A Multi-Objective Constraint-Handling Method with PSO Algorithm for Constrained Engineering Optimization Problems

Lily D Li, Xiaodong Li, Member, IEEE, and Xinghuo Yu, Fellow, IEEE

Abstract — This paper presents a multi-objective constraint handling method incorporating the Particle Swarm Optimization (PSO) algorithm. The proposed approach adopts a concept of Pareto dominance from multi-objective optimization, and uses a few selection rules to determine particles' behaviors to guide the search direction. A goal-oriented programming concept is adopted to improve efficiency. Diversity is maintained by perturbing particles with a small probability. The simulation results on the three engineering benchmark problems demonstrate the proposed approach is highly competitive.

I. INTRODUCTION

Over the last decade or two, evolutionary algorithms have been extensively studied as search and optimization tools in various problems domains. The primary reasons for their success are their broad applicability, ease of use and global perspective [1]. Although evolutionary algorithms have been successful in many applications, their uses in solving constrained optimization problems remain problematic because their original versions lack a mechanism to incorporate constraints into the fitness function [2-5]. There has been little work on handling constraints by the Particle Swarm Optimization (PSO) algorithm [5]. PSO is a relative new stochastic method for optimizing hard numerical functions based on metaphor of social behaviour of flocks of birds and schools of fish [6]. The PSO technique has proven to be effective and efficient for solving real-valued global unconstrained optimization problems [7, 8]. For constrained optimization problems, there have been only a few attempts. Hu and Eberhart [9] proposed to use a preserving feasibility strategy to handle constraints with PSO. The drawback of this model is that the initialization process may be impractically long or almost impossible for those CNOPs (Constrained Nonlinear Optimization Problems) that have extremely small feasible spaces [10]. Parsopoulos and Vrahatis [11] adopted the penalty functions in a PSO. However, in this model the penalty factors need to be carefully fine-tuned [4] and they are problem-dependent [12, 13]. Pulido and Coello introduced a selection rule based on feasibility checking and constraint violation measurement to handle constraints with PSO [14]. However, it appears that it is not convenient to measure the constraint violation units, and the performance is not consistent. Wei and Wang [15] integrated the multi-objective constraint handling mechanism with PSO. In [15], a selection strategy similar to [14] is used for constraint handling. The approach is only tested by four lower-constrained optimization problems with promising results.

Multi-objective constraint-handling was firstly proposed by Fonseca and Fleming back in 1995 (as cited in [3]). The main idea is to treat the constraints as extra objectives. By doing this, an original single objective constrained optimization problem can be transformed into a multi-objective unconstrained optimization problem. Then the techniques for multi-objective optimization can be employed. Since then, a number of models have been developed using this idea. These models can be classified into two groups - The first group transfers the original problem into an unconstrained bi-objective optimization problem (the original objective function and the sum of constraint violations); and the second group transfers the original problem into an unconstrained multi-objective optimization problem (constraints as separate objectives) [2, 3]. Several representative examples include COMO [16], Camponogara & Talukdar [17], Mezura-Montes & Coello [18-20] and Jimenez et al. [21]. Unfortunately, these models have some shortcomings. For example, some of them add extra computational cost [17, 20]; some of them require extra parameters [16, 18, 21]. A detailed review can be found in [3]. It is noticed that most of these models are built on genetic algorithms.

In this research, we propose a new method to integrate the multi-objective constraint handling mechanism with the PSO. By converting a single objective constrained optimization problem into a bi-objective unconstrained optimization problem, the proposed approach aims to minimize the original objective function and the total amount of constraint violations (the second objective). The concept of Pareto domination from multi-objective optimization is adopted in determining a particle's best past experience and the best social experience in the group. The second objective is used as a benchmark to select particles (defined in selection rules). A goal-oriented concept is adopted to improve efficiency. Diversity is maintained by a small perturbation.

The rest of the paper is organized as follows: Section II presents the problem definition, formulation and transformation; Section III describes the proposed multi-objective constraint handling incorporating with the PSO.
algorithm. It includes a brief description to the PSO algorithm, selection rules and diversity control scheme. Section IV presents the simulation results for the three well-known engineering benchmark functions. Section V concludes the paper and indicates some future work.

