Co-evolutionary particle swarm optimization to solve constrained optimization problems

Xiaoli Kou\textsuperscript{a}, Sanyang Liu\textsuperscript{a}, Jianke Zhang\textsuperscript{b}, Wei Zheng\textsuperscript{c,}\textsuperscript{*}

\textsuperscript{a} Department of Mathematics Science, Xidian University, Xi’an, 710071, China
\textsuperscript{b} Department of Applied Mathematics and Physics, Xi’an Institute of Posts and Telecommunications, Xi’an, 710121, China
\textsuperscript{c} Department of Computer Science, Xidian University, Xi’an, 710071, China

\textbf{ARTICLE INFO}

\textbf{Keywords:}
Global nonlinear optimization
Particle swarm optimization
Co-evolutionary
Extrapolation
Infeasible degree
Diversity mechanism

\textbf{ABSTRACT}

This paper presents a co-evolutionary particle swarm optimization (CPSO) algorithm to solve global nonlinear optimization problems. A new co-evolutionary PSO (CPSO) is constructed. In the algorithm, a deterministic selection strategy is proposed to ensure the diversity of population. Meanwhile, based on the theory of extrapolation, the induction of evolving direction is enhanced by adding a co-evolutionary strategy, in which the particles make full use of the information each other by using gene-adjusting and adaptive focus-varied tuning operator. Infeasible degree selection mechanism is used to handle the constraints. A new selection criterion is adopted as tournament rules to select individuals. Also, the infeasible solution is properly accepted as the feasible solution based on a defined threshold of the infeasible degree. This diversity mechanism is helpful to guide the search direction towards the feasible region. Our approach was tested on six problems commonly used in the literature. The results obtained are repeatedly closer to the true optimum solution than the other techniques.

\textbf{1. Introduction}

Particle Swarm Optimization (PSO) originally developed by Kennedy and Eberhart in 1995 \cite{1} is inspired with social behavior, such as bird flocking, fish schooling, and has been widely used to solve several types of optimization problems \cite{2, 3}. Nevertheless, their application in constraint optimization is limited \cite{4, 5} because they lack an explicit mechanism to bias the search in constrained search space.

Many search and optimization problems in science and engineering involve a number of constraints which the optimal solution must satisfy. The most common approach adopted to deal with constrained search space is the use of penalty functions \cite{6, 7}. The penalty function approach involves a number of penalty factors which must be set right in any problem to obtain feasible solutions \cite{8, 9}. These approaches is a more robust method, whereas they have several drawbacks from which the main one is they require extensive experimentation for setting up appropriate penalty factors needed to define the penalty function.

Deb \cite{10} developed a constraint handling method based on the penalty function approach which does not require any penalty factor. Powell \cite{11} proposed to map the feasible and infeasible solutions into the two different intervals. Coello \cite{12} proposed different multiobjective-based techniques to solve constraint optimizations problems and so on. In these approaches, the pair-wise comparison used in tournament selection is exploited to make sure that: (i) when two
feasible solutions are compared, the one with better objective function value is chosen, (ii) when one feasible and one infeasible solutions are compared, the feasible solution is chosen, and (iii) when two infeasible solutions are compared, the one with smaller constraint violation is chosen. By all appearances, the second one is not completely reasonable, the constraint boundary handling method has not been taken into account. Many real-world constrained optimization problems have optimum solutions in or near the boundary of the feasible region; it is possible that the infeasible solution with good performance near the optimum can induce global searching to the boundary, so the infeasible solutions near the optimum solutions are very helpful for search processing. However, it is a puzzle how to efficiently use the infeasible solution.

In this paper, we use the infeasible degree selection to handle the constraints. A new selection criterion is adopted as tournament rules to select individuals. Also, a simple diversity mechanism is added, the idea is to define a threshold of the infeasible degree based on the infeasible degree, and the infeasible solution with good performance is probably accepted as the feasible solution. This solution can guide the search toward the boundary of the feasible region. A co-evolutionary PSO (CPSO) is used to search the global optimum, in which a deterministic selection strategy is proposed to ensure the diversity of population; meanwhile, the induction of evolving direction is enhanced by adding a gene-adjusting operator. This technique can effectively improve the cooperation of particles.

