Towards an ideal Particle Swarm Optimizer for multidimensional functions

Vasileios Charilogis, Ioannis G. Tsoulos

Department of Informatics and Telecommunications, University of Ioannina, Greece

Abstract

The Particle Swarm Optimization (PSO) method is a global optimization technique based on the gradual evolution of a population of solutions called particles. The method evolves the particles based on both the best position of each of them in the past and the best position of the whole. Due to its simplicity, the method has found application in many scientific areas and for this reason, during the last years, many modifications have been presented. This paper introduces three modifications to the method that aim to reduce the required number of function calls while maintaining the accuracy of the method in locating the global minimum. These modifications affect important components of the method, such as how fast the particles change or even how the method is terminated. The above modifications were tested on a number of known universal optimization problems from the relevant literature and the results were compared with similar techniques.

Keywords: Optimization, Evolutionary techniques, Stochastic methods, Termination rules

1 Introduction

The global optimization problem is usually defined as:

$$x^* = \arg\min_{x \in S} f(x) \tag{1}$$

with S:

$$S = [a_1, b_1] \otimes [a_2, b_2] \otimes \ldots [a_n, b_n]$$

The function f is considered a continuous and differentiable function, formulated as $f : S \to R, S \subset R^n$. This problem finds application in a variety of objective problems in the real world, such as problems of physics [1, 2, 3], chemistry [4, 5, 6], economics [7, 8], medicine [9, 10] etc. Global optimization methods are grouped into two broad categories: deterministic and stochastic methods. The most common methods of the first category are the so called Interval methods [11, 12], where the set S is divided iteratively into subregions and some subregions that not contain the global solution are discarded using some pre-defined criteria. In stochastic methods, the finding of the global minimum is guided by randomness. In these methods there is no guarantee to find the global minimum but they constitute the vast majority of global optimization methods, mainly due to the simplicity in their implementation. There have been proposed many stochastic methods by various researchers such as Controlled Random Search methods [13, 14, 15], Simulated Annealing methods [16, 17, 18], Differential Evolution methods [19, 20], Particle Swarm Optimization methods [21, 22, 23], Ant Colony Optimization [24, 25], Genetic algorithms [26, 27, 28] etc. Also, many works have appeared utilizing the modern GPU processing units [29, 30, 31].

The method of Particle Swarm Optimization is a method based on a population of candidate solutions that also called particles. These particles have two basic characteristics: their position at any given time $\overrightarrow{x}$ and the speed $\overrightarrow{u}$ at which they move. The purpose of the method is to move the particles repetitively and their next position is calculated as a function not only of their position but also of the best position they had in the past, as well as the best position of the population. The method was successfully used in a variety of scientific and practical problems in areas such as physics [32, 33], chemistry [34, 35], medicine [36, 37], economics [38] etc. Due to its high popularity, the method has received a number of interventions in recent years, such as combination with the mutation mechanism [39, 40, 41], improved initialization of the velocity vector [42], hybrid techniques [43, 44, 45], parallel techniques [46, 47, 48], methods aim to improve the velocity calculation [49, 50, 51] etc. This text introduces three distinct modifications to the original method, which drastically improve the time required to find the total minimum by reducing the required number of function evaluations. These modifications cover a large part of the method: the speed calculation, a new method of avoiding running local search methods and a new adaptive termination rule. The proposed modifications were applied to a number of problems from the relevant literature and compared with similar techniques and the results are presented in a separate section.

The rest of this article is divided as follows: in section 2 the initial method and the proposed modifications are discussed, in section 3 the experiments are listed and finally, in section 4 some conclusions and guidelines for future improvements are presented.

2 Method description

The base algorithm of PSO and the proposed modifications are outlined in detail in the following subsections. The discussion starts with a new mechanism that calculates the velocity of the populations, continuous with a discarding procedure used to minimize the number of local searches performed, and ends with a discussion about the new stopping rule proposed here.

2.1 The base algorithm

The base algorithm is listed below with the periodical application of the local search method in order to enhance the estimation of the global minimum, ie. at every iteration a decision with probability p_l is made in order to apply a local search procedure to some of the particles. Usually, this probability is small, for example 0.05 (5%).

1. **Initialization**.
   (a) **Set** iter $= 0$ (iteration counter).
   (b) **Set** the number of particles m.
   (c) **Set** the maximum number of iterations allowed iter_{max}
   (d) **Set** the local search rate $p_l \in [0, 1]$.
   (e) **Initialize** randomly the positions of the m particles $x_1, x_2, ..., x_m$, with $x_i \in S \subset R^n$
   (f) **Initialize** randomly the velocities of the m particles $u_1, u_2, ..., u_m$, with $u_i \in S \subset R^n$
   (g) **For** $i = 1..m$ do $p_i = x_i$. The p_i vector are the best located values for every particle i.
   (h) **Set** $p_{\text{best}} = \arg\min_{i \in 1..m} f(x_i)$
2. **Termination Check**. Check for termination. If termination criteria are met then stop.
3. **For** $i = 1..m$ **Do**
   (a) **Update** the velocity u_i as a function of u_i, p_i and p_{best}
   (b) **Update** the position $x_i = x_i + u_i$
   (c) **Set** $r \in [0, 1]$ a random number. If $r \leq p_m$ then $x_i = \text{LS}(x_i)$, where $\text{LS}(x)$ is a local search procedure.
   (d) **Evaluate** the fitness of the particle i, $f(x_i)$
   (e) **If** $f(x_i) \leq f(p_i)$ then $p_i = x_i$
4. **End For**
5. **Set** $p_{\text{best}} = \arg\min_{i \in 1..m} f(x_i)$
6. **Set** iter $=$ iter $+ 1$.
7. **Goto** Step 2

2.2 Velocity calculation

The algorithm of subsection 2.1 calculates at every iteration the new position x_i is calculated using the old position x_i and the associated velocity u_i according to the scheme:

$$x_i = x_i + u_i \tag{2}$$

Typically the velocity is calculated as a combination of the old velocity and the best located values p_i and p_{best} and may be defined as:

$$u_i = \omega u_i + r_1 c_1 \left(p_i - x_i\right) + r_2 c_2 \left(p_{\text{best}} - x_i\right) \tag{3}$$

where

1. The parameters $r_1,\ r_2$ are random numbers with $r_1 \in [0,1]$ and $r_2 \in [0,1]$.
2. The constant number $c_1,\ c_2$ are in the range $[1,2]$.
3. The variable ω is called inertia, with $\omega \in [0,1]$.