II. PROBLEM DEFINITION, TRANSFORMATION AND FORMULATION

A general single objective constrained optimization problem can be stated as to:

\[
\begin{align*}
\text{Find } & \mathbf{X} \text{ which optimizes } f(\mathbf{X}) \\
\text{subject to } & \quad g_i(\mathbf{X}) \leq 0, \quad i = 1, 2, \ldots, m; \\
& \quad h_j(\mathbf{X}) = 0, \quad j = 1, 2, \ldots, p;
\end{align*}
\]

where \( \mathbf{X} = [x_1, x_2, \ldots, x_n]^T \), with each \( x_i \) (\( i = 1, \ldots, n \)) is bounded by lower and upper limits \( L_i \leq x_i \leq U_i \); \( m \) is the number of inequality constraints and \( p \) is the number of equality constraints. In both cases, constraints can be linear or nonlinear.

Note that a maximization problem can be converted into a minimization problem by setting \( \max(\{x\}) = -\min(-\{x\}) \), and equality constraints \( h_j(\mathbf{X}) = 0 \) can be translated into inequality constraints \( |h_j(\mathbf{X})| \leq \delta, \) where \( \delta \) is a allowed tolerance. The equation (1) then can be transformed into:

\[
\begin{align*}
\text{minimize } & \quad f(\mathbf{X}) \\
\text{subject to } & \quad g_i(\mathbf{X}) \leq 0, \quad i = 1, 2, \ldots, m.
\end{align*}
\]

where \( m \) is the total number of constraints.

Camponogara & Talukar [17] proposed an approach in which a global optimization problem was transformed into a bi-objective problem where the first objective is to minimize the original objective function and the second is to minimize the total amount of constraint violations:

\[
\Phi(\mathbf{X}) = \sum_{i=1}^{m} \max(0, g_i(\mathbf{X}))
\]

where \( \Phi(\mathbf{X}) \) is a total amount of constraint violations. From the equation (3), if a solution vector \( \mathbf{X} \) satisfies all constraints, i.e., \( g_i(\mathbf{X}) \leq 0 \) (\( i = 1, \ldots, m \)), \( \Phi(\mathbf{X}) \) returns a zero, otherwise, it return a positive number (total amount of constraint violations). Thus, the optimum value for \( \Phi(\mathbf{X}) \) is zero.

Therefore, the equation (2) can be transformed into the following:

\[
\begin{align*}
\text{minimize } & \quad f(X) - \Phi(\mathbf{X}) \\
\text{where } & \quad \Phi(\mathbf{X}) = \sum_{i=1}^{m} \max(0, g_i(\mathbf{X}))
\end{align*}
\]

Now, the original single objective constrained optimization problem is transformed into a bi-objective unconstrained optimization problem.

For a general multi-objective optimization problem, the ideal procedure is to find a set of Pareto-optimal solutions first and then choose one solution from the set by using some other higher level information [12]. For global constrained optimization, constraint satisfaction is a must and it is more important than real objective function minimization. That is, if a solution is not feasible, no matter how fit its objective function is, it is of little use. In other words, if a solution is feasible, even if it is not fit enough, it can be still considered as a candidate solution. Therefore, the second objective \( \Phi = 0 \) (totally constraint satisfied) or \( \Phi \leq \varepsilon \) (total constraint nearly satisfied), can be used as higher-level information to guide decision making during the search. The \( \varepsilon \) is a small positive number which indicates the feasibility tolerance. This is the so called “decision making during the search” approach in multi-objective optimization [22]. Fig. 1 below is an example to illustrate the Pareto-front, feasible solutions and the desired solution to the established bi-objective optimization problem as described by equation (4), the final solution will fall between A to B depending on how \( \varepsilon \) is selected.

Another concept adopted in the proposed algorithm is the so called goal oriented programming. The main idea in goal programming is to find solutions which attain a predefined target for one or more objective functions [12]. If there is no solution that has achieved pre-specified targets in all objective functions, the task is to find solutions which minimize deviations from the targets[12]. Suppose \( t \) is a predefined target for \( f(\mathbf{X}) \), the optimization problem in equation (4) can be transformed into:

\[
\begin{align*}
\text{minimize } & \quad f(\mathbf{X}) - t, \Phi(\mathbf{X}) \\
\text{goal } & \quad \Phi(\mathbf{X}) \leq \varepsilon \text{ and } f(\mathbf{X}) - t \leq \Delta
\end{align*}
\]

where \( \Delta \) and \( \varepsilon \) are two small positive numbers which indicate how close a solution to the predefined target. Please note there is no need to use \( |f(\mathbf{X}) - t| \leq \Delta \) because any better-than-target solutions are allowed to be found.
Fig. 2. The solutions area for a bi-objective constraint handling optimization problem.