This paper is organized as follows. Section 2 describes the standard PSO. In Section 3, we present a detailed description of our approach. Then, experimental results and performance analysis are presented in Section 4. Finally, in Section 5, we offer conclusions and an outlook to future work.

2. Particle swarm optimization

PSO is a heuristic optimization algorithm. It hypothesizes that there are m particles in the D dimensions space, whose position is \( x_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \), and have a fitness function, related with optimization objective function. Every particle moves gradually at a certain speed \( v_i = (v_{i1}, v_{i2}, \ldots, v_{iD}) \) in the D dimensions space. During moving, \( p_i = (p_{i1}, p_{i2}, \ldots, p_{iD}) \) records the best position of the fitness function of the particle \( i \), \( p_g \) records the best position of the fitness function of the whole particle swarm. For the particle \( i \) of iterative \( t \) generation, its position and velocity are updated as follows:

\[
\begin{align*}
    v_{id}(t + 1) &= w v_{id}(t) + c_1 r_1 d(p_{id}(t) - x_{id}(t)) + c_2 r_2 d(p_{gd}(t) - x_{id}(t)) \\
    x_{id}(t + 1) &= x_{id}(t) + v_{id}(t + 1)
\end{align*}
\]  

(2.1)

where \( w \) is an inertial effect coefficient, \( c_1 \) and \( c_2 \) are weight coefficient of the particle individual, \( r_{1d} \) and \( r_{2d} \) are two random functions whose values are between 0 and 1, and the maximum velocity of the particle is limited by \( v_{max} \). The velocity of the particle consists of three parts: the first part represents the degree of momentum of the particles, the second part is the “cognition” part, which represents the independent behavior of the particle itself, the third part is the “social” part, which represents collaboration among the particles. In [1], PSO algorithm’s calculation step is particularly described.

3. Our approach

Our new approach uses the theory of the mathematic extrapolation to enhance the cooperation of the particles, and uses the infeasible degree selection mechanism as the constraint handling method.

3.1. CPSO

The detail features of our CPSO algorithm are the following.

a. Selection strategy

We use a selection strategy which has a random and deterministic property. The population for the initialized generation \( N \) is produced randomly, the population for the other generations are composed of preserving population \( N_1 \) and offspring population \( N_2 \), where \( N = N_1 + N_2 \). Based on the minimum value of the objective function, we arrange the individual of the parent’s population in the order from small to large, we also preserve the best \( N_1 \) individuals to compose the preserving population, and add it to the next population. To the anterior \( N_2 + 1 \) individuals of the parent’s population, we use Co-evolutionary based on the theory of the mathematic extrapolation to produce \( N_2 \) individuals, which compose the offspring population. So, a deterministic selection strategy is proposed based on the idea of “preserving the best, adjusting the part”. This selection strategy has the following advantages:

(1) However, from the good individuals of the parent’s population which are “excellent”, only part of them can be added to the next generation, which holds a greater selection pressure to avoid a stagnation behavior.

(2) The technique of gene-adjusting makes individuals of the population use each other’s information, and induces the evolution direction.

b. Co-evolutionary strategy.

Based on the theory of extrapolation, we proposed a new co-evolutionary strategy, in which the induction of the evolution direction is enhanced by adding a gene-adjusting and adaptive focus-varied tuning operator.

Theorem 1. For the n dimensional multi-variables optimization problem:

Assume that the positions of two particles are \( x_1 = (x_{11}, x_{12}, \ldots, x_{1n}) \) and \( x_2 = (x_{21}, x_{22}, \ldots, x_{2n}) \), and their fitness are \( f(x_1) \) and \( f(x_2) \) respectively. But neither of them is the extreme point. The new individual \( x_3 = (x_{31}, x_{32}, \ldots, x_{3n}) \) is produced by the
Fig. 1. \( x_1 > x_2 \).

Fig. 2. \( x_1 < x_2 \).