The inertia was proposed by Shi and Eberhart [21]. They argued that high values of the inertia coefficient cause better exploration of the search area, while small values of this variable are used when we want to achieve better local research around promising areas for the global minimum. The value of the inertia factor generally starts with large values and decreases with repetition. This article proposes a new adaptive technique for the inertia parameter and this mechanism is compared against three others from the relevant literature.

2.2.1 Random inertia

The inertia calculation is proposed in [52] and it is defined as

$$\omega_{\text{iter}} = 0.5 + \frac{r}{2} \tag{4}$$

where r a random number with $r \in [0,1]$. This inertia calculation will be called I1 in the tables with the experimental results.

2.2.2 Linear time varying inertia (min version)

This inertia schema has been proposed in used in various studies [52, 53, 54] and it is defined as:

$$\omega_{\text{iter}} = \frac{\text{iter}_{\text{max}} - \text{iter}}{\text{iter}_{\text{max}}} \left(\omega_{\text{max}} - \omega_{\text{min}}\right) + \omega_{\text{min}} \tag{5}$$

where ω_{min} is the minimum value of inertia and ω_{max} the maximum value for inertia. This inertia calculation will be called I2 in the tables with the experimental results.

2.2.3 Linear time varying inertia (max version)

This method is proposed in [55, 56] and it is defined as:

$$\omega_{\text{iter}} = \frac{\text{iter}_{\text{max}} - \text{iter}}{\text{iter}_{\text{max}}} \left(\omega_{\text{min}} - \omega_{\text{max}}\right) + \omega_{\text{max}} \tag{6}$$

This inertia calculation will be called I3 in the tables with the experimental results.

2.2.4 Proposed technique

This calculation of inertia involves the number of iterations where the method manages to find a new minimum. In the first iterations and when the method has to do more exploration of the research area, the inertia will be great. When the method should focus on a minimum, then the inertia will decrease. For this reason, at every iteration iter the quantity

$$\delta^{(\text{iter})} = \left| \sum_{i=1}^{m} \left| f_i^{(\text{iter})} \right| - \sum_{i=1}^{m} \left| f_i^{(\text{iter}-1)} \right| \right| \tag{7}$$

is measured. In the first steps of the algorithm this quantity will change from repetition to repetition at a fast pace and at some point and then it will no longer change at the same rate or will be zero. Hence, we define a metric to model the changes in $\delta^{(\text{iter})}$ as

$$\zeta^{(\text{iter})} = \begin{cases} 1, & \delta^{(i)} = 0 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Using this observation two additional metrics are created, $S_\delta^{(\text{iter})}$ and $C_\delta^{(\text{iter})}$. The metric $S_\delta^{(\text{iter})}$ is given by

$$S_\delta^{(\text{iter})} = \sum_{i=1}^{\text{iter}} \zeta^{(i)} \tag{9}$$

and the metric C_δ is given by:

$$C_\delta^{(\text{iter})} = \frac{S_\delta^{(\text{iter})}}{\text{iter}} \tag{10}$$

The following definition for the inertia calculation is proposed:

$$\omega_{\text{iter}} = \omega_{max} - \frac{\text{iter}}{C_\delta^{(\text{iter})}} \left(\omega_{\text{max}} - \omega_{\text{min}}\right) \tag{11}$$

This mechanism will be called IP in the tables with the experimental results.

2.3 The discarding procedure

The method in each iteration performs under condition of a series of local searches. However, these searches will often either lead to local minima already found or locate values far below the global minimum. This means that much of the computing time will be wasted on actions that could have been avoided. In order to be able to group points that would lead by local search to the same local minimum, we introduce the concept of cluster, which refers to a set of points that are believed, under some asymptotic considerations, to belong to the same region of attraction of the function. The region of attraction for a local minimum x^* is defined as:

$$A\left(x^*\right) = \left\{x : \ x \in S \subset R^n, \ \mathrm{LS}(x) = x^*\right\} \tag{12}$$

where $\mathrm{LS}(x)$ is a local search procedure that starts from a given point x and terminates when a local minimum is discovered. The discarding procedure suggested here prevents the method from starting a local search from a point x if that point belongs to the same region of attraction with other points. This procedure is composed by two two major parts:

1. The first part is the so called typical distance, which is a measure calculated after every local search and it is given by

   $$r_C = \frac{1}{M}\sum_{i=1}^{M}\left\|x_i - x_{iL}\right\| \tag{13}$$

   where the local search procedure $\mathrm{LS}(x)$ initiates from x_i and x_{iL} is the outcome of $\mathrm{LS}\left(x_i\right)$. This measure has been used also in [57]. If a point x is close enough to an already discovered local minima then it is highly possible that the point belongs to the so called region of attraction of the minima.

2. The second part is a check using the gradient values between a candidate starting point and an already discovered local minimum. The function value $f(x)$ near to some local minimum z can be calculated using:

   $$f(x) \simeq f(z) + \frac{1}{2}\left(x-z\right)^T B\left(x-z\right) \tag{14}$$

   where B is the Hessian matrix at the minimum z. By taking gradients in both sides of Equation 14 we obtain:

   $$\nabla f(x) \simeq B\left(x-z\right) \tag{15}$$

   Of course equation 15 holds for any other point y near to z

   $$\nabla f(y) \simeq B\left(y-z\right) \tag{16}$$

   By subtracting the equation 16 from 15 and by multiplying with $(x-y)^T$ we have the following equation:

   $$\left(x-y\right)^T\left(\nabla f(x) - \nabla f(y)\right) \simeq \left(x-y\right)^T B\left(x-y\right)^T > 0 \tag{17}$$

Hence, a candidate start point x can be rejected if

$$\|x-z\| \leq r_C \text{ AND } (x-y)^T\left(\nabla f(x)-\nabla f(z)\right) \tag{18}$$

for any already discovered local minimum z.

