Research Article

Journal of Algorithms & Computational Technology

Journal of Algorithms & Computational Technology
2017, Vol. 11(1) 23–30
© The Author(s) 2016
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/1748301816665021
journals.sagepub.com/home/act
SAGE

A review of velocity-type PSO variants

Ivo Sousa-Ferreira and Duarte Sousa

Abstract

This paper presents a review of the particular variants of particle swarm optimization, based on the velocity-type class. The original particle swarm optimization algorithm was developed as an unconstrained optimization technique, which lacks a model that is able to handle constrained optimization problems. The particle swarm optimization and its inapplicability in constrained optimization problems are solved using the dynamic-objective constraint-handling method. The dynamic-objective constraint-handling method is originally developed for two variants of the basic particle swarm optimization, namely restricted velocity particle swarm optimization and self-adaptive velocity particle swarm optimization. Also on the subject velocity-type class, a review of three other variants is given, specifically: (1) vertical particle swarm optimization; (2) velocity limited particle swarm optimization; and (3) particle swarm optimization with scape velocity. These velocity-type particle swarm optimization variants all have in common a velocity parameter which determines the direction/movements of the particles.

Keywords

Stochastic algorithms, particle swarm optimization, velocity-type, constrained optimization, nonlinear programming

Date received: 15 July 2015; accepted: 31 March 2016

Introduction

The particle swarm optimization (PSO) has a cooperative nature.[1] Over time several applications of PSO have been developed, but a disadvantage of this algorithm is the lack of assurance that will find the optimum solution, and also the high computational cost associated with the fitness function (FF). Compared to genetic algorithm, the PSO is faster when looking close to ideal solutions, although it is also faster to prematurely convergence.[2]

In the literature, many variants of PSO are available. Sedighizadeh and Masehian[3] categorized these variants according to fuzziness, continuity, accordance, topology, attraction, velocity-type, activity, grouping, mobility, hierarchy, divisibility, interaction, etc.

In this review, we rely on the category called ''velocity-type'' which is composed by five PSO variants including the PSO with: restricted (and unrestricted) velocity, self-adaptive velocity, vertical velocity, limited velocity, and escape velocity.

The restricted velocity PSO (RVPSO) appeared due to constrained optimization problems (COPs). Lu and Chen[4] went further, introducing a new constraint treatment technique called dynamic-objective constraint-handling method (DOCHM). This method transforms COPs into unconstrained optimization problems (UPSO).

The basic PSO algorithm was initially developed to solve the unconstrained optimization problems. Hence, it lacks a mechanism to adapt the algorithm to optimization problems with constraints. Thus, the self-adaptive velocity PSO (SAVPSO) is an algorithm that employs a mechanism to study the impact of constraints on the basic PSO algorithm in order to enhance the particle's search ability in the feasible region. Therefore, each particle will have the capacity to self-adaptively regulate its velocity according to the existing characteristics in the feasible region.[5]

In some interactions of the PSO, the global optimum is not perfected and the particles remain near the optimum point but do not reach it. As the movements of

Department of Mathematics, Faculty of Exact Sciences and Engineering, University of Madeira, Funchal, Portugal

Corresponding author:
Ivo Sousa-Ferreira, University of Madeira, Campus da Penteada, 9020-105 Funchal, Portugal.
Email: 2044511@student.uma.pt

the particles depend on its previous movements, they tend to move to a local optimum. The vertical PSO (VPSO) algorithm was developed to overcome the limitations of the basic PSO algorithm. The VPSO assumes that the particles fly in two directions: towards the global best particle or in vertical direction. At each iteration, a random number is generated, measuring the probability of flying in both directions. Yang[6] states that the PSO and the VPSO are used to train neural network (NN) and applies a neural network based on VPSO (VPSONN) in soft-sensor modeling of acrylonitrile yield.

The velocity limited variant (VLPSO) introduced by Xu and Chen[7] arose to help with financial decisions about the construction of a portfolio of investments with minimal risk but with maximum expected return, considering constraints for velocity and position of the particles. Here, the return rates were considered as stochastic variables being applied to the VLPSO algorithm to solve this problem.

The PSO with escape velocity (EVPSO) is an algorithm that enables the particles with an escape velocity so that they can free themselves from the local optimum. Thus, it is possible to prevent premature convergence observed in the basic PSO algorithm and enhance the population diversity of the swarm, safeguarding the efficient performance of PSO.[8]

Particle swarm optimization

Kennedy and Eberhart[9] were the first to introduce the particle swarm optimization (PSO) algorithm based on the observation of the behavior of a birds flock in search for food. This meta-heuristic algorithm has a population nature based on cooperation. In comparison with other evolutionary algorithms, the PSO does not have crossover between individuals nor mutation during iterations of the algorithm.