Following formulæ:

\[
x_{3i} = x_{1i} + k(x_{1i} - x_{2i}), \quad i = 1, \ldots, n
\]

(3.1)

where \( k \) is the coefficient of modulation, it decides the scope of modulation. \( k \) is set to 0.1 based on experience.

**Proof.** We use the single-variable optimization problem to prove the theory of the extrapolation as follows.

Assume that the positions of two particles are \( x_1 \) and \( x_2 \), and their fitness is \( f(x_1) \) and \( f(x_2) \) respectively. But neither of them is the extreme point. The new individual \( x_3 \) is produced by the following formulate:

\[
x_3 = x_1 + k(x_1 - x_2).
\]

(3.2)

So, we have the following conclusion.

i. If \( x_1 > x_2 \), we can get \( x_3 > x_1 \) based on Eq. (3.2), for a proper small positive \( k \), and we can obtain \( f(x_3) < f(x_1) \), under the above assumption \( f(x_1) < f(x_2) \), as seen in Fig. 1.

ii. If \( x_1 < x_2 \), we can get \( x_3 < x_1 \) based on Eq. (3.2), for a proper small positive \( k \), and we can obtain \( f(x_3) < f(x_1) \), under the above assumption \( f(x_1) < f(x_2) \), as seen in Fig. 2. \( \square \)

For the \( n \) dimensional multi-variables optimization problem, there are so many elements of the individual that we can easily find out that some elements are very near or same. For these elements, it is hopeless using Eq. (3.1) to modulate them. So, based on Eq. (3.1), we introduce a tiny-modulation operator \( h \), and obtain the following co-evolutionary operator equation.

\[
x_{3i} = x_{1i} + k(x_{1i} - x_{2i}) + h, \quad i = 1, \ldots, n
\]

(3.3)

where \( h \) is a random number, and it is very helpful when the value of the \( (x_{1i} - x_{2i}) \) is very small. Eq. (3.3) has the property of adaptive modulation. At the beginning of the evolution process, the individuals are dispersed and far from the optimum, so the scope of modulation is large. At late stages of the evolutionary process, the better individuals in the population gradually become close to each other, so the scope of modulation is small. So we can obtain a better solution quickly.

Compared with the standard PSO algorithm, the CPSO algorithm uses a new selection strategy, and makes full use of the information between the particles by the new co-evolutionary operator equation. This approach can improve the search efficiency and avoid the algorithm becoming trapped in local optima. In the following section, the CPSO algorithm’s calculation step is described in particular.
3.2. Constraint handling method

A constrained optimization problem is usually written as a nonlinear programming (NLP) problem of the following type.

Minimize  \( f(\vec{x}) \)
subject to  \( g_j(\vec{x}) \geq 0, \quad j = 1, \ldots, J \)
\( h_k(\vec{x}) = 0, \quad k = 1, \ldots, K \)
\( x_i^l \leq x_i \leq x_i^u, \quad i = 1, \ldots, n. \)

In the above NLP problem, there are \( n \) variables, \( J \) greater-than-equal-to type inequality constraints, and \( K \) equality constraints. The function \( f \) is the objective function, \( g_j \) is the \( j \)th inequality constraint, and \( h_k \) is the \( k \)th equality constraint. The \( i \)th variable varies in the range \([x_i^l, x_i^u]\).

Our new approach is based on the selection criteria proposed by Deb [10]. In his approach, the following expression is used to assign fitness to a solution:

\[
\text{fit}_i(\vec{x}) = \begin{cases} 
    f_i(\vec{x}), & \text{if feasible} \\
    f_{\text{max}} + \sum_{j=1}^{J} g_j(\vec{x}), & \text{otherwise} 
\end{cases} 
\]  

(3.4)

where the parameter \( f_{\text{max}} \) is the objective function value of the worst feasible solution in the population. Thus, the fitness of an infeasible solution not only depends on the amount of constraint violation, but also on the population of solutions at hand. However, the fitness of a feasible solution is always fixed and is equal to its objective function value.

Motivated by the idea of exploring the capabilities of multi-objective optimization concepts to solve global optimization problems [12], we develop a new diversity mechanism described in the following.