2.4 Stopping rule

A common used way to terminate a global optimization method is to utilize a maximum number of allowed iterations iter_{max}, i.e. stop when $\text{iter} \geq \text{iter}_{\text{max}}$. Although it is a simple criterion but is not an efficient one since, if iter_{max} is tool small, then the algorithm will terminate without locating the global optimum. Also, when iter_{max} is too high, the optimization algorithm will spend computation time in unnecessary function calls. In this paper, a new termination rule is proposed to terminate the PSO process and it is compared against two other methods used in various optimization methods.

2.4.1 Ali's stopping method

A method proposed in the work of Ali and Kaelo[58], where at every generation the measure

$$\alpha=\left|f_{\text{max}}-f_{\text{min}}\right| \tag{19}$$

is calculated. The f_{max} stands for the maximum function value of the population and f_{min} represents the lowest function value of the population. The method will terminate when

$$\alpha \leq \epsilon \tag{20}$$

where ϵ is a predefined small positive value, for example $\epsilon=10^{-3}$.

2.4.2 Doublebox method

The second method utilized is a method initially proposed [59]. In this method we denote with $\sigma^{(iter)}$ the variance of f_{min} calculated at iteration iter. If the algorithm can not locate a new lower value for f_{min} for a number of iterations, then the global minimum has already located and the algorithm should terminate, i.e. terminate when

$$\sigma^{(\text{iter})} \leq \frac{\sigma^{\left(\text{iter}_{\text{last}}\right)}}{2} \tag{21}$$

where $\text{iter}_{\text{last}}$ stands for the last iteration where a new lower value for f_{min} was discovered.

2.4.3 Proposed technique

In the proposed termination technique in each iteration k the difference between the current best value and the previous best value is measured, ie the difference $\left|f_{\text{min}}^{(k)}-f_{\text{min}}^{(k-1)}\right|$. If this difference is zero for a series of predefined number of iterations k_{max}, then the method terminates.

3 Experiments

To measure the effect of the proposed modifications on the original method, a series of experiments were performed on test functions from the relevant literature. [60, 61] and they have been used widely by various researchers [62, 63, 64, 65]. The experiments evaluated both the effect of the new method of calculating inertia, as well as the criterion for avoiding local minima as well as the new termination rule. The experiments were recorded in separate tables, so that it is more possible to understand the effect of each modification separately.

3.1 Test functions

The definition of the test functions used are given below

- **Bf1** (Bohachevsky 1) function:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$$

with $x \in [-100, 100]^2$.

- **Bf2** (Bohachevsky 2) function:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$$

  with $x \in [-50, 50]^2$.

- **Branin** function: $f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$ with $-5 \le x_1 \le 10,\ 0 \le x_2 \le 15$. with $x \in [-10, 10]^2$.

- **CM** function:

$$f(x) = \sum_{i=1}^{n} x_i^2 - \frac{1}{10}\sum_{i=1}^{n}\cos(5\pi x_i)$$

  where $x \in [-1, 1]^n$. In the current experiments we used $n = 4$.

- **Camel** function:

$$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, \quad x \in [-5, 5]^2$$

- **Easom** function:

$$f(x) = -\cos(x_1)\cos(x_2)\exp\left((x_2 - \pi)^2 - (x_1 - \pi)^2\right)$$

  with $x \in [-100, 100]^2$.

- **Exponential** function, defined as:

$$f(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right), \quad -1 \le x_i \le 1$$

  In the current experiments we used this function with $n = 2, 4, 8, 16, 32$.

- **Goldstein and Price function**

$$\begin{aligned} f(x) &= \Big[1+(x_1+x_2+1)^2 \\ &\quad \left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\Big] \times \\ &\quad [30+(2x_1-3x_2)^2 \\ &\quad \left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)] \end{aligned}$$

  With $x \in [-2,2]^2$.

- **Griewank2** function:

$$f(x) = 1 + \frac{1}{200}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2} \frac{\cos(x_i)}{\sqrt{(i)}}, \quad x \in [-100,100]^2$$

- **Gkls** function. $f(x) = \mathrm{Gkls}(x,n,w)$, is a function with w local minima, described in [67] with $x \in [-1,1]^n$ and n a positive integer between 2 and 100. The value of the global minimum is -1 and in our experiments we have used $n = 2,3$ and $w = 50,\ 100$.

- **Hansen** function: $f(x) = \sum_{i=1}^{5} i\cos\left[(i-1)x_1+i\right]\sum_{j=1}^{5} j\cos\left[(j+1)x_2+j\right]$, $x \in [-10,10]^2$.

- **Hartman 3** function:

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}\left(x_j-p_{ij}\right)^2\right)$$

  with $x \in [0,1]^3$ and $a = \begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}$, $c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix}$ and

$$p = \begin{pmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{pmatrix}$$

- **Hartman 6** function:

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}\left(x_j-p_{ij}\right)^2\right)$$

with $x \in [0,1]^6$ and $a = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}$, $c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix}$

and

$$p = \begin{pmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{pmatrix}$$

- **Potential** function. The molecular conformation corresponding to the global minimum of the energy of N atoms interacting via the Lennard-Jones potential[68] is used a test function here and it is defined by:

$$V_{LJ}(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \tag{22}$$

  For our experiments we used: $N = 3,\ 4,\ 5$

- **Rastrigin** function.

$$f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2), \quad x \in [-1,1]^2$$

- ***Rosenbrock*** *function.*

$$f(x) = \sum_{i=1}^{n-1} \left(100 \left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \right), \quad -30 \le x_i \le 30.$$

  In our experiments we used the values $n = 4,\ 8,\ 16$.

- **Shekel 7** function.

$$f(x) = -\sum_{i=1}^{7} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$$

with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 3 & 5 & 3 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \end{pmatrix}$.

- **Shekel 5** function.

$$f(x) = -\sum_{i=1}^{5} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$$

with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \end{pmatrix}$.

- **Shekel 10** function.

$$f(x) = -\sum_{i=1}^{10} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$$

with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.6 \end{pmatrix}$.