Fig. 2 is an example to illustrate the solution area (goal area) for equation (5). By adding the goal to the optimization process, all particles should fly toward the goal area. Once the goals are achieved, no more evolution is needed. In case there is no solution satisfying both goals, the priority is given to $\Phi(X) \leq \varepsilon$, the solution is found from all particles where their $\Phi(X) \leq \varepsilon$.

Most multi-objective optimization methods use a domination concept to search for non-dominated solution, since this concept allows a way to compare solutions with multiple objectives. The definition for domination as stated below.

Definition: A solution $X^{(j)}$ is said to dominate the other solution $X^{(j)}$, if both conditions 1 and 2 are true:

1. The solution $X^{(j)}$ is no worse than $X^{(j)}$ in all objectives, (for all $j = 1, 2, \ldots, m$).
2. The solution $X^{(j)}$ is strictly better than $X^{(j)}$ in at least one objective, for at least one $j \in \{1, 2, \ldots, m\}$

III. THE PROPOSED APPROACH

A. The PSO Algorithm in Brief

A particle swarm optimization algorithm consists of individuals, called particles, that form a swarm. Each particle represents a candidate solution to the problem. Particles change their positions by flying in a multi-dimensional search space looking for the optimal position. During flight, each particle adjusts its position according to its own experience and the experience of its neighboring particles, making use of the best position encountered by itself and the best position in the entire population (or its local neighborhood). The performance of each particle is measured by a predefined fitness function (objective function) which is problem-dependent.

Let $i$-th particle in a D-dimensional search space be represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$. The best previous position of the $i$-th particle in the flight history is $p_{Best_i} = (p_{i1}, p_{i2}, \ldots, p_{id})$. The position of the best particle of the neighborhood is $p_{Best} = (p_{k1}, p_{k2}, \ldots, p_{kd})$.

The velocity for particle $i$ is $V_i = (v_{i1}, v_{i2}, \ldots, v_{id})$. In the PSO algorithm, the next position $(t+1)$ of particle $i$ on the dimension $d$ is manipulated by the following equations ($r$ denote the iteration):

$$v^{(t+1)}_{id} = \chi \left[ w \cdot v^{(t)}_{id} + c_1 \cdot r_{id} \cdot (p_{Best}^{(t)} - x^{(t)}_{id}) ight]$$

$$v^{(t+1)}_{id} = c_2 \cdot r_{id} \cdot (x^{(t)}_{id} - x^{(t)}_{Best_id})$$

$$x^{(t+1)}_{id} = x^{(t)}_{id} + v^{(t+1)}_{id}$$

where $d = 1, 2, \ldots, D$, $D$ is the search dimension; $i = 1, 2, \ldots, N$, and $N$ is the number of particles in the swarm; $w$ is the weight; $c_1$ and $c_2$ are two positive constants, called the cognitive and social parameters respectively (or acceleration constants); $r_{sid}$ and $r_{sid}$ are two random numbers within the range [0, 1].

Although Clerc and Kennedy [23] suggested use of a constriction coefficient $\chi$ to the velocity formula and shows that the constriction coefficient can converge without using $V_{max}$. In order to ensure convergence and explore a wider area, in this research, both $\chi$ and $V_{max}$ will be used.

As mentioned above, the original PSO algorithm and its variations have no mechanism to incorporate constraints handling into the algorithm. In order to integrate constraints handling with PSO, we introduce a few selection rules to determine particles’ behavior in the next section. A diversity control scheme is also introduced in the next section.

B. Selection Rules and Diversity Control

In PSO algorithm, a particle’s best past experience and its group’s best experience play a key role in guiding its search direction. For a multi-objective optimization problem, due to many objectives involved, the notion of dominance comparison is adopted [12]. The following selection rules are defined:

- Non-dominated particles are better than dominated ones.
- A particle with lower $\Phi$ (constraint violations) is better than a particle with higher $\Phi$.