To obtain the constrained conflicting degree of a solution \( x_i \), we first introduce the following definition of infeasible degree (IFD).

\[
\text{IFD}(x_i) = \sum_{j=1}^{J} \left[ \min \{0, g_j(x_i)\} \right]^2 + \sum_{k=1}^{K} [h_k(x_i)]^2 
\]  

(3.5)

where the infeasible degree can be considered as the distance between solution \( x_i \) and the feasible region. When solution \( x_i \) is a feasible solution, the infeasible degree value is zero. When solution \( x_i \) is an infeasible solution, the infeasible degree value is inversely proportional to the distance between solution \( x_i \) and the feasible region.

To increase the selection pressure with the evolutionary process, the threshold of infeasible degree is defined as the product of a linearly decreasing gene and the average infeasible degree value for population (IFD_p).

\[
\text{IFD}_p = \frac{\sum_{i=1}^{M} \text{IFD}(x_i)}{T(t)M} 
\]  

(3.6)

where \( T(t) = 0.8 + (t/\text{Max-number}) \times 2.2 \). Max-number is maximum number of generations. So, the value \( T(t) \) linearly increases from 0.8 to 3 with evolution. \( M \) is the number of particles.

In the every generation, one solution is accepted or not based on the comparison between the infeasible degree and the threshold of the infeasible degree. The acceptance rule is:

- When the solution \( x_i \) is feasible solution, \( x_i \) is accepted.
- When the solution \( x_i \) is infeasible solution, if \( \text{IFD}(x_i) \leq \text{IFD}_p \), \( x_i \) is accepted; if \( \text{IFD}(x_i) > \text{IFD}_p \), \( x_i \) is rejected.
- To preserve the number of particles unchanged, stochastically produce some particles to substitute for the rejected particles.

Based on the above analysis, we proposed the following expression used to assign fitness to a solution.

\[
\text{fit}_i(\vec{x}) = \begin{cases} 
    f_i(\vec{x}), & \text{if accepted} \\
    f_{\text{max}} + \sum_{j=1}^{J} g_j(\vec{x}), & \text{otherwise} 
\end{cases} 
\]  

(3.7)

where the parameter \( f_{\text{max}} \) is the objective function value of the worst feasible solution in the population.

This technique can help the particles reach the feasible region of the search space, and to properly keep the infeasible solution attracting the particles to the constraint boundary. The CPSO with the constraints handling mechanism algorithm is described in the following.

(i) Initialization.
(ii) For each particle, evaluate its infeasible degree \( \text{IFD}(x_i) \).
(ii) Evaluate the threshold of infeasible degree \( IFD_p \).
(iv) For each particle, evaluate its fitness value based on the Eq. (3.7) and arrange them in the order from small to large.
(v) Produce a new particle population, in which the good \( N_1 \) individuals are preserved, and the \( N_2 \) individuals are produced by the above co-evolutionary equation operator (3.3).
(vi) Evaluate the new particle population. If the evolutionary results are unsatisfactory, then decrease the tiny-modulation operator \( h \).
(vii) For each particle in the new population, evaluate its fitness value.
(viii) Compare each particle’s fitness value with the current particle’s \( p_{\text{best}} \). If current value is better than \( p_{\text{best}} \), set its \( p_{\text{best}} \) value to the current value and the \( p_{\text{best}} \) location to the current location in \( n \)-dimensional space.
(ix) Compare the fitness value with the population’s overall previous best. If current value is better than \( p_{\text{best}} \), then reset \( p_{\text{best}} \) to the current particle’s array index and value.
(x) Change the velocity and position of the particle according Eq. (2.1).
(xi) Loop to step (2) until a stopping criterion is met.

Summarizing, our CPSO algorithm can make full use of the information between the particles to induce the evolutionary direction. At the same time, we proposed the infeasible degree selection with the threshold as the constraint handling method. It allow infeasible particles to search for the global optimum. The technique can provide a search direction towards the feasible region and avoid that the algorithm defines the penalty function and straight way removes the infeasible ones with good performance.