- **Sinusoidal** function:

$$f(x) = -\left(2.5\prod_{i=1}^{n}\sin\left(x_i - z\right) + \prod_{i=1}^{n}\sin\left(5\left(x_i - z\right)\right)\right), \quad 0 \le x_i \le \pi.$$

The case of $n = 4, 8, 16, 32$ and $z = \frac{\pi}{6}$ was used in the experimental results.

- **Test2N** function:

$$f(x) = \frac{1}{2}\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i, \quad x_i \in [-5,5].$$

The function has 2^n in the specified range and in our experiments we used $n = 4, 5, 6, 7$.

- **Test30N** function:

$$f(x) = \frac{1}{10}\sin^2\left(3\pi x_1\right)\sum_{i=2}^{n-1}\left(\left(x_i - 1\right)^2\left(1 + \sin^2\left(3\pi x_{i+1}\right)\right)\right) + \left(x_n - 1\right)^2\left(1 + \sin^2\left(2\pi x_n\right)\right)$$

with $x \in [-10,10]$, with 30^n local minima in the search space. For our experiments we used $n = 3, 4$.

3.2 Experimental setup

All the experiments have been performed 30 times using a different seed for the random number generator each. The code has been implemented in ANSI C++ and the well - known function drand48() was used to produce random numbers. The local search method used as BFGS method [66]. All the parameters used in the conducted experiments are listed in Table 1.

Table 1: Values for the experimental parameters.

PARAMETER	VALUE
m	100
iter_{max}	100
p_l	0.05
c_1	1.0
c_2	1.0
ω_{min}	0.4
ω_{max}	0.9
ϵ	0.001
k_{max}	15

3.3 Experimental results

For every stopping rule, two tables are listed: the first one contains experiments with the relevant stopping rule without the gradient discarding procedure and the second table contains experiments with the gradient discarding procedure enabled. The numbers in table cells stand for average function calls. The fraction in parentheses denotes the fraction of runs where the global optimum was found. The absence of these fractions means that the global minimum was discovered in every run (100% success). The experimental results using the stopping rule of Equation 20 are listed in Tables 2 and 3. The experimental results for the Double Box stopping rule of Equation 21 are listed in Tables 4 and 5. Finally, for the proposed stopping rule, the results are listed in the Tables 6 and 7. The above results lead to a number of observations:

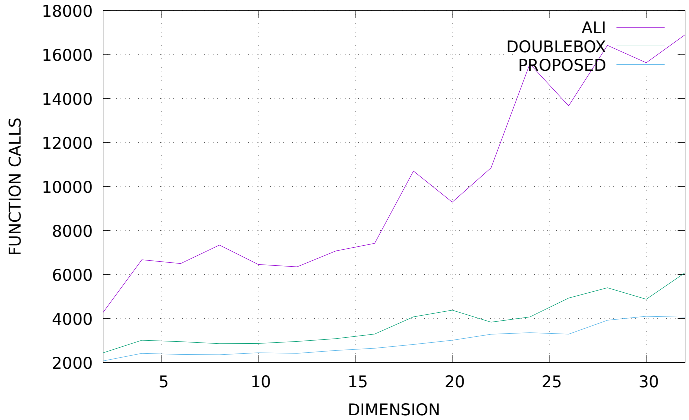
1. The PSO method is a robust method and this is evident by the high success rate in finding the global minimum, although the number of particles used was relatively low (100).

2. The proposed inertia calculation method as defined in Equation 11 achieves a significant reduction in the number of calls between 11 and 25% depending on the termination criterion used. However, the presence of the gradient check mechanism of the Equation 18 nullifies any gain of the method, as the rejection criterion significantly reduces the number of calls regardless of the inertia calculation mechanism used.

3. The local optimization avoidance mechanism of the gradient check drastically reduces the required number of calls for each termination criterion while maintaining the success rate of the method at extremely high levels.

4. The proposed termination criterion is significantly superior to the other two with which the comparison was made. Also, if the termination criterion is combined with the mechanism for avoiding local optimizations, then the gain in number of calls grows even more.

Figure 1: Experiments with the SINU function for a series of problem dimension from $n = 2$ to $n = 32$.



To show the effect of the proposed termination method, an additional experiment was performed in which the dimension of the sinus problem increased from 2 to 32 and in each case all three termination techniques were tested. The result of this experiment is graphically represented in Figure 1. This graph shows that the doublebox method is significantly superior to the ali method but, of course, the new proposed method further reduces the required number of function calls.

4 Conclusions

In the current work, three new modifications of the PSO method for locating the global minimum of continuous and differentiable functions were presented. The first modification alters the population velocity calculation in an attempt to cause large changes in velocity when the method is in its infancy and constantly finds new local minima and small velocity changes when the method is to be centered around a promising area of a global minimum. The second modification limits the number of local searches performed by the method through an asymptotic criterion based on derivative computation. The third modification introduces a new termination criterion based on the observation that the method from some iteration onwards will not be able to detect a new minimum and therefore its termination should be considered. All of these modifications have low computational requirements.

The experimental results showed that the second and third modifications drastically reduced the required number of function calls without affecting the performance of the method. In addition, the first modification reduces the number of required calls, but only when the criterion for avoiding local minimization is not present.

Table 2: Experiments with the Ali stopping rule, without gradient check.

FUNCTION	I1	I2	I3	IP
BF1	24929	22874	18739	22088
BF2	24043	22254	17172	20743
BRANIN	17691	16205	13397	12471
CM4	20117	22568	26867	14941
CAMEL	19474	17813	14461	13492
EASOM	13327	13106	9969	9212
EXP2	6339	8243	7853	3501
EXP4	7816	10066	10900	4458
EXP8	8667	10937	13126	4761
EXP16	8748	11402	15754	5098
EXP32	9567	12323	18189	5471
GKLS250	10907	12562	9673	8552
GKLS2100	12960	13403	9930	9541
GKLS350	15410(0.97)	14722	10542	9298
GKLS3100	16639	14495	10412	13075(0.97)
GOLDSTEIN	20437	22877	16410	8935
GRIEWANK2	27620	24230	18473	20133
HANSEN	21513	20279	16326	15046
HARTMAN3	16233	17152	12305	6511
HARTMAN6	47038	48947	46852	23431
POTENTIAL3	31684	32175	36930	24463
POTENTIAL4	184602	181231	168962	129267
POTENTIAL5	74508	70519	76890	54042
RASTRIGIN	23574	20865	15596	16198
ROSENBROCK4	145178	161136	160341	129891
ROSENBROCK8	95290	97035	96687	80408
ROSENBCROK16	118614	116454	115122	97004
SHEKEL5	27458	27088	25927	18036
SHEKEL7	27521	27271	25967	18805
SHEKEL10	29699(0.97)	28082	25511	20823
TEST2N4	26740	27050	22905	19495
TEST2N5	20243(0.97)	20290	17729	16024(0.97)
TEST2N6	33118	33366	30118	25235(0.93)
TEST2N7	23266(0.90)	22804	21294	18218(0.90)
SINU4	17035	20487	18971	11079
SINU8	22827	27176	27732	12379
SINU16	31055	35998	42984	15692
SINU32	44736(0.97)	51624	82114	25991
TEST30N3	18733	20119	17803	17543
TEST30N4	20348	22191	20679	20085
TOTAL	1365704(0.99)	1399419	1367612	1001436(0.99)