A perturbation with a minor probability of $p$ is also introduced to maintain diversity and prevent premature convergence. It can be empirically derived.

C. The Proposed Algorithm

Fig. 3 illustrates the proposed algorithm. Comparing with the original PSO, the algorithm has the following features:

- Whenever calculating fitness, both objectives $F = f(X) - t$ and $\Phi$ need to be evaluated;
- If a particle’s new location is better than its best past location, the $p_{Best}$ is updated (decided by selection rules);
- A particle’s best neighboring particle is determined by the two steps:
  a) Find all the non-dominated particles in the neighborhood;
  b) If there is only one non-dominated particle in the neighborhood, select it as $p_{Best}$; otherwise
select one with the lowest $\Phi$ as $\mathbf{b}_{\text{Best}}$ (the lower $\Phi$ means closer to the feasible region).

- A minor perturbation with the probability of $p$ is introduced after calculating the next particle position.
- If perturbed particle is better than the particle before perturbation, replace the particle with the perturbed one.
- After each iteration, check whether the goal is obtained; if obtained, the iteration ends.

![Pseudo code of the proposed multi-objective constraint-handling method with PSO algorithm](image)

### IV. ENGINEERING OPTIMIZATION EXAMPLES

This section reports the experiment results to the three well-known engineering optimization problems - the welded beam design problem (E01), the pressure vessel design problem (E02) and the spring design problem (E03). The proposed approach has been implemented in multithreaded Java programming language. The PSO neighborhood topology is set to ring topology with the neighbor size of 2. For example, if the neighbor size is 2, a particle with index $i$ will have the particle index $i-1$ and $i+1$ as its neighbors. For each case, 30 independent runs (implemented in 30 threads) have been performed. The PSO parameters are: $w = 0$ (empirically derived); $c_1 = c_2 = 2$; $\gamma = 0.63$; $V_{\text{max}} = 0.5 \ast$ (decision variable range); $p = 0.1\%$; number of particles is 100; the maximum iteration is set to 10,000. The feasibility tolerance allowed $\varepsilon = 1.0 \ast 10^{-9}$, that is, if a solution's total amount of constraint violation $\Phi \leq \varepsilon$, the solution is feasible.

In order to see the improvement made from different approaches, results from other three most recent approaches are included in this section. The three approaches, CPSO [24], Hu et al. [25] (denoted as HES-PSO) and Coello & Montes [26] have been selected for comparison for the following reasons. Both CPSO and HES-PSO are based on the PSO algorithm as we adopted, however, they use different constraint handling methods. CPSO adopted a penalty function method and HES-PSO adopted a preserving feasibility method. The Coello & Montes' approach uses the similar multi-objective constraint handling method as we adopted, but their implementation is through genetic algorithm. Therefore, we can evaluate the algorithm from different perspectives.

A. E01: Welded beam design problem

A welded beam is designed for minimum cost subject to constraints of shear stress ($\tau$), bending stress in the beam ($\sigma$), buckling load on the bar ($P_c$), end deflection of the beam ($\delta$), and side constraints. There are four design variables: the thickness of the weld $h - x_1$, the length of the welded joint $l - x_2$, the width of the beam $t - x_3$, and the thickness of the beam $b - x_4$.

Please note the welded beam problem included in this paper is not exactly same as the original version proposed by Reklaitis et al in 1983 [27]. There are five constraints in the original version. For some reason, many researchers have studied another version of this problem as following with seven constraints. We include this version for comparison purpose.

