016 18th international symposium on IEEE, pp 219–226
76. Atashpendar A, Dorronsoro B, Danoy G, Bouvry P (2018) A scalable parallel cooperative coevolutionary PSO algorithm for multi-objective optimization. J Parallel Distrib Comput 112:111–125
77. Jararweh Y, Alzubi S, Hariri S (2011) An optimal multi-processor allocation algorithm for high performance GPU accelerators. In: 2011 IEEE Jordan conference on applied electrical engineering and computing technologies (AEECT), pp 1–6, Dec 2011
78. AlZubi S, Jararweh Y, Shatnawi R (2012) Medical volume segmentation using 3D multiresolution analysis. In: 2012 International conference on innovations in information technology (IIT)
79. AlZu'bi S, Shehab MA, Al-Ayyoub M, Benkhelifa E, Jararweh Y (2016) Parallel implementation of FCM-based volume segmentation of 3D images. In: 2016 IEEE/ACS 13th international conference of computer systems and applications (AICCSA), pp 1–6
80. Kothari V, Anuradha J, Shah S, Mittal P (2012) A survey on particle swarm optimization in feature selection. In: Krishna PV, Babu MR, Ezendu A (eds) Global trends in information systems and software applications. Springer, Berlin, pp 192–201
81. Souad LM-S (2015) A survey of particle swarm optimization techniques for solving university examination timetabling problem. Artif Intell Rev 44(4):537–546
82. Sun S, Abraham A, Zhang G, Liu H (2007) A particle swarm optimization algorithm for neighbor selection in peer-to-peer networks. In: Computer information systems and industrial management applications, 2007. CISIM '07. 6th International conference on June, pp 166–172
83. Koo Simon GM, Karthik K, George LCS (2006) On neighbor-selection strategy in hybrid peer-to-peer networks. Fut Gener Comput Syst 22(7):732–741
84. Papagianni C, Papadopoulos K, Pampas C, Tselikas ND, Kaklamani DT, Venieris IS (2008) Communication network design using particle swarm optimization. In: 2008 international multiconference on computer science and information technology, pp 915–920
85. Pióro M, Medhi D (2004) Routing, flow, and capacity design in communication and computer networks. Elsevier, Amsterdam
86. Mauricio GCR, Panos MP (2006) Handbook of optimization in telecommunications. Springer, New York
87. Mohemmed AW, Sahoo NC (2007) Efficient computation of shortest paths in networks using particle swarm optimization and noising metaheuristics. Discrete Dyn Nat Soc. https://doi.org/10.1155/2007/27383
88. Ali MKM, Kamoun F (1993) Neural networks for shortest path computation and routing in computer networks. IEEE Trans Neural Netw 4(6):941–954
89. Kuri J, Puech N, Gagnaire M, Dotaro E (2002) Routing foreseeable lightpath demands using a tabu search meta-heuristic. In: Global telecommunications conference, 2002. GLOBECOM '02. IEEE, vol 3, pp 2803–2807
90. Wook AC, Ramakrishna RS (2002) A genetic algorithm for shortest path routing problem and the sizing of populations. IEEE Trans Evolut Comput 6(6):566–579
91. Charon I, Hudry O (1993) The noising method: a new method for combinatorial optimization. Oper Res Lett 14(3):133–137
92. Shahin G, Falah A, Elias S (2008) Trained particle swarm optimization for ad-hoc collaborative computing networks. In: AISB 2008 convention, symposium on swarm intelligence algorithms and applications. Aberdeen, UK
93. Alfawaer Z, Hua G, Abdullah M, Mamady I (2007) Power minimization algorithm in wireless ad-hoc networks based on PSO. J Appl Sci 7(17):2523–2526
94. Muqattash A, Krunz M (2003) Power controlled dual channel (PCDC) medium access protocol for wireless ad hoc networks. In: IEEE INFOCOM 2003. Twenty-second annual joint conference of the IEEE computer and communications societies (IEEE Cat. no. 03CH37428), vol 1, pp 470–480
95. Ramanathan R, Rosales-Hain R (2000) Topology control of multihop wireless networks using transmit power adjustment. In: Proceedings IEEE INFOCOM 2000. Conference on computer communications. Nineteenth annual joint conference of the IEEE computer and communications societies
96. Dutta D, Choudhury K (2013) Network anomaly detection using PSO-ANN. Int J Comput Appl 77(2):35–42
97. Shing-Han L, Yu-Cheng K, Zong-Cyuan Z, Ying-Ping C, David CY (2015) A network behavior-based botnet detection mechanism using PSO and k-means. ACM Trans Manag Inf Syst 6(1):3
98. Priyadharshini C, ThamaraiRubini K (2012) PSO based route lifetime prediction algorithm for maximizing network lifetime in MANET. In: Recent trends in information technology (ICRTIT), 2012 international conference on IEEE, pp 270–275
99. Swain RR, Khilar PM (2017) Soft fault diagnosis in wireless sensor networks using PSO based classification. In: Region 10 conference, TENCON 2017 IEEE, pp 2456–2461
100. Li K, Bao J, Lu Z, Qi Q, Wang J (2017) A PSO-based virtual SDN customization for multi-tenant cloud services. In: Proceedings of the 11th international conference on ubiquitous information management and communication ACM, p 91
101. Lakshmanan L, Tomar DC (2014) Optimizing localization route using particle swarm-a genetic approach. Am J Appl Sci 11(3):520
102. Vimalarani C, Subramanian R, Sivanandam SN (2016) An enhanced PSO-based clustering energy optimization algorithm for wireless sensor network. Sci World J. https://doi.org/10.1155/2016/8658760
103. Cheng L, Wang Y, Chengdong W, Han Q (2015) A PSO-based maintenance strategy in wireless sensor networks. Intell Autom Soft Comput 21(1):65–75
104. Ren W, Zhao C (2013) A localization algorithm based on SFLA and PSO for wireless sensor network. Inf Technol J 12(3):502–505

105. Keun-Chang K (2012) An optimization of granular networks based on PSO and two-sided Gaussian contexts. Int J Adv Res Artif Intell 1(9):2012
106. Mahmoud A-A, Shadi A, Yaser J, Shehab Mohammed A, Gupta Brij B (2018) Accelerating 3D medical volume segmentation using GPUs. Multimed Tools Appl 77(4):4939–4958
107. Kaur H, Sharma S (2016) Analysis of metrics: improved hybrid ACO-PSO based routing algorithm for mobile ad-hoc network. In: 2016 Fourth international conference on parallel, distributed and grid computing (PDGC), pp 703–708
108. Aziz IT, Jin H, Abdulqadder IH, Imran RM, Flaih FMF (2017) Enhanced PSO for network reconfiguration under different fault locations in smart grids. In: 2017 International conference on smart technologies for smart nation (SmartTechCon), pp 1250–1254
109. Pluhacek M, Senkerik R, Viktorin A, Kadavy T (2017) Exploring the shortest path in PSO communication network. In: Computational intelligence (SSCI), 2017 IEEE symposium series on IEEE, pp 1–6
110. Hou R, Chang Y, Yang L (2017) Multi-constrained QoS routing based on PSO for named data networking. IET Commun 11(8):1251–1255
111. Salama Hussein F, Reeves Douglas S, Yannis V (1997) Evaluation of multicast routing algorithms for real-time communication on high-speed networks. IEEE J Sel Areas Commun 15(3):332–345