Table 3: Experiments with the Ali stopping rule, with the gradient check enabled.

FUNCTION	**I1**	**I2**	**I3**	**IP**
BF1	9709	8918	9531	10932
BF2	10196	9588	9089	10730
BRANIN	10718	9597	9813	9501
CM4	6242	7503	12531	6985
CAMEL	10422	9306	8491	9624
EASOM	11565	11366	8497	8196
EXP2	3364	4443	4558	1926
EXP4	3558	4767	6023	2122
EXP8	3716	4787	7753	2186
EXP16	3784	5076	9696	2211
EXP32	4137	5698	11379	2323
GKLS250	5917	7080	7517	5273
GKLS2100	6843	8261	7449	7296
GKLS350	6845	8076	7833	5881(0.97)
GKLS3100	10290(0.93)	10187	7828	8066(0.97)
GOLDSTEIN	7977	9035	8505	4381
GRIEWANK2	12567	12222	12000	12037
HANSEN	13441	13360	11876	10818
HARTMAN3	9758	9548	8123	4114
HARTMAN6	12893(0.90)	12889(0.93)	22309	10126(0.93)
POTENTIAL3	17912	16420	21904	15969
POTENTIAL4	73629	64886	95707	67084
POTENTIAL5	40585	35239	47807	33661
RASTRIGIN	11305	10101	11141	10046
ROSENBROCK4	18115	21919	38407	43093
ROSENBROCK8	12869	14192	31923	25405
ROSENBCROK16	12096	13023	38486	23165
SHEKEL5	10347	11466	14446	11802
SHEKEL7	11511	10521	13944	10399
SHEKEL10	10834	10842	13785	12253
TEST2N4	11133	10869	12161	11546
TEST2N5	10923(0.97)	10315	10868	11072(0.97)
TEST2N6	12331(0.97)	12345	14123	15652
TEST2N7	11342(0.93)	11354	12118	12370(0.93)
SINU4	7724	9845	12294	6575
SINU8	8468	10969	18122	5382
SINU16	9334	13213	31589	9294
SINU32	13290	17502(0.97)	63111(0.97)	14959
TEST30N3	12675	12954	12472	12482
TEST30N4	13964	14903	14999	15389
TOTAL	494119(0.99)	504585(0.99)	720208(0.99)	502326(0.99)

Table 4: Experiments with the double box stopping rule, without gradient check.

FUNCTION	**I1**	**I2**	**I3**	**IP**
BF1	6807	6866	6712	6757
BF2	6102	6150	6057	6207
BRANIN	4551	4596	4470	4435
CM4	9814	10101	9580	9342
CAMEL	5055	5202	4897	5004
EASOM	2975	2788	3014	3000
EXP2	4436	4541	4377	4543
EXP4	5443	5562	5331	5290
EXP8	5682	5754	5614	5504
EXP16	5707	5799	5638	5526
EXP32	5871	5797	5769	5659
GKLS250	3973	3906	3971	3921
GKLS2100	4009	3862	4073	3958
GKLS350	4558	3965	4525(0.97)	4266
GKLS3100	4701(0.87)	4266	4361(0.90)	4465
GOLDSTEIN	10259	9145	7945	7625
GRIEWANK2	5932	6194	5700	5915
HANSEN	6386	6260	5688(0.97)	5874
HARTMAN3	4681	4694	4625	4675
HARTMAN6	14245	14091	13793	13825
POTENTIAL3	7219	7206	7532	7234
POTENTIAL4	38053	37924	38421	38897
POTENTIAL5	15196	14459	15708	15358
RASTRIGIN	5915	5797	5944(0.83)	5844
ROSENBROCK4	91574	101485	117512	76367
ROSENBROCK8	66648	61974	58831	41591
ROSENBCROK16	62029	54550	63406	55800
SHEKEL5	9119	10271(0.97)	8975	8538
SHEKEL7	9197	9831	9638	8732
SHEKEL10	10417	10449	9373	9721(0.90)
TEST2N4	8512	8272	8884	7992
TEST2N5	5793	5704	5511(0.90)	5515
TEST2N6	9797(0.93)	9731	9657(0.83)	9666(0.97)
TEST2N7	6435(0.80)	6659	6713(0.77)	5990(0.87)
SINU4	7567	7774	7334	7063
SINU8	9882	10083	9643	9331
SINU16	12750	12947	12569	12207
SINU32	20164	21112	19684(0.90)	19239
TEST30N3	6388	7942	5934	5855
TEST30N4	7611	9251	6385	8284
TOTAL	**531453(0.99)**	**532690(0.99)**	**543794(0.98)**	**475015(0.99)**

Table 5: Experiments with the double box stopping rule, with the gradient check enabled.