The formal statement of the problem is the following:

Minimize

$$f(X) = 1.10471x_1^3x_2 + 0.04811x_3x_4(14.0 + x_1)$$

Subject to

$$g_1(X) = \tau(X) - \tau_{\text{max}} \leq 0$$
$$g_2(X) = \sigma(X) - \sigma_{\text{max}} \leq 0$$
$$g_3(X) = x_1 - x_4 \leq 0$$
$$g_4(X) = 0.10471x_1^3 + 0.04811x_3x_4(14 + x_1) - 5 \leq 0$$
$$g_5(X) = 0.125 - x_3 \leq 0$$
$$g_6(X) = \delta(X) - \delta_{\text{max}} \leq 0$$
$$g_7(X) = P - P_c(X) \leq 0$$

Where:

$$\tau(X) = \sqrt{(\tau^\prime)^2 + 2r \tau^\prime \tau^\prime \frac{x_2}{2R}}$$
$$\tau^\prime = \frac{P}{2 \sqrt{x_1 x_2}}$$
$$\tau^\prime = \frac{MR}{J}$$
$$R = \frac{x_1^2 + (x_1 + x_2)^2}{2}$$
$$J = 2 \left( \frac{x_1^2}{12} + (x_1 + x_2)^2 \right)$$

$$2008 IEEE Congress on Evolutionary Computation (CEC 2008)$$

1531
\[
\delta(x) = \frac{4P L^3}{EI_x x^2}, \quad \sigma(x) = \frac{6P L}{x x^3} \frac{1}{x}
\]

\[
P(x) = \frac{4.013 E \left( x^2 x^3 y^2 \right)}{36} \sqrt{\frac{E}{2L}} \frac{1}{x} \frac{E}{4G}
\]

The ranges for the design variables are given as follows:

\[
0.1 \leq x_t \leq 2.0, \quad 0.1 \leq x_s \leq 10, \quad 0.1 \leq x_s \leq 0.1, \quad 0.1 \leq x_t \leq 2.0.
\]

The best solutions found from our simulations and the three approaches discussed above are listed in Table I. In our approach, the best result is 1.724852321, which is close to the best-known result 1.724852; the mean result for 30 independent runs is 1.72409194, and the standard deviation is 0.054262394. According to the Table I, the solution found from our approach is better than those achieved by the other three approaches (the result presented in CPSO [24] seems incorrect because they used \( \tau_{max} = 13,000 \) psi rather than \( \tau_{max} = 13,600 \) psi; the results presented in HFS-PSO [25] have some constraints violated). The statistical data demonstrate our approach performs well in both search quality and consistency.

### Table I

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_t (J) )</td>
<td>0.205720642</td>
<td>0.202369</td>
<td>0.20573</td>
<td>0.205986</td>
</tr>
<tr>
<td>( x_s (J) )</td>
<td>1.47048637</td>
<td>1.447144</td>
<td>1.47049</td>
<td>1.47173</td>
</tr>
<tr>
<td>( x_s (J) )</td>
<td>0.903622843</td>
<td>0.903621</td>
<td>0.90362</td>
<td>0.902022</td>
</tr>
<tr>
<td>( x_s (J) )</td>
<td>0.205720642</td>
<td>0.205723</td>
<td>0.20573</td>
<td>0.206480</td>
</tr>
<tr>
<td>( g_s (X) )</td>
<td>-1.617-09</td>
<td>-1.8293967</td>
<td>0.0</td>
<td>-0.183590</td>
</tr>
<tr>
<td>( g_s (X) )</td>
<td>-1.325-05</td>
<td>-1.24747</td>
<td>0.0</td>
<td>-0.231748</td>
</tr>
<tr>
<td>( g_s (X) )</td>
<td>-0.08E-10</td>
<td>-0.001498</td>
<td>-5.55E-17</td>
<td>-0.000495</td>
</tr>
<tr>
<td>( g_s (X) )</td>
<td>-3.43298378</td>
<td>-3.429347</td>
<td>-3.432983788</td>
<td>-3.430043</td>
</tr>
<tr>
<td>( g_s (X) )</td>
<td>-0.07E-02</td>
<td>-0.079381</td>
<td>-0.0807296</td>
<td>-0.080986</td>
</tr>
<tr>
<td>( g_s (X) )</td>
<td>-2.36E-01</td>
<td>-0.235356</td>
<td>-0.235540</td>
<td>-0.235514</td>
</tr>
<tr>
<td>( g_s (X) )</td>
<td>-2.47E-04</td>
<td>-1.681355</td>
<td>-9.09494E-13</td>
<td>-58.64688</td>
</tr>
<tr>
<td>( f(X) )</td>
<td>1.724852321</td>
<td>1.720824</td>
<td>1.724385084</td>
<td>1.7273226</td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>Test Case</th>
<th>This paper</th>
<th>Best</th>
<th>Mean</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>1.72485231</td>
<td>1.73612022</td>
<td>2.46E-02</td>
<td></td>
</tr>
<tr>
<td>Case 2</td>
<td>1.72485231</td>
<td>1.73612022</td>
<td>2.46E-02</td>
<td></td>
</tr>
<tr>
<td>Case 3</td>
<td>1.72485231</td>
<td>1.73612022</td>
<td>2.46E-02</td>
<td></td>
</tr>
</tbody>
</table>