In the PSO, the swarm's particles start the exploration of the search space with randomly generated position and velocity, and the particles move only within the search space guided by the velocity in the previous instant, the best position found by the particle in question and the best position found by the set of particles in the neighborhood of the particle itself, in each iteration.

The PSO concept consists in updating the velocity of each particle, at each instant of time, relatively to the best position found by the particle and the best position found by the neighborhood. Thus, supposing that the search space is d-dimensional, then we can represent the current position of the particle in the search space by vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})'$, the best position of the particle until then, i.e., its cognitive memory can be represented by $P_i = (p_{i1}, p_{i2}, \ldots, p_{id})'$ and finally the particle's velocity is given by $V_i = (v_{i1}, v_{i2}, \ldots, v_{id})'$. Also, defining g as an index to the best particle in the swarm, updating the position and velocity at each iteration is made by the following equations[10]

$$\vec{v}_i^{k+1} = w\vec{v}_i^k + c_1 r_1(\vec{p}_i^k - \vec{x}_i^k) + c_2 r_2(\vec{p}_g^k - \vec{x}_i^k) \qquad (1)$$

$$\vec{x}_i^{k+1} = \vec{x}_i^k + \vec{v}_i^{k+1} \qquad (2)$$

w is the inertia weight; c_1, c_2 are two positive constants called cognitive and social coefficients, respectively; r_1, r_2 are random numbers of the interval [0, 1] generated at each iteration of the algorithm, for each particle in each dimension; and $k = 1,2,\ldots$, defines the number of iterations.

PSO is influenced by multiple control parameters such as the inertia weight, the neighborhood size, the size of the problem, the population size, the maximum velocity, the cognitive and social parameters, and so on. The update equation of velocity presented includes the inertia weight, w, which is important to ensure convergent behavior. The most famous way to adjust the inertia weight is to use a large parameter w at the beginning of the exploration, and gradually reduce it along the iterations.

Compared to other evolutionary computation (EC) techniques, the lack of a velocity control mechanism in the PSO resulted in its low efficiency in terms of performance.[11] There are several researchers who study the neighborhood's size. For example, Eberhart and Kennedy[12] concluded that a small local neighborhood is best to avoid local minimum, since a global neighborhood converges rapidly.

PSO and its inapplicability to COPs

The basic PSO algorithm, first proposed in 1995,[9,12] has been developed to handle unconstrained optimization problems.[5] In this section, the aspects of the inapplicability of basic PSO algorithm for solving COPs will be examined.

Considering equation (1), the right member can be decomposed into three parts.[13] The first part $w\vec{v}_i^k$ represents the particle's velocity in the previous instant. The second part $c_1 r_1(\vec{p}_i^k - \vec{x}_i^k)$ represents the cognitive portion of the particle, i.e. symbolizes what it has been capable of learning in the optimization process so far. The third part $c_2 r_2(\vec{p}_g^k - \vec{x}_i^k)$ represents the social behavior of the particle, i.e. indicates the best position found in the particle's neighborhood.

The sum of $c_1 r_1(\vec{p}_i^k - \vec{x}_i^k)$ with $c_2 r_2(\vec{p}_g^k - \vec{x}_i^k)$ is seen as the new velocity term acquired, while the term $w\vec{v}_i^k$ is the particle's velocity at the previous instant. The sum of the terms described above results in the obtainment of $\vec{v}_i^{k+1}$. These two terms do not consider the influence of the feasible region, and thus, for example, if one or both terms are large, the particle will leave the feasible region. This shows the great difficulty of the basic PSO in solving COPs.

Dynamic-objective constraint-handling method

Lu and Chen[4,5] formulated COPs as follows

$$\text{minimize} \quad f(\vec{x}) = \vec{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n \tag{3}$$

where $f(\vec{x})$ is the objective function, $\vec{x} \in S \cap F$, $S \subseteq F$, $S \subseteq \mathbb{R}^n$ defines the *search space* by parametric constraints $l_d \leq x_2 \leq u_d$ and *feasible region* F is defined by

$$g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, q \tag{4}$$

$$h_i(\vec{x}) = 0, \quad j = q+1, \ldots, m \tag{5}$$

Typically, the equality constraints are transformed into inequalities shaped

$$|h_i(\vec{x})| - \epsilon \leq 0, \quad j = q+1, \ldots, m \tag{6}$$

The solution $\vec{x}$ is considered *feasible* if $g_i(\vec{x}) \leq 0$, for $i = 1, \ldots, q$ and $|h_i(\vec{x})| - \epsilon \leq 0$, for $j = q+1, \ldots, m$, where ϵ was defined as 0.001.