FUNCTION	I1	I2	I3	IP
BF1	3296	3038	3063	3003
BF2	2922	2762	2863	2845
BRANIN	2562	2641	2538	2564
CM4	3569	4277	2944	3230
CAMEL	2646	2854	2467	2577
EASOM	2490	2390	2479	2464
EXP2	2377	2489	2261	2315
EXP4	2456	2669	2282	2389
EXP8	2429	2671	2268	2385
EXP16	2358	2569	2227	2326
EXP32	2337	2533	2248	2312
GKLS250	2394	2535	2274	2321
GKLS2100	2384	2511	2267	2333
GKLS350	2492	2410	2212	2339
GKLS3100	2800(0.90)	2708	2648(0.83)	2571
GOLDSTEIN	3161	3701	3166	2799
GRIEWANK2	3910	4520	3543	3641
HANSEN	4409	4268	3755	4325
HARTMAN3	2423	2518	2374	2425
HARTMAN6	3913	4390	4199(0.93)	3700
POTENTIAL3	3951	4093	4482	4021
POTENTIAL4	18555	19559	19506	18691
POTENTIAL5	8771	8397	9677	9154
RASTRIGIN	3111	3244(0.97)	3031	3146
ROSENBROCK4	9729	12980	11453	8587
ROSENBROCK8	4987	6738	4688	5512
ROSENBCROK16	4410	5939	4553	4002
SHEKEL5	3906	4095	3203	3495
SHEKEL7	3119	3965	2950	3528
SHEKEL10	3497(0.97)	4464	3142(0.97)	3353
TEST2N4	3468(0.97)	4059	4167	3881(0.93)
TEST2N5	3318(0.97)	3786	2926(0.90)	3157(0.97)
TEST2N6	4523(0.93)	5046(0.93)	5537(0.83)	4066(0.97)
TEST2N7	3364(0.80)	4191(0.90)	4183(0.80)	3315(0.87)
SINU4	3173	3807	2610	3004
SINU8	3055	3742	2592	2857
SINU16	3160	3746	3854	3290
SINU32	6613	7377	6327	6450
TEST30N3	5129	6367	5605	4451
TEST30N4	5649	6441	6074	5543
TOTAL	**162816(0.99)**	**182490(0.99)**	**164638(0.98)**	**158367(0.99)**

Table 6: Experiments with the proposed stopping rule, without the gradient check.

FUNCTION	I1	I2	I3	IP
BF1	5305	5326	5240	5209
BF2	4760	4841	4750	4856
BRANIN	3599	3703	3520	3443
CM4	7674	7835	7430	7057
CAMEL	3996	4131	3864	3825
EASOM	2370	2292	2425	2478
EXP2	3528	3613	3455	3675
EXP4	4292	4350	4178	4020
EXP8	4579	4632	4515	4278
EXP16	4576	4637	4505	4236
EXP32	4692	4771	4588	4296
GKLS250	3105	3065	3115	3024
GKLS2100	3193	3049	3193	3099
GKLS350	3308	3000	3560	3401
GKLS3100	2935(0.97)	2777	3158(0.83)	3088
GOLDSTEIN	5534	5595	5332	5265
GRIEWANK2	4225	4332	4413	4489
HANSEN	3865	3762	3824	3769
HARTMAN3	3724	3770	3714	3705
HARTMAN6	11901(0.97)	11829(0.97)	11386	10573
POTENTIAL3	5910	5850	6134	6501
POTENTIAL4	30880	30570	31180	30682
POTENTIAL5	12021	11643	12521	13475
RASTRIGIN	4583	4595	4625	4360
ROSENBROCK4	58299	61266	55759	35517
ROSENBROCK8	31778	30888	30989	22055
ROSENBCROK16	32719	30503	30957	24478
SHEKEL5	6806	7047(0.97)	6636	6233
SHEKEL7	6807	7001	6626	6270
SHEKEL10	6774	6987	6583	6534
TEST2N4	6111	6127	5909	5893
TEST2N5	4455(0.97)	4558	4372(0.97)	4271(0.93)
TEST2N6	7446(0.97)	7419	7218(0.87)	7122(0.93)
TEST2N7	4992(0.90)	5057	4888(0.83)	4680(0.90)
SINU4	5948	6043	5750	5229
SINU8	7965	8095	7778	6963
SINU16	10121	10252	9968	9219
SINU32	16093	16509	15663	14478
TEST30N3	4331	4953	4230	3957
TEST30N4	6290	6341	4288	4717
TOTAL	**361490(0.99)**	**363013(0.99)**	**352239(0.99)**	**310420(0.99)**

Table 7: Experiments with the proposed stopping rule, with the gradient check enabled.

FUNCTION	I1	I2	I3	IP
BF1	2276	2379	2266	2250
BF2	2157	2274	2098	2191
BRANIN	2132	2178	2051	2170
CM4	3098	3717	2538	2791
CAMEL	2198	2335	1974	2058
EASOM	2007	2011	2031	2084
EXP2	1952	2030	1842	1861
EXP4	2046	2266	1877	1909
EXP8	1990	2240	1849	1879
EXP16	1944	2110	1828	1838
EXP32	1953	2126	1859	1867
GKLS250	1982	2079	1850	1900
GKLS2100	1983	2064	1859	1891
GKLS350	1882	1944	1793	1831
GKLS3100	1898	1909(0.97)	1850(0.83)	1833(0.83)
GOLDSTEIN	2523	2670	2110	2164
GRIEWANK2	2893	2885	2791	2681
HANSEN	2766	2879	2731	2804
HARTMAN3	1988	2093	1949	2015
HARTMAN6	3366	3871(0.97)	2767	3133
POTENTIAL3	3312	3487	3613	3892
POTENTIAL4	15392	16390	16223	17497
POTENTIAL5	7109	7104	7732	8477
RASTRIGIN	2591	2648	2474	2732
ROSENBROCK4	8023	12179	4433	6025
ROSENBROCK8	4376	6081	2721	3314
ROSENBCROK16	3643	4954	2746	2485
SHEKEL5	2849	3296	2274	2390
SHEKEL7	2696	3294	2262	2283
SHEKEL10	2624	3251	2338(0.93)	2359
TEST2N4	2536	2637	2427	2782
TEST2N5	2266(0.97)	2336(0.97)	2163(0.90)	2342(0.90)
TEST2N6	2724(0.93)	2832	2694(0.80)	3133(0.90)
TEST2N7	2283(0.80)	2370	2279(0.80)	2585(0.90)
SINU4	2789	3245	2228	2436
SINU8	2601	3151	2233	2348
SINU16	2721	3086	2443	2624
SINU32	4652	5135	4086	4089
TEST30N3	3031	3349	3007	2562
TEST30N4	3747	3797	3250	3237
TOTAL	**126999(0.99)**	**142682(0.99)**	**115539(0.98)**	**122742(0.99)**

References

[1] L. Yang, D. Robin, F. Sannibale, C. Steier, W. Wan, Global optimization of an accelerator lattice using multiobjective genetic algorithms, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **609**, pp. 50-57, 2009.