The optimization problem can be expressed as follows. Minimize

\[
f(x) = 0.6224x_1 x_2 + 1.7781x_2 x_3^2 + 3.1661x_1 x_3 + 19.84x_1^2 x_2
\]

Subject to

\[
\begin{align*}
g_s(X) &= 0.0193 x_3 - x_1 \leq 0 \\
\gamma(X) &= 0.00954 x_1 - x_2 \leq 0 \\
g_s(X) &= 1.296000 - x_1 x_2 - \frac{4}{3} x_3 \leq 0
\end{align*}
\]

### Table III

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \delta(T) )</td>
<td>0.79641436</td>
<td>0.812500</td>
<td>0.8125</td>
<td>0.812500</td>
</tr>
<tr>
<td>( \delta(T) )</td>
<td>0.39944942</td>
<td>0.437500</td>
<td>0.4375</td>
<td>0.437500</td>
</tr>
<tr>
<td>( \delta(R) )</td>
<td>40.0009194</td>
<td>40.091296</td>
<td>40.0984</td>
<td>40.097398</td>
</tr>
<tr>
<td>( \delta(L) )</td>
<td>190.801191</td>
<td>176.746500</td>
<td>176.6368</td>
<td>176.654047</td>
</tr>
<tr>
<td>( \delta(X) )</td>
<td>-5.04E-03</td>
<td>-0.000139</td>
<td>0.0</td>
<td>-0.000020</td>
</tr>
<tr>
<td>( \delta(X) )</td>
<td>-8.27E-03</td>
<td>-0.035949</td>
<td>-0.03588</td>
<td>-0.035891</td>
</tr>
<tr>
<td>( \delta(X) )</td>
<td>-595.450104</td>
<td>-116.382700</td>
<td>-5.820E-11</td>
<td>-27.886075</td>
</tr>
<tr>
<td>( \delta(X) )</td>
<td>-9.01988809</td>
<td>-9.375490</td>
<td>-9.3634</td>
<td>-9.345641</td>
</tr>
<tr>
<td>( \delta(X) )</td>
<td>2971.4063</td>
<td>6601.0777</td>
<td>6059.1313</td>
<td>6059.94634</td>
</tr>
</tbody>
</table>

### B. E02: Pressure vessel design problem

The pressure vessel design problem is a cylindrical vessel capped at both ends by hemispherical heads. The objective is to minimize the total cost, including the cost of the materials forming the welding. There are four design variables: Thickness of the shell \( T_s = x_t \), thickness of the head \( T_h = x_s \), the inner radius \( R = x_r \), and the length of the cylindrical section of the vessel \( L = x_1 \). \( T_s \) and \( T_h \) are discrete values which are integer multiples 0.0625 in., in accordance with the available thickness of rolled steel plates, \( R \) and \( L \) are continuous.
\[ g_i(X) = x_i - 240 \leq 0 \]

Where the design variables have to be in the following ranges:
\[ 0.0625 \leq x_1 \leq 6 \times 10^{-7.5}, \quad 0.0625 \leq x_2 \leq 6 \times 10^{-7.5}, \]
\[ 0 \leq x_3 \leq 200, \quad 0 \leq x_4 \leq 200. \]

The best solution found from our approach and the other three approaches are listed in Table III. The best result found from our approach is 5971.4003, which is the best solution ever reported. Before this paper, the best-known result was 6059.94634; the mean result for 30 independent runs is 6049.1590; and the standard deviation is 22.841537. As we can see, our approach performs well and is reasonably consistent. As above, the best results are generated with a particle size of 100 and the maximum iteration is 10,000.