4. Experiments and results

To evaluate the performance of the proposed approach, we used the six test functions [14] as follows.

(1) \( g_01 \)

\[
\begin{align*}
\text{Max } f(\vec{x}) &= \left| \sum_{i=1}^{n} \cos^4(x_i) - 2 \prod_{i=1}^{n} \cos^2(x_i) \right| \\
\text{s.t. } g_1(\vec{x}) &= 0.75 - \prod_{i=1}^{n} x_i \leq 0 \\
g_2(\vec{x}) &= \sum_{i=1}^{n} x_i - 0.75n \leq 0
\end{align*}
\]

where \( n = 20 \) and \( 0 \leq x_i \leq 10 \) (\( i = 1, \ldots, n \)). The global maximum is unknown, the best reported solution is \( f(x^*) = 0.803619 \) [13]. Constraint \( g_1 \) is close to being active (\( g_1 = 10^{-8} \)).

(2) \( g_02 \)

\[
\begin{align*}
\text{Min } f(\vec{x}) &= 5 \sum_{i=1}^{4} x_i - 5 \sum_{i=1}^{4} x_i^2 - 13 x_i \\
\text{s.t. } g_1(\vec{x}) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \\
g_2(\vec{x}) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \\
g_3(\vec{x}) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \\
g_4(\vec{x}) &= -8x_1 + x_{10} \leq 0 \\
g_5(\vec{x}) &= -8x_2 + x_{11} \leq 0 \\
g_6(\vec{x}) &= -8x_3 + x_{12} \leq 0 \\
g_7(\vec{x}) &= -2x_4 - x_5 + x_{10} \leq 0 \\
g_8(\vec{x}) &= -2x_6 - x_7 + x_{11} \leq 0 \\
g_9(\vec{x}) &= -2x_8 - x_9 + x_{12} \leq 0
\end{align*}
\]

where the bounds are \( 0 \leq x_i \leq 1 \) (\( i = 1, \ldots, 9 \)), \( 0 \leq x_i \leq 100 \) (\( i = 10, 11, 12 \)) and \( 0 \leq x_{13} \leq 1 \). The global optimum is at \( x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1) \), where \( f(x^*) = -15 \). Constraint \( g_1, g_2, g_3, g_4, g_5, \) and \( g_6 \) are active.

(3) \( g_03 \)

\[
\begin{align*}
\text{Max } f(\vec{x}) &= (\sqrt{n})^n \prod_{i=1}^{n} x_i \\
\text{s.t. } h(\vec{x}) &= \sum_{i=1}^{n} x_i^2 - 1 = 0
\end{align*}
\]
where $n = 10$ and $0 \leq x_i \leq 1 (i = 1, \ldots, n)$. The global optimum is at $x_i^* = 1/\sqrt{n}$ $(i = 1, \ldots, n)$, where $f(x^*) = 1$.

(4) g04

\[
\begin{align*}
\text{Min} & \quad f(\vec{x}) = (x_1 - 10)^2 + (x_2 - 20)^2 \\
\text{s.t} & \quad g_1(\vec{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \\
& \quad g_2(\vec{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0
\end{align*}
\]

where $13 \leq x_1 \leq 100, 0 \leq x_2 \leq 100$. The global optimum is at $x^* = (14.095, 0.84296)$, where $f(x^*) = -6961.81388$. Both constraints are active.

(5) g05

\[
\begin{align*}
\text{Max} & \quad f(\vec{x}) = [\sin(3(2\pi x_1)) \sin(2\pi x_2)][x_2^3(x_1 + x_2)] \\
\text{s.t} & \quad g_1(\vec{x}) = x_1^2 - x_2 + 1 \leq 0 \\
& \quad g_2(\vec{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0
\end{align*}
\]

where $0 \leq x_1 \leq 10$ and $0 \leq x_2 \leq 10$. The global optimum is at $x^* = (1.2279713, 4.2453733)$, where $f(x^*) = 0.095825$.