[2] E. Iuliano, Global optimization of benchmark aerodynamic cases using physics-based surrogate models, Aerospace Science and Technology **67**, pp.273-286, 2017.

[3] Q. Duan, S. Sorooshian, V. Gupta, Effective and efficient global optimization for conceptual rainfall-runoff models, Water Resources Research **28**, pp. 1015-1031 , 1992.

[4] S. Heiles, R. L. Johnston, Global optimization of clusters using electronic structure methods, Int. J. Quantum Chem. **113**, pp. 2091– 2109, 2013.

[5] W.H. Shin, J.K. Kim, D.S. Kim, C. Seok, GalaxyDock2: Protein–ligand docking using beta-complex and global optimization, J. Comput. Chem. **34**, pp. 2647– 2656, 2013.

[6] A. Liwo, J. Lee, D.R. Ripoll, J. Pillardy, H. A. Scheraga, Protein structure prediction by global optimization of a potential energy function, Biophysics **96**, pp. 5482-5485, 1999.

[7] Zwe-Lee Gaing, Particle swarm optimization to solving the economic dispatch considering the generator constraints, IEEE Transactions on **18** Power Systems, pp. 1187-1195, 2003.

[8] C. D. Maranas, I. P. Androulakis, C. A. Floudas, A. J. Berger, J. M. Mulvey, Solving long-term financial planning problems via global optimization, Journal of Economic Dynamics and Control **21**, pp. 1405-1425, 1997.

[9] Eva K. Lee, Large-Scale Optimization-Based Classification Models in Medicine and Biology, Annals of Biomedical Engineering **35**, pp 1095-1109, 2007.

[10] Y. Cherruault, Global optimization in biology and medicine, Mathematical and Computer Modelling **20**, pp. 119-132, 1994.

[11] M.A. Wolfe, Interval methods for global optimization, Applied Mathematics and Computation **75**, pp. 179-206, 1996.

[12] T. Csendes and D. Ratz, Subdivision Direction Selection in Interval Methods for Global Optimization, SIAM J. Numer. Anal. **34**, pp. 922–938, 1997.

[13] W. L. Price, Global optimization by controlled random search, Journal of Optimization Theory and Applications **40**, pp. 333-348, 1983.

[14] Ivan Křivý, Josef Tvrdík, The controlled random search algorithm in optimizing regression models, Computational Statistics & Data Analysis **20**, pp. 229-234, 1995.

[15] M.M. Ali, A. Törn, and S. Viitanen, A Numerical Comparison of Some Modified Controlled Random Search Algorithms, Journal of Global Optimization **11**,pp. 377–385,1997.

[16] S. Kirkpatrick, CD Gelatt, , MP Vecchi, Optimization by simulated annealing, Science **220**, pp. 671-680, 1983.

[17] L. Ingber, Very fast simulated re-annealing, Mathematical and Computer Modelling **12**, pp. 967-973, 1989.

[18] R.W. Eglese, Simulated annealing: A tool for operational research, Simulated annealing: A tool for operational research **46**, pp. 271-281, 1990.

[19] R. Storn, K. Price, Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization **11**, pp. 341-359, 1997.

[20] J. Liu, J. Lampinen, A Fuzzy Adaptive Differential Evolution Algorithm. Soft Comput **9**, pp.448–462, 2005.

[21] J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.

[22] Riccardo Poli, James Kennedy kennedy, Tim Blackwell, Particle swarm optimization An Overview, Swarm Intelligence **1**, pp 33-57, 2007.

[23] Ioan Cristian Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters **85**, pp. 317-325, 2003.

[24] M. Dorigo, M. Birattari and T. Stutzle, Ant colony optimization, IEEE Computational Intelligence Magazine **1**, pp. 28-39, 2006.

[25] K. Socha, M. Dorigo, Ant colony optimization for continuous domains, European Journal of Operational Research 185, pp. 1155-1173, 2008.

[26] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, Massachussets, 1989.

[27] Z. Michaelewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer - Verlag, Berlin, 1996.

[28] S.A. Grady, M.Y. Hussaini, M.M. Abdullah, Placement of wind turbines using genetic algorithms, Renewable Energy **30**, pp. 259-270, 2005.

[29] Y. Zhou and Y. Tan, "GPU-based parallel particle swarm optimization," 2009 IEEE Congress on Evolutionary Computation, pp. 1493-1500, 2009.

[30] L. Dawson and I. Stewart, "Improving Ant Colony Optimization performance on the GPU using CUDA," 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 1901-1908, doi: 10.1109/CEC.2013.6557791.

[31] Barkalov, K., Gergel, V. Parallel global optimization on GPU. J Glob Optim 66, 3–20 (2016).

[32] Anderson Alvarenga de Moura Meneses, Marcelo Dornellas, Machado Roberto Schirru, Particle Swarm Optimization applied to the nuclear reload problem of a Pressurized Water Reactor, Progress in Nuclear Energy **51**, pp. 319-326, 2009.

[33] Ranjit Shaw, Shalivahan Srivastava, Particle swarm optimization: A new tool to invert geophysical data, Geophysics **72**, 2007.

[34] C. O. Ourique, E.C. Biscaia, J.C. Pinto, The use of particle swarm optimization for dynamical analysis in chemical processes, Computers & Chemical Engineering **26**, pp. 1783-1793, 2002.

[35] H. Fang, J. Zhou, Z. Wang et al, Hybrid method integrating machine learning and particle swarm optimization for smart chemical process operations, Front. Chem. Sci. Eng. **16**, pp. 274–287, 2022.

[36] M.P. Wachowiak, R. Smolikova, Yufeng Zheng, J.M. Zurada, A.S. Elmaghraby, An approach to multimodal biomedical image registration utilizing particle swarm optimization, IEEE Transactions on Evolutionary Computation **8**, pp. 289-301, 2004.