When we apply the PSO to a COP, each particle moves toward the promising region in the search space. According to Lu and Chen,[4] a COP can be seen as a bi-objective unconstrained optimization. The first objective is to enter the feasible region, and the second is to optimize the original function of the COP, in order to find the optimal solution. Therefore, it is reasonable to imagine that only after a particle entered the feasible region, it is possible to approach the optimal solution.

During the search process of the PSO, the mechanism presented by the authors is designed in such a way that the particle has the ability to adjust depending on their goals, whether it lies inside or outside the feasible region, and hence this mechanism is designated dynamic-objective constraint-handling method (DOCHM).

A simple way to optimize all optimal solutions to the COP is given by the following function

$$\phi(\vec{x}) = \sum_{i=1}^{q} \max(0, g(\vec{x})) + \sum_{i=1}^{q} \max(0, |h_j(\vec{x})| - \epsilon) \tag{7}$$

Evidently $\phi(\vec{x})$ is the sum of constraint violations, $\phi(\vec{x}) \geq 0$, and thus $\phi(\vec{x}) = 0$ for $\forall \vec{x} \in F$. Here, all the great solutions $\phi(\vec{x})$ are the feasible region F of the original problem.

In DOCHM, the function (3) is only used as a way to measure the distance that the particle is from the feasible region. Therefore, the initial COP is converted into the following bi-objective unconstrained problem

$$\begin{aligned} \min \ & F(\vec{x}) = (\phi(\vec{x}), f(\vec{x})) \\ & \vec{x} = (x_1, \ldots, x_n) \subset S \in \mathbb{R}^n \end{aligned} \tag{8}$$

In theory, only when $\phi(\vec{x}) = 0$ does the particle begins to minimize $f(\vec{x})$. But in practice, $f(\vec{x})$ can be optimize initially by indicating a threshold $\delta \geq 0$ such that $\phi(\vec{x}) \leq \delta$, so that the particle begins to optimize $f(\vec{x})$. Thus, δ controls when the optimization process $f(\vec{x})$ should start.

Notice that after starting the optimization $f(\vec{x})$, there is a probability of the particle to be projected out of the feasible region. If it happens, the particle should give up optimizing $f(\vec{x})$ and go back to minimizing $\phi(\vec{x})$. Thus, each particle has the ability to dynamically adjust its objective, according to its current position in the search space.

The incorporation of DOCHM with a PSO algorithm makes it easy to get the update strategies $\vec{p}_i$ and $\vec{p}_g$, i.e. if the current position of a particle is within the feasible region, then $\vec{p}_i$ is defined as the strongest viable solution of $f(\vec{x})$ found so far. Otherwise, the solution found $\vec{p}_i$ is defined as the position closer to the feasible region. For $\vec{p}_g$, if all cluster positions fall outside the feasible region then $\vec{p}_g$ is taken as the nearest position from the feasible region. Moreover, $\vec{p}_g$ is defined as the best suited feasible solution found by the swarm. This constraint of the generic method of treatment is suitable for several variants of the PSO algorithm. The DOCHM pseudo-code on upgrading $\vec{p}_i$ and $\vec{p}_g$ is described as

```
$\phi_{ibest} = \phi(\vec{p}_i)$, $f_{ibest} = f(\vec{p}_i)$, $\phi_i = \phi(\vec{x}_i^k)$
**if** $\phi_i < \phi_{ibest}$ **then** $\vec{p}_i \leftarrow \vec{x}_i^k$, $\phi_{ibest} \leftarrow \phi_i$ **end**
**if** $\phi_i = \phi_{ibest}$ and $\phi_{ibest} \leq \delta$ **then**
    $f_i = f(\vec{x}_i^k)$
    **if** $f_i \leq f_{ibest}$ **then** $\vec{p}_i \leftarrow \vec{x}_i^k$, $f_{ibest} \leftarrow f_i$ **end**
**end**
$\phi_{gbest} = \phi(\vec{p}_g)$, $f_{gbest} = f(\vec{p}_g)$
**if** $\phi_i < \phi_{gbest}$ **then** $\vec{p}_g \leftarrow \vec{x}_i^k$, $\phi_{gbest} \leftarrow \phi_i$ **end**
**if** $\phi_g = \phi_{gbest}$ and $\phi_{gbest} \leq \delta$ **then**
    **if** $f_i \leq f_{gbest}$ **then** $\vec{p}_g \leftarrow \vec{x}_i^k$ $f_{gbest} \leftarrow f_i$ **end**
**end**
```

Restricted velocity PSO

As mentioned above, the basic PSO was originally designed to solve unconstrained optimization problems, so it does not take into account the impact of the feasible region, i.e. constraints on the search mechanism, when it comes to solving COPs.