From the Table IV, the proposed approach in this paper performs better in search quality than the other three. The mean results and standard deviations obtained by our approach are not as good as the others. By looking up the best solution generated from the original setting (100 particles, 10,000 maximum iterations) as stated above, it would appear that our approach needs more iterations to achieve more consistent results for this problem.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Approach</th>
<th>Best</th>
<th>Mean</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>This paper</td>
<td>2598.10542</td>
<td>6160.110/1</td>
<td>145.152</td>
</tr>
<tr>
<td></td>
<td>CPSSO [24]</td>
<td>6061.0777</td>
<td>6147.1332</td>
<td>86.4545</td>
</tr>
<tr>
<td>Case 2</td>
<td>This paper</td>
<td>5900.105542</td>
<td>6160.11071</td>
<td>145.152</td>
</tr>
<tr>
<td></td>
<td>HES-PSO [26]</td>
<td>6059.1313</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Case 3</td>
<td>This paper</td>
<td>6035.857686</td>
<td>6362.82540</td>
<td>266.134</td>
</tr>
<tr>
<td></td>
<td>Coello &amp; Montes [26]</td>
<td>6039.9463</td>
<td>6177.2333</td>
<td>130.93</td>
</tr>
</tbody>
</table>

Case 1 and Case 2: 40 particles, 5000 iterations; Case 3: 40 particles, 2000 iterations

C. E03: Spring design problem

This problem consists of minimizing the weight of a tension/compression spring subject to constraints of minimum deflection, shear stress, surge frequency, and limits on outside diameter and on design variables. The three design variables are: the wire diameter \( d = x_1 \), the mean coil diameter \( D = x_1 \), and the number of active coils \( N = x_2 \).

The formal statement of the problem is as follows:

**Minimize**

\[ f(X) = (x_3 + 2) x_3 x_1^2 \]

**Subject to**

\[ g_1(X) = 1 - \frac{4 x_1 x_3^2}{717585} \leq 0 \]
\[ g_2(X) = \frac{4 x_1^2 - 1}{12560 (x_2 x_1^3 - x_1)} - \frac{1}{5108 x_1^3} - 1 \leq 0 \]

The boundaries of the design variables are as follows:
\[ 0.05 \leq x_1 \leq 2, \quad 0.25 \leq x_2 \leq 1.3, \quad 2 \leq x_3 \leq 15. \]

The best solution found from our approach and other three approaches are listed in Table V. The best result found from our approach is 0.012665236, which is very close to the best-known solution 0.012656 and better than any of the three comparing approaches, the mean result from our approach is 0.012714534; and the standard deviation is 6.28E-05. These data demonstrate our approach performs really well in both search quality and consistency.

Table VI is a collection of data obtained by comparing with other approaches.

<table>
<thead>
<tr>
<th>Design variables</th>
<th>This paper</th>
<th>CPSO [24]</th>
<th>HES-PSO [26]</th>
<th>Coello &amp; Montes [26]</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 (d) )</td>
<td>0.051702169</td>
<td>0.051728</td>
<td>0.05146637</td>
<td>0.051989</td>
</tr>
<tr>
<td>( x_2 (D) )</td>
<td>0.357033166</td>
<td>0.357644</td>
<td>0.3518395</td>
<td>0.363965</td>
</tr>
<tr>
<td>( x_3 (N) )</td>
<td>11.27049760</td>
<td>11.244543</td>
<td>11.6806592</td>
<td>10.890522</td>
</tr>
<tr>
<td>( g_1(X) )</td>
<td>-0.078</td>
<td>-0.000845</td>
<td>-0.003366</td>
<td>-0.000013</td>
</tr>
<tr>
<td>( g_2(X) )</td>
<td>5.14E-9</td>
<td>-1.26E-05</td>
<td>-1.0970128</td>
<td>0.000021</td>
</tr>
<tr>
<td>( g_3(X) )</td>
<td>-0.05400797</td>
<td>-0.051300</td>
<td>-0.026318</td>
<td>-0.046138</td>
</tr>
<tr>
<td>( g_4(X) )</td>
<td>-0.72750978</td>
<td>-0.727090</td>
<td>-0.7312393</td>
<td>-0.722698</td>
</tr>
<tr>
<td>( f(X) )</td>
<td>0.012665236</td>
<td>0.0126747</td>
<td>0.0126614</td>
<td>0.012681</td>
</tr>
</tbody>
</table>