[37] Yannis Marinakis. Magdalene Marinaki, Georgios Dounias, Particle swarm optimization for pap-smear diagnosis, Expert Systems with Applications **35**, pp. 1645-1656, 2008.

[38] Jong-Bae Park, Yun-Won Jeong, Joong-Rin Shin, Kwang Y. Lee, An Improved Particle Swarm Optimization for Nonconvex Economic Dispatch Problems, IEEE Transactions on Power Systems **25**, pp. 156-16**216**6, 2010.

[39] A. Stacey, M. Jancic, I. Grundy, Particle swarm optimization with mutation, In: 2003 Congress on Evolutionary Computation, 2003. CEC '03., pp. 1425-1430, 2003.

[40] M. Pant, R. Thangaraj, A. Abraham, Particle Swarm Optimization Using Adaptive Mutation, In: 2008 19th International Workshop on Database and Expert Systems Applications, pp. 519-523, 2008.

[41] N. Higashi, H. Iba, Particle swarm optimization with Gaussian mutation, In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706), pp. 72-79, 2003.

[42] A. Engelbrecht, "Particle swarm optimization: Velocity initialization," 2012 IEEE Congress on Evolutionary Computation, pp. 1-8, 2012.

[43] B. Liu, L. Wang, Y.H. Jin, F. Tang, D.X. Huang, Improved particle swarm optimization combined with chaos, Chaos Solitons and Fractals **25**, pp. 1261-1271, 2005.

[44] X.H. Shi, Y.C. Liang, H.P. Lee, C. Lu, L.M. Wang, An improved GA and a novel PSO-GA based hybrid algorithm, Information Processing Letters **93**, pp. 255-261, 2005.

[45] Harish Garg, A hybrid PSO-GA algorithm for constrained optimization problems, Applied Mathematics and Computation **274**, pp. 292-305, 2016.

[46] J. F. Schutte, J. A. Reinbolt, B. J. Fregly, R. T. Haftka, A. D. George, Parallel global optimization with the particle swarm algorithm, Int. J. Numer. Meth. Engng. **61**, pp. 2296-2315, 2004.

[47] B-Il Koh, A.D. George, R.T. Haftka, B.J. Fregly, Parallel asynchronous particle swarm optimization. Int. J. Numer. Meth. Engng., **67**, pp. 578-595, 2006.

[48] G. Venter, J. Sobieszczanski-Sobieski, Parallel Particle Swarm Optimization Algorithm Accelerated by Asynchronous Evaluations, Journal of Aerospace Computing, Information, and Communication **3**, pp. 123-137, 2006.

[49] Z.L. Gaing, Particle swarm optimization to solving the economic dispatch considering the generator constraints, IEEE Transactions on Power Systems **18**, pp. 1187-1195, 2003.

[50] X. Yang, Jinsha Yuan, Jiangy Yuan, H. Mao, A modified particle swarm optimizer with dynamic adaptation, Applied Mathematics and Computation **189**, pp. 1205-1213, 2007.

[51] Y. Jiang, T. Hu, C. Huang, X. Wu, An improved particle swarm optimization algorithm, Applied Mathematics and Computation **193**, pp. 231-239, 2007.

[52] R.C. Eberhart, Y.H. Shi, Tracking and optimizing dynamic systems with particle swarms, in: Congress on Evolutionary Computation, Korea, 2001.

[53] Y.H. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Congress on Evolutionary Computation, Washington DC, USA, 1999.

[54] Y.H. Shi, R.C. Eberhart, Experimental study of particle swarm optimization, in: SCI2000 Conference, Orlando, 2000.

[55] Y. Zheng, L. Ma, L. Zhang, J. Qian, Empirical study of particle swarm optimizer with an increasing inertia weight, in: IEEE Congress on Evolutionary Computation, 2003.

[56] Y. Zheng, L. Ma, L. Zhang, J. Qian, On the convergence analysis and param- eter selection in particle swarm optimization, in: Proceedings of the Second International Conference on Machine Learning and Cybernetics, 2003.

[57] Ioannis G. Tsoulos, Isaac E. Lagaris, MinFinder: Locating all the local minima of a function, Computer Physics Communications **174**, pp. 166-179, 2006.

[58] M.M. Ali, P. Kaelo, Improved particle swarm algorithms for global optimization, Applied Mathematics and Computation **196**, pp. 578–593, 2008.

[59] I.G. Tsoulos, Modifications of real code genetic algorithm for global optimization, Applied Mathematics and Computation **203**, pp. 598-607, 2008.

[60] M. Montaz Ali, Charoenchai Khompatraporn, Zelda B. Zabinsky, A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems, Journal of Global Optimization **31**, pp 635-672, 2005.

[61] C.A. Floudas, P.M. Pardalos, C. Adjiman, W. Esposoto, Z. Gümüs, S. Harding, J. Klepeis, C. Meyer, C. Schweiger, Handbook of Test Problems in Local and Global Optimization, Kluwer Academic Publishers, Dordrecht, 1999.

[62] M.M. Ali and P. Kaelo, Improved particle swarm algorithms for global optimization, Applied Mathematics and Computation **196**, pp. 578-593, 2008.

[63] H. Koyuncu, R. Ceylan, A PSO based approach: Scout particle swarm algorithm for continuous global optimization problems, Journal of Computational Design and Engineering **6**, pp. 129–142, 2019.

[64] Patrick Siarry, Gérard Berthiau, François Durdin, Jacques Haussy, ACM Transactions on Mathematical Software **23**, pp 209–228, 1997.

[65] I.G. Tsoulos, I.E. Lagaris, GenMin: An enhanced genetic algorithm for global optimization, Computer Physics Communications **178,** pp. 843-851, 2008.

[66] M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.

[67] M. Gaviano, D.E. Ksasov, D. Lera, Y.D. Sergeyev, Software for generation of classes of test functions with known local and global minima for global optimization, ACM Trans. Math. Softw. **29**, pp. 469-480, 2003.

[68] J.E. Lennard-Jones, On the Determination of Molecular Fields, Proc. R. Soc. Lond. A **106**, pp. 463–477, 1924.