Lu and Chen[4] conducted a modification in the basic PSO search system in order to embed the impact factor from the feasible region and to improve algorithm performance on the COP resolution. Thus, the authors propose a new algorithm, the restricted velocity particle swarm optimization (RVPSO), which includes the impact of the feasible region in the velocity of the particle swarm.

Consider the update velocity equation (1) of basic PSO divided into two parts. The first part $w\vec{v}_i^k$ is the particle's velocity in the immediately preceding moment, and the second part $c_1r_1(\vec{p}_i^k - \vec{x}_i^k) + c_2r_2(\vec{p}_g^k - \vec{x}_i^k)$ is the particle's velocity towards the potential region around $\vec{p}_i^k$ and $\vec{p}_g^k$. Thus, the authors presented the following two changes

- Set $c_1 = c_2 = 1$, making the potential location $\vec{x}_i^k + r_1(\vec{p}_i^k - \vec{x}_i^k) + r_2(\vec{p}_g^k - \vec{x}_i^k)$ not to go far from $\vec{p}_i^k$ and $\vec{p}_g^k$, improving efficiency of the particles.
- Replacing $w\vec{v}_i^k$ by $w(\vec{p}_g^k - \vec{p}_j^k)$, where p_j^k is the best position of a particle randomly selected, and j is a random integer number uniformly distributed in the interval $[1, N]$. This is because, on one hand, $w\vec{v}_i^k$ in the basic PSO does not use the information on the feasible region and should be replaced by a term of appropriate velocity, being able to reflect the impact of the feasible region. On the other hand, when $\vec{p}_g^k$ and $\vec{p}_j^k$ lie within the feasible region $(\vec{p}_g^k - \vec{p}_j^k)$, it reflects the approximate size of the feasible region around $\vec{p}_g^k$.

Bearing in mind the two modifications presented above, the authors presented a new PSO algorithm, the restricted velocity particle swarm optimization, which is especially applied in COPs. Furthermore, the term $\vec{v}_i^{k+1}$ of velocity update equation (1) can be canceled. As a result, there is only one RVPSO update equation, the update equation of the position of the form:

$$\vec{x}_i^{k+1} = \vec{x}_i^k + r_1(\vec{p}_i^k - \vec{x}_i^k) + r_2(\vec{p}_g^k - \vec{x}_i^k) + w(\vec{p}_g^k - \vec{p}_j^k) \tag{9}$$

When applying RVPSO for COPs, it is necessary to incorporate a method of treating constraints, as noted above. The authors propose the DOCHM that is rooted in the PSO search mechanism. Thus, Lu and Chen[4] proposed the combination of RVPSO with DOCHM, a union of algorithms in order to solve COPs.

Moreover, the authors presented a simple way to hold the particles within the search space, i.e. if $x_{id}^k > u_d$ or $x_{id}^k < l_d$, then

$$x_{id}^k = (p_{id}^k + p_{gd}^k + p_{jd}^k + p_{j'd}^k)/4, \quad d = 1, \ldots, N \tag{10}$$

where d is the d-dimension of the search space, j and j' are two uniformly distributed random integers in the range $[1, N]$.

Pseudo-code of DOCHM + RVPSO

```
Step 0: Initialize $\vec{x}_i^0$, $l_i \le x_{id}^0 \le u_i$, $i = 1, \ldots, N$ and
    $d = 1, \ldots, n$;
Step 1: Update $\vec{x}_i^{k+1}$ according to equation (9), and
    clamp down the particles by use of equation (10)
    (if needed);
Step 2: Calculate $\vec{p}_i$ and $\vec{p}_g$ according to the procedures
    describes in DOCHM pseudo-code;
Step 3: If stop criteria are not satisfied, then go to
    Step 1.
```

Self-adaptive velocity PSO

The basic PSO algorithm was initially developed to solve unconstrained optimization problems; hence, it lacks a mechanism to solve COPs. Much of the literature about the solving of COPs with the basic PSO focuses on the way to deal with such constraints and not on their impact on the PSO search mechanism.

Lu and Chen[5] developed an algorithm called SAVPSO that uses a mechanism capable of dealing with COPs and that studies the impact of the constraints in that mechanism, in order to improve the PSOs ability. The SAVPSO algorithm claims that each particle of the swarm has the ability to self-adapt its velocity taking into account some characteristics of the feasible region.

To deal with constraints, SAVPSO adopted the same mechanism used in RVPSO. DOCHM operates in the inherent search mechanism, showing the impact of constraints and forcing the particles to seek the feasible region in the search space.