(6) g06

\[
\begin{align*}
\text{Min} & \quad f(\vec{x}) = x_1 + x_2 + x_3 \\
\text{s.t} & \quad g_1(\vec{x}) = -1 + 0.00245(x_1 + x_2) \leq 0 \\
& \quad g_2(\vec{x}) = -1 + 0.0025(x_3 + x_4 - x_4) \leq 0 \\
& \quad g_3(\vec{x}) = -1 + 0.01(x_5 - x_5) \leq 0 \\
& \quad g_4(\vec{x}) = -0.3333333325x_4 + 100x_1 - 83333.3333 \leq 0 \\
& \quad g_5(\vec{x}) = -x_3x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0 \\
& \quad g_6(\vec{x}) = -x_3x_8 + 12500000 + x_8x_5 - 2500x_5 \leq 0
\end{align*}
\]

where the bounds are $100 \leq x_1 \leq 10000, 1000 \leq x_i \leq 10000 (i = 2, 3)$ and $10 \leq x_i \leq 1000 (i = 4, \ldots, 8)$. The global optimum is at $x^* = (579.3167, 1359.943, 5110.071, 180.0174, 295.5985, 217.9799, 286.4162, 395.5979)$, where $f(x^*) = 7049.3307$. Constraint $g_1, g_2$ and $g_3$ are active.

4.1. The design of parameters

For the experiments, we used the following parameters: number of particles $M = 500; c_1 = c_2 = 1.8, k = 0.1, h = 0.00001$; the inertia weights $w$ linearly decreases from 1.0 to 0.4 with evolution. For equality constraints, we convert them into inequality constraints by the tolerance value used in [10] as follows:

\[
g_{k+j}(\vec{x}) = \delta - |h_k(\vec{x})| \geq 0
\]

where $\delta$ defined as the tolerance value is a small positive value. In this paper, the tolerance value $\delta$ is set to 0.00001.

The algorithm terminates as soon as a certain convergence criterion is reached. We use two kinds of criterion. If the function value of the optimum parameters $f_{opt}$ is known, then the criterion is based on the difference between the function value of the current parameters and the optimum parameters $f_{opt}$. In each iteration step, this difference $|f - f_{opt}|$ is compared with a previously set precision value. As soon as the difference becomes smaller than the target precision value, the algorithm terminates. If the function value of the optimum parameters $f_{opt}$ is unknown, then the criterion is based on a previously set a maximum number of iterations $N_{max}$. As soon as the iteration step reach the value $N_{max}$, the algorithm terminates.

We performed 30 independent runs for each test problem, the results obtained the “Best”, “Worst” and “Mean” solution by our approach are presented in Table 1. In addition, we showed the number of runs (out of 30) in which the global optimum (or best known solution) was found in Table 1, and compared our approach against three approaches: the SR [14], the SMES [15], and the HM [16]. The compared results including the “Best” solution and the “Worst” solution are summarized in Tables 2 and 3.
Table 2
Comparison of the Best results by the proposed algorithm against the SR, SMES and HM algorithms.

<table>
<thead>
<tr>
<th>TF</th>
<th>IEPSO</th>
<th>SR</th>
<th>SMES</th>
<th>HM</th>
</tr>
</thead>
<tbody>
<tr>
<td>g01</td>
<td>0.803619</td>
<td>0.803619</td>
<td>0.863601</td>
<td>0.79953</td>
</tr>
<tr>
<td>g02</td>
<td>−15.000</td>
<td>−15.000</td>
<td>−15.000</td>
<td>−14.7886</td>
</tr>
<tr>
<td>g03</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.9997</td>
</tr>
<tr>
<td>g04</td>
<td>−6961.814</td>
<td>−6961.814</td>
<td>−6961.814</td>
<td>−6952.1</td>
</tr>
<tr>
<td>g05</td>
<td>0.095825</td>
<td>0.095825</td>
<td>0.095825</td>
<td>0.095825</td>
</tr>
<tr>
<td>g06</td>
<td>7092.8145</td>
<td>7054.316</td>
<td>7051.903</td>
<td>7147.9</td>
</tr>
</tbody>
</table>

Table 3
Comparison of the Worst results by the proposed algorithm against the SR, SMES and HM algorithms.