D. Discussion

As mentioned above, the best solutions generated from our approach are based on 100 particles and 10,000 maximum iterations, which forms a total number of 1,000,000 function evaluations. Although the maximum number of function evaluations looks large, the actual numbers of function evaluations are less than this maximum because a goal oriented programming concept is adopted in
the program. That is, once the goal is reached, evolution stops. This approach suits real world applications better. Therefore, we have performed an experiment to illustrate how many minimum iterations/generations are needed to reach the best-known solutions or to achieve even better solutions. Table VII is a summary of the experiment results based on 30 independent runs and with a particle population size of 100. In Table VII, a goal result is the existing best-known result. The minimum tolerance allowed δ indicates how close a solution is to a goal solution. If a solution is less than the goal solution (better than the best-known) or if its \( f - goal \leq \delta \) (close enough to the goal), evolution stops. Since the problem E02 has a larger magnitude than E01 and E03, the δ is set to a relative larger value.

According to the Table VII, the lowest number of iterations needed to reach the goal solution for the E01, E02 and E03 problems are 610, 680 and 183 respectively; the average number of iterations needed are 1291, 4210 and 1158 respectively. This mechanism makes the computation cost more rational. Under our multi-threaded Java programming implementation, the computation costs to these three engineering problems are not a big issue. The feasibility tolerance ε impacts on the results. A larger ε can sometimes make the constraints not fully satisfied but the search will be easier. We used a fairly small ε = 1.0E-9 to ensure the constraints are fully satisfied.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Goal (Best-Known)</th>
<th>Tolerance allowed</th>
<th>Lowest*</th>
<th>Average*</th>
<th>Highest*</th>
</tr>
</thead>
<tbody>
<tr>
<td>E01</td>
<td>1.74852</td>
<td>1.0E-04</td>
<td>610</td>
<td>129/1</td>
<td>3324</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(61000)</td>
<td>(159700)</td>
<td>(332400)</td>
</tr>
<tr>
<td>E02</td>
<td>6059.946</td>
<td>1.0E-01</td>
<td>680</td>
<td>4210</td>
<td>9998</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(68000)</td>
<td>(421000)</td>
<td>(999800)</td>
</tr>
<tr>
<td>E03</td>
<td>0.012665</td>
<td>1.0E-04</td>
<td>183</td>
<td>1158</td>
<td>4590</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(18300)</td>
<td>(115800)</td>
<td>(459000)</td>
</tr>
</tbody>
</table>

* indicate lowest iteration, average iteration and highest iteration needed (number in brackets indicates the number of function evaluation)

V. CONCLUSIONS

This paper has presented a multi-objective constraint handling method with the PSO algorithm for tackling constrained global optimization problems. The proposed approach adopts the concept of domination from multi-objective optimization, and uses a few selection rules and a goal oriented programming concept to improve search performance. Diversity is maintained by perturbing particles with a small probability. The simulation results to the three well-known engineering problems have been presented.

Compared with other work, our approach has achieved a better or very similar performance on the three well-known engineering problems. Remarkably, a best ever solution has been found for the engineering problem E02- Pressure vessel design problem. The proposed approach also performed well in consistency. Since the proposed approach has advantages of no problem-dependent parameters, and rational computation cost, it has potential to be applied to other constrained engineering optimization problems.

Our future work includes an extensive study of multi-objective constraint handling methods with the PSO algorithms and application of the proposed approach to more challenging real-world constrained engineering problems.

REFERENCES

to handle constraints.

multiobjective optimization technique," *Civil Engineering and

Evolutionary Multiobjective Optimization can be used for Goals and
Priorities based Optimization," in *Proceeding of AEB'02 Merida
Espana*, 2002.

[22] J. Horn, "Evolutionary Computation Applications F1.9 Multicriterion
decision making," in *Handbook of Evolutionary Computation: IOP

[23] M. Clerc and J. Kennedy, "The Particle Swarm: Explosion,
Stability, and Convergence in a Multidimensional Complex Space," *IEEE

optimization for constrained engineering design problems," *Engineering Application of Artificial Intelligence*, vol. 20, pp. 89-99,
2007.

Particle Swarm. Available:
http://www.swarmintelligence.org/papers/S182003Engineering.pdf,