In SAVPSO, the particles of the swarm are handled according to the following equations

$$\vec{v}_i^{k+1} = w\,|\vec{p}_{i'}^k - p_i^k|\text{sign}(\vec{v}_i^k) + r(\vec{p}_i^k - \vec{x}_i^k) + (1-r)(\vec{p}_g^k - \vec{x}_i^k) \tag{11}$$

$$\vec{x}_i^{k+1} = \vec{x}_i^k + \vec{v}_i^{k+1} \tag{12}$$

where $r \in U[0, 1]$; $i' \in intU[1, N]$; w is the inertia weight and sign $(\vec{v}_i^k)$ is the sign of $(\vec{v}_i^k)$.

The update equation of velocity of SAVPSO was obtained by changing the update equation of velocity of basic PSO stated in equation (1). Comparing with equation (1), in the equation $\vec{v}_i^{k+1}$ of SAVPSO was set $c_1 = c_2 = 1$ and $r_1 = 1 - r_2$. So the new velocity term will cause the particle i to fly to a position located between p_g and p_i and will push the particle i to near the feasible region or to the feasible region.

Another modification done was the replacement of $w\vec{v}_i^k$ by $w\,|\vec{p}_{i'}^k - \vec{p}_i^k|\text{sign}(\vec{v}_i^k)$, where $\text{sign}(\vec{v}_i^k)$ represents the sign of $(\vec{v}_i^k)$ and indicates the flight direction of $(\vec{v}_i^k)$ taken in the search space. The magnitude of the particle i is determined by $w|\vec{p}_{i'}^k - \vec{p}_i^k|$, taking into account the effect of the feasible region. Thus, since p_i and $p_{i'}$ are contained or near the feasible region, $|\vec{p}_{i'}^k - \vec{p}_i^k|$ represents approximately the size of the feasible region.

Consequently, with $w\,|\vec{p}_{i'}^k - \vec{p}_i^k|$, the particle i will not diverge much from the feasible region. It should be noted that the value of $|\vec{p}_{i'}^k - \vec{p}_i^k|$ may change in a self-adaptively way, depending on the changes of the search scope of the swarm.

Making an integration of SAVPSO to solve COPs, we have to keep in mind that the limits of the feasible region may be very close to the parametric limits x^l and/or x^u so it is very probable that some particles close to the limits of the feasible region violate the parametric constraints.

In order to overcome this problem, Lu and Chen[4,5] have adopted the same technique used above in RVPSO. Thus, $\vec{x}_i^k$ was randomly reviewed and the following equation appeared

$$\vec{x}_i^k = \begin{cases} \bar{x}^k + ar_4(x^l - \bar{x}^k), & \text{if } \vec{x}_i^k < x^l \\ \bar{x}^k + ar_4(x^u - \bar{x}^k), & \text{if } \vec{x}_i^k > x^u \end{cases} \tag{13}$$

where $\bar{x}^k = \left(\sum_{i=1}^N \vec{x}_i^k\right)/N$, $r_4 \in U[0,1]$, and a is a constant number in the range $[0,1]$.

Finally, the authors describe the integrated SAVPSO algorithm in pseudo-code, where $I_{\max}$ is the maximum number of iterations, where it is possible to see that the SAVPSO incorporates DOCHM as one of the components of its search technique.[5]

```
Pseudo-code of the Integrated SAVPSO:
Create and initialize an n-dimensional swarm
For $k = 0$ to $I_{\max}$
  $\bar{x}^k = \left(\sum_{i=1}^{N} \vec{x}_i^k\right)/N,\ k = 1, 2, \ldots, n$
  For $i = 1$ to $N$
      For $k = 1$ to $n$
      $\vec{x}_i^{k+1} = \vec{x}_i^k + \vec{v}_i^{k+1}$
      $\vec{v}_i^{k+1} = w|\vec{p}_{i'}^k - p_i^k|\text{sign}(\vec{v}_i^k) + r(\vec{p}_i^k - \vec{x}_i^k)$
            $+(1-r)(\vec{p}_g^k - \vec{x}_i^k)$
      If $\vec{x}_i^{k+1} > x^u$ Then $\vec{x}_i^{k+1} = \bar{x}^k + ar_4(x^u - \bar{x}^k)$
      End
      If $\vec{x}_i^{k+1} < x^l$ Then $\vec{x}_i^{k+1} = \bar{x}^k + ar_4(x^l - \bar{x}^k)$
      End
    End
    Call Procedure for Calculation $\vec{p}_i^{k+1}$ and $\vec{p}_g^{k+1}$
  End
End
```

Vertical PSO

The vertical PSO algorithm is a variant of the PSO developed to solve the basic PSOs problems such as premature convergence to local optimums, the loss of population diversity, and so on.

Considering equation (1), if there is not an improvement of the best global fitness of the neighborhood for a few iterations, so when the swarm's particles are close to the best global position found, they will move in the same direction and converge to a local optimum. Therefore, to solve this problem VPSO was proposed.