<table>
<thead>
<tr>
<th>TF</th>
<th>IEPSO</th>
<th>SR</th>
<th>SMES</th>
<th>HM</th>
</tr>
</thead>
<tbody>
<tr>
<td>g01</td>
<td>0.802518</td>
<td>0.781975</td>
<td>0.7852238</td>
<td>0.79671</td>
</tr>
<tr>
<td>g02</td>
<td>−15.000</td>
<td>−15.000</td>
<td>−15.000</td>
<td>−14.7082</td>
</tr>
<tr>
<td>g03</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.9989</td>
</tr>
<tr>
<td>g04</td>
<td>−6961.712</td>
<td>−6875.940</td>
<td>−6961.284</td>
<td>−6342.6</td>
</tr>
<tr>
<td>g05</td>
<td>0.095825</td>
<td>0.095825</td>
<td>0.095825</td>
<td>0.0891568</td>
</tr>
<tr>
<td>g06</td>
<td>7200.1603</td>
<td>7559.192</td>
<td>7253.047</td>
<td>8163.6</td>
</tr>
</tbody>
</table>

Table 4
Comparison results by Experiment 1 for g01 and g02.

<table>
<thead>
<tr>
<th>Solution</th>
<th>CPSO/g01</th>
<th>PSO/g01</th>
<th>CPSO/g02</th>
<th>PSO/g02</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>0.803619</td>
<td>0.803619</td>
<td>−15.000</td>
<td>−15.000</td>
</tr>
<tr>
<td>Worst</td>
<td>0.747290</td>
<td>0.779328</td>
<td>−15.000</td>
<td>−15.000</td>
</tr>
<tr>
<td>Mean</td>
<td>0.802518</td>
<td>0.799362</td>
<td>−15.000</td>
<td>−15.000</td>
</tr>
<tr>
<td>St. Dev</td>
<td>0.005</td>
<td>0.041</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AG</td>
<td>802.539030</td>
<td>958.340000</td>
<td>534.336735</td>
<td>629.029441</td>
</tr>
</tbody>
</table>

4.2. Discussion of results

As described in Table 1, our approach was able to find the global optimum in five test functions (g01, g02, g03, g04, g05), and it found solutions very close to the global optimum in the preserving function (g06). Additionally, the St. Dev value of the entire test functions except for g05 is very small. From the process of our experiment, we observe that the stability performance of our algorithm is good. Furthermore, in 30 independent runs, our approach can find the optimum (or best known solution) in seven test functions (g01, g02, g03, g04, g05, g06) every time. Compared against SR, our approach find the same better “Best” solution in functions g01, g02, g03, g04, g05, and find the better “Mean” solution in function g06. Compared against SMES, our approach is able to find the better “Best” solution in functions g01 and the same better “Best” solution in functions g02, g03, g04 and g05, and find a similar better “Mean” solution in functions g04, and a better “Mean” solution in function g06. Compared against HM, our approach is able to find a better “Best” and “Mean” solution in functions g01, g02, g03, g04 and g05. As we can see, our approach can deal with different types of constraint optimization problems, especially the problems where such an optimum lies on the boundaries of the feasible region (g01, g02, g03, g04, g05), it can precisely and highly efficiently find the optimum solutions. These results show that our approach is efficient and has a competitive performance with respect to other commonly used optimization algorithms.

4.3. Performance analysis

After discussing the experimental results, we want to extensively analyze the impact of the strategies of our CPSO on its performance. So we performed another three experiments using the six test functions previously described.

(1) Experiment 1: We compared the CPSO with the co-evolutionary strategy versus the PSO without the co-evolutionary strategy using the original set of parameters.

We performed 30 independent runs for each test problem, the results obtained the “Best”, the “Worst” solution, the “Mean” solution, the St. Dev and the Average generation (AG) by the CPSO and the PSO are presented in Tables 4–6.

From the results shown in Tables 4–6, it is clear that the version with the CPSO provided better “Best” results, as well as the “Worst” and the “Mean” results for the most of the test functions. Also, it found better “Average generation (AG)” results for the all test functions. All these results suggest that the co-evolutionary strategy produces an improvement in the convergence speed.