VPSO claims that the particles can move in two different directions: towards the position of the global best particle or in vertical direction. Furthermore, in each iteration, a random number is generated to measure the probability of a particle to move in any of the abovementioned directions.[6]

In VPSO, the velocity of the particle and its new position will be given by the following equations

$$v_i = wv_i + c_1r_1(p_i - x_i) + c_2r_2(p_g - x_i) \tag{14}$$

$$vv_i = vv_i - \left(\frac{dot(v_i, vv_i)}{dot(v_i, v_i)}\right) \times v_i \tag{15}$$

$$x_i = x_i + rand \times v_i + (1 - rand) \times vv_i \tag{16}$$

where $rand$ is a random number generated at each iteration representing the probability of moving in either direction and belongs to the interval $U(0,1)$; $dot()$ is a dot distributing between v_i and vv_i; $v_i = (v_{i1}, v_{i2}, \ldots, v_{id})$ represents the direction of the global optimum; $vv_i = (vv_{i1}, vv_{i2}, \ldots, vv_{id})$ represents the vertical direction to v_i.

The application of VPSO in the soft-sensor model of acrylonitrile yield

Acrylonitrile is the main material of polyacrylonitrile production. The fabrication of acrylonitrile is a complex industrial process and thus, during the making of acrylonitrile, its yield is a very significant product indicator. Therefore, it is vital to obtain the precise acrylonitrile yield online and on real time to accurately control the fabrication of acrylonitrile.

Monitoring the acrylonitrile yield online with an analysis instrument can be ineffective, so Yang[6] suggests soft-sensing technology to control acrylonitrile yield online. He also claims that the VPSO algorithm is used to train neural network, and so the VPSONN emerges and it is applied in soft-sensor modeling of acrylonitrile yield.

The acrylonitrile yield is affected by seven variables: reaction pressure, temperature, the amount of pure propylene, the amount of ammonia, the amount of air, activators density, and reaction speed. To measure the acrylonitrile yield, it is crucial to discover the relationship between the acrylonitrile yield and the seven variables mentioned before. The VPSONN is utilized to determine that relationship. Since VPSONN is applied in soft-sensor modeling of acrylonitrile yield, it is important to define the objective function of the soft-sensing model[6]

$$\min E = \sum (t - y)^2 \tag{17}$$

where t is the observed value of acrylonitrile yield; y is the predicted value of acrylonitrile yield. The mean square error (MSE) and the mean absolute error (MAE) are used to calculate the performance of the soft-sensor model and are defined as following

$$MSE = \frac{1}{n}\sum_{1}^{n} (t - y)^2 \tag{18}$$

$$MAE = \frac{1}{n}\sum_{1}^{n} |t - y| \tag{19}$$

The VPSONN algorithm is defined in the subsequent stages[6]

Stage 1: Initialize the structure of NN and the parameters of VPSO.
Stage 2: Initialize the state of each particle. Calculate the corresponding output fitness of NN, and store the individual best position and the best fitness of each particle, and the best position and the best fitness of the whole swarm.
Stage 3: Update the velocity and position according to equations (14), (15), (16), respectively. If necessary, limit the particles velocity and position.
Stage 4: Calculate the corresponding fitness of each particle, update and store the individual best position and individual best fitness of each particle, update and store the global best position and global best fitness of whole swarm.
Stage 5: If the stopping condition is not satisfied, go to Step3. Otherwise, stop iterating and output the global best position and the global best fitness of the whole swarm as the result.

Velocity limited PSO

One of the most important financial decisions that people and institutions have to deal with is to elaborate an investment portfolio. Here comes the modern portfolio theory, where the investor chooses the proportions of the portfolio assets, rationally, seeking to maximize the expected return and minimize the associated risks.

In order to formalize the problem, the authors resorted to the mean-variance model of Markowitz[14] that elucidates how it should be done with the selection of the portfolio with N risky assets ($N \geq 2$). The rate of return $\bar{r}_i$ for i assets is a random variable with expected return $R_i = E(\bar{r}_i)$, $i = 1, \ldots, N$. Let x_i to be the investment rate for i assets. Conveniently set $X = (x_1, x_2, \ldots, x_n)'$, $\bar{R} = (\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_n)'$, $F = (1, 1, \ldots, 1)'$, $v = (\sigma_{ij})_{N\times N}$ where $\bar{R}$ is the return vector and V is the covariance matrix of returns. Its description is given by the following quadratic programming

$$\begin{aligned} \min \quad & X'VX \\ s.t. \quad & X'R = R_0 \\ & X'F = 1 \end{aligned} \tag{20}$$

here R_0 is the expected return of the investor and $X'F = 1$ is the constraint condition.

Xu and Chen[7] considered the rates of return expected as stochastic variables, in order to solve the problem by applying swarm optimization. A major goal of the authors is to overcome the global and local search limitation of traditional numerical algorithms in order to solve nonlinear programming problems. So a new variant of PSO algorithm is introduced, called velocity limited particle swarm optimization (VLPSO).

The basic PSO algorithm is not enough to solve optimization problems in small regions, because the best solution is frequently lost. We also know that the efficiency and accuracy of an optimization are determined by the exploration-exploitation trade-off, and is controlled by the velocity update equation (1). Therefore, focusing on the information obtained by the velocity equation, if we limit the velocity in a different scope, we can find a different best solution.

To VLPSO, the following limits were defined: firstly, if the particle velocity is greater than v_{max}, it was set equal to v_{max}; if the velocity is smaller than v_{min}, it was set equal to v_{min}. Secondly, if the found solution is greater than p_{max}, it was set equal to p_{max}; if the solution is smaller than p_{min}, it was set equal to p_{min}, where p_{max} and p_{min} are the range solution. For the constraint condition, the strategy used is preserve only the particles that satisfy the constraint condition, while the others are abandoned. The authors also presented the pseudo-code of VLPSO algorithm for the model of the investment portfolio[7]

```
Initialize $N$ particle;
{Set constants $w$, $c_1$,$c_2$;
Randomly initialize particle positions $x_0^i \in D$ in $\mathbb{R}$ for $i = 1, 2, \ldots, p$, which satisfy constraint condition $X'F = 1$;
Randomly initialize particle velocities $0 \leq v_0^i \leq v_0^{max}$ for $i = 1, 2, \ldots, p$;
Evaluate function values $f_0^i$ using design space coordinates $x_0^i$ for $i = 1, 2, \ldots, p$;
Set $f_{best}^i = f_0^i$ and $p_0^i = x_0^i$ for $i = 1, 2, \ldots, p$;
Set $f_{best}^g$ to best $f_{best}^i$ and $g_0$ to corresponding $x_0^i$;
}
While (not determinated) do
    {
    For each particle
        {Evaluate function values $f_k^i$ using design coordinates $x_k^i$, for $i = 1, 2, \ldots, p$;
        If $f_k^i$ is not satisfied constraint condition then ($f_k^i = C$(C is big enough))
        If $f_k^i \leq f_{best}^i$ then $f_{best}^i = f_k^i$, $p^i = x_k^i$
        }
    For each particle
        {Calculate particle velocity $v_{k+1}^i$ according to equation. (1);
            If $v_{k+1}^i > v_{max}$, then $v_{k+1}^i = v_{max}$
            If $v_{k+1}^i < v_{min}$, then $v_{k+1}^i = v_{min}$
        Update particle position $x_{k+1}^i$ according to equation (2)
            If $x_{k+1}^i > p_{max}$, then $x_{k+1}^i = p_{max}$
            If $v_{k+1}^i < p_{min}$, then $x_{k+1}^i = p_{min}$
        }
    }
```

PSO with escape velocity

The basic PSO algorithm often converges prematurely and its performance is adversely affected by the loss of population diversity. The variant of PSO with escape velocity equips the swarm's particles with an escape velocity, and thus they escape from local optimum. Consequently, through this variant, it will be possible to overcome the imperfections of the basic PSO namely the rapid convergence and the lack of population diversity.

To make it possible to provide the particles with the escape velocity, i.e. the ability to move continuously, it is necessary to define a new equation for each particles velocity[8]

$$\vec{v}_i^{k+1} = w\vec{v}_{ev}^k + c_1 r_1(\vec{p}_i^k - \vec{x}_i^k) + c_2 r_2(\vec{p}_g^k - \vec{x}_i^k) \tag{21}$$

$$\vec{v}_{ev} = \begin{cases} V_{i,j}(t), & |V_{i,j}(t)| > e_c \\ r_j \times \dfrac{V_{max}}{\rho}, & |V_{i,j}(t)| < e_c \end{cases} \tag{22}$$

where r_j is a sequence of random numbers selected from $U(-1, 1)$; ρ is a factor that controls the domain of the escape velocity; and e_c is the limit that affects the condition of the escape case.

The EVPSO is useful when most of the particles are concentrated in the same convex subset, indicating other points outside of this subset by updating the particles velocity. If the point found in another convex subset is less than the current best global value, the new point will be adopted as the new best global value. Eventually, all the particles will move to the new convex subset. The exploration of the search space will end only when the algorithm converges to the crucial minimum subset.

The performance of the EVPSO algorithm is directly dependent on the choice of values for the parameters e_c and ρ. Large e_c values provide a global search while small e_c values favor a local search. And large ρ values reduce the particle's escape domain, while small ρ values increase the escape domain.