(2) Experiment 2: We compared the CPSO with an infeasible degree selection mechanism (CPSO) versus the CPSO without infeasible degree selection mechanism using the original set of parameters.

We performed 30 independent runs for each test problem, the results obtained the “Best”, the “Worst” solution, the “Mean” solution by the CIPSO and the CPSO are presented in Tables 7–9.
From the results shown in Tables 7–9, it is clear that the version with the CIPSO provided better “Best” results, as well as the “Worst” and the “Mean” results for the all test functions. The version with the CPSO was unable to reach the feasible region in g02, g04 and g06 where their optimums lie on the boundaries of the feasible region. These results suggest that the infeasible degree selection mechanism help to sample feasible region as to find competitive results.

(3) Experiment 3: We modified the number of particles of the iterations. We perform runs using 200 and 400 particles. We performed 30 independent runs for each test problem, the results obtained the “Best”, “Worst”, “Mean” solution and the St. Dev by CIPSO are presented in Tables 10–12.

As indicated before, in this experiment, we varied the number of particles of the iterations. We want to see the effect of this parameter in the performance of our CIPSO. From the results shown in Tables 10–12, we can see that the “Best” results, as well as the “Worst” and the “Mean” results with 200 particles were unable to be obtained for all the test functions, and the “Best” results with 400 particles were obtained for the g02 and g05. But, it found the “St. Dev” results for all the test functions were not good. All these results suggest that the value of 400 and 500 for the number of the particles provided better results in most problems.
Table 11
Comparison results by Experiment 3 for g03 and g04.

<table>
<thead>
<tr>
<th>Solution</th>
<th>200/g03</th>
<th>400/g03</th>
<th>200/g04</th>
<th>400/g04</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>0.482</td>
<td>1.000</td>
<td>−4235.817</td>
<td>−6902.719</td>
</tr>
<tr>
<td>Worst</td>
<td>0.302</td>
<td>0.827</td>
<td>−3900.576</td>
<td>−6104.531</td>
</tr>
<tr>
<td>Mean</td>
<td>0.412</td>
<td>0.736</td>
<td>−4086.723</td>
<td>−6562.828</td>
</tr>
<tr>
<td>St. Dev</td>
<td>17.59</td>
<td>4.45</td>
<td>191.256</td>
<td>27.39</td>
</tr>
</tbody>
</table>

Table 12
Comparison results by Experiment 3 for g05 and g06.

<table>
<thead>
<tr>
<th>Solution</th>
<th>200/g05</th>
<th>400/g05</th>
<th>200/g06</th>
<th>400/g06</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>0.095100</td>
<td>0.095825</td>
<td>22678.924</td>
<td>7319.405</td>
</tr>
<tr>
<td>Worst</td>
<td>0.093721</td>
<td>0.095825</td>
<td>29043.003</td>
<td>7811.630</td>
</tr>
<tr>
<td>Mean</td>
<td>0.094820</td>
<td>0.095825</td>
<td>25704.481</td>
<td>7560.92</td>
</tr>
<tr>
<td>St. Dev</td>
<td>4.62</td>
<td>0</td>
<td>160.42</td>
<td>21.73</td>
</tr>
</tbody>
</table>

5. Conclusion

The purpose of this paper was to present a new approach to handle constraints in Particle Swarm Optimization. We developed a PSO with a co-evolutionary strategy based on the theory of extrapolation. Furthermore, the infeasible degree selection mechanism guides the search toward the global optimum. Additionally, a simple diversity mechanism was added; it properly accepted the infeasible solutions with good performance. Finally, the simulation results on the test functions demonstrate that our approach is competitive and easy to implement. In the future, the convergent proof of the proposed algorithm should be studied. It would be interesting to implement other constraint handling mechanism using a PSO algorithm. This aims to explore an explicit mechanism to bias the search using a PSO algorithm in constrained search space.

Acknowledgements

The authors would like to thank the anonymous referees for their constructive suggestions, which led to the improvement of the paper.

References