Wang et al.[8] stated that to obtain the desired exploration-exploitation trade-off, it is necessary to divide the particle search into two parts: initially e_c is set at a large value and ρ is set at a small value; in the last stage, e_c is set as a small value and ρ as a large value. Thus, initially, the particles will perform big movements and will search the entire search space in pursuit of good local optimums while in the final stage, the particles will execute a fine grain search. Modifying equation (22), a new update equation for velocity is given

$$\vec{v}_{ev} = \begin{cases} V_{i,j}(t), & |V_{i,j}(t)| > e_c(t) \\ r_j \times \dfrac{V_{max}}{\rho(t)}, & |V_{i,j}(t)| < e_c(t) \end{cases} \tag{23}$$

Conclusion

In this review, we tried to present a specifically type of basic PSO variants, called velocity-type PSO variants. The main feature they all have in common is that it is possible to determine the direction or movements of the particles by defining a velocity parameter.

Based on the current literature, the PSO algorithm has experienced several changes in the form of variants, in order to overcome its faults. The variants presented in this paper are only a small portion of a long list of PSO variants.

Since its beginnings, the basic PSO algorithm has proven incapable of dealing with COPs. In order to solve this obstacle, the DOCHM emerged as a potential solution. Moreover, two PSO velocity-type variants have appeared and are associated with this mechanism, namely RVPSO and SAVPSO. The incorporation of

DOCHM in RVPSO and SAVPSO provides efficiency and effectiveness to the final results.

To solve another limitations of the basic PSO, three other variants of velocity-type are analyzed, specifically the VPSO, VLPSO, and EVPSO algorithms. The VPSO states that particles fly towards the global particle or in vertical direction. The VLPSO suggests the finding of optimal solution with the lowest number of interactions as possible. The EVPSO introduces an escape velocity to improve the particle's search capability. They all try to avoid the classical issue of basic PSO, which is the rapid convergence to local optimums.

The purpose of this review is to serve as a guide for future research, since due to the flexibility of the PSO algorithm, it has enormous applications in several areas. The PSO algorithm has a huge potential and room for improvement, placing it as a method of excellence for solving the most complex optimization problems. It is also important to study its variants in order to constantly update the algorithm and subsequently achieve the best results possible.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

References

1. El-Abd M. *Cooperative models of particle swarm optimizers*. PhD thesis, 2008, University of Waterloo, Waterloo, Ontario.
2. Banks A, Vincent J and Anyakoha C. A review of particle swarm optimization. Part I: background and development. *Nat Comput* 2007; 6: 467–484.
3. Sedighizadeh D and Masehian E. An particle swarm optimization method, taxonomy and applications. *Proc Int J Comput Theor Eng* 2009; 5: 486–502.
4. Lu H and Chen W. Dynamic-objective particle swarm optimization for constrained optimization problems. *J Combin Optimiz* 2006; 12: 409–419.
5. Lu H and Chen W. Self-adaptive velocity particle swarm optimization for solving constrained optimization problems. *J Global Optim* 2008; 41: 427–445.
6. Yang WP. Vertical particle swarm optimization algorithm and its application in soft-sensor modeling. *Proc IEEE/ICMLC Int Confer Mach Learn Cybernet* 2007; 4: 1985–1988.
7. Xu F and Chen W. Stochastic portfolio selection based on velocity limited particle swarm optimization. In: *Proceedings of IEEE/ WCICA of the sixth world congress on intelligent control and automation*, Dalian, 2006, IEEE, Vol. 1, pp. 3599–3603.
8. Wang X, Wang Y, Zeng H, et al. Particle swarm optimization with escape velocity. *Int Confer Comput Intell Security* 2006; 1: 457–460.
9. Kennedy J and Eberhart R. Particle swarm optimization. *Proc IEEE Int Confer Neural Networks* 1995; 4: 1942–1948.
10. Shi Y and Eberhart R. A modified particle swarm optimizer. *Proc IEEE Int Confer Evolution Comput* 1998; 1: 69–73.
11. Angeline PJ. Evolutionary optimization versus particle swarm optimization: philosophy and performance differences. In: Porto VW, Saravanan N, Waagen D, et al. (eds) *Evolutionary programming VII, Vol. 1447*. Berlin Heidelberg: Springer, 1998, pp.601–610.
12. Eberhart R and Kennedy J. A new optimizer using particle swarm theory. In: *Sixth international symposium on micro machine and human science*, Nagoya, 1995, IEEE, pp.39–43.
13. Shi Y. Particle swarm optimization. *IEEE Neural Network Soc* 2004; 2: 8–13.
14. Markowitz HM. Portfolio selection. *J Finance* 1952; 7: 77–91.