A Modified PSO with a Dynamically Varying Population and Its Application to the Multi-Objective Optimal Design of Alloy Steels

Qian Zhang and Mahdi Mahfouf

Abstract—In this paper, a new mechanism for dynamically varying the population size is proposed based on a previously modified PSO algorithm (nPSO). This new algorithm is extended to the multi-objective optimisation case by applying the Random Weighted Aggregation (RWA) technique and by maintaining an archive for preserving the suitable Pareto-optimal solutions. Both the single objective and multi-objective optimisation algorithms were tested using well-known benchmark problems. The results show that the proposed algorithms outperform some of the other salient Evolutionary Algorithms (EAs). The proposed algorithms were further applied successfully to the optimal design problem of alloy steels, which aims at determining the optimal heat treatment regime and the required weight percentages for chemical composites to obtain the desired mechanical properties of steel hence minimising production costs and achieving the overarching aim of ‘right-first-time production’ of metals.

I. INTRODUCTION

PARTICLE swarm optimisation (PSO) is a powerful evolutionary computation technique that was originally introduced by Kennedy and Eberhart [1]. It was developed via the simulation of a simplified animal social behaviour of birds flocking and fish schooling. Since its introduction in 1995, the PSO technique is becoming very popular due to its simplicity of implementation and ability to quickly converge to a reasonably good solution.

In the previously reported work [2], an alternative structure for PSO, named ‘nPSO’, was introduced. In nPSO, a ‘momentum term’ was proposed to replace the original inertia term of the standard PSO, which can help to avoid premature convergence and encourage the particles to jump out of any local optimum. To provide the particles of nPSO with more adaptability, a separate momentum weight was assigned to each particle as it dynamically adjusts itself according to the particle’s own flying experience.

In the paper, the nPSO algorithm will be further improved, whereby the population size of this algorithm can also be dynamically varied according to the algorithm’s search performance in the optimisation process. The modified algorithm will then be extended to a multi-objective optimisation case by applying the Random Weighted Aggregation (RWA) [3] technique and by maintaining an archive for preserving the suitable Pareto-optimal solutions. Furthermore, the proposed multi-objective optimisation algorithm will be applied to the optimal design problem of alloy steels, which aims at determining the optimal heat treatment regime and the required weight percentages for chemical composites to obtain the desired mechanical properties of steel, such as Ultimate Tensile Strength (UTS) and Reduction of Area (ROA). This study will also include economic factors, such as the costs associated with the composites and the processing operations.

The remaining parts of this paper are organised as follows. Section 2 will introduce nPSO as well as the proposed population varying mechanism in details. In this section, a set of comparison experiments will be carried-out to examine the performance of the new algorithm. In Section 3, the new algorithm will be extended to include the multi-objective optimisation case. The results of applying this new multi-objective optimisation algorithm to well-known benchmark problems, namely the ZDT series and the DTLZ series problems, will also be presented. In Section 4, the new proposed algorithms will be applied to the optimal design of alloy steels, which will consider both the mechanical properties and the economic factors. Finally, summary remarks will be given in Section 5.

II. NPSO WITH A VARYING POPULATION SIZE

A. Algorithm Formulation of nPSO

In PSO (including nPSO), each particle represents an alternative solution in the multi-dimensional search space. Initially, a population of particles is generated with random positions and random velocities. Each particle then flies through the search space with a velocity, constantly updated according to its own flying experience and its companions’ flying experience. It is expected that the particles will have a tendency to fly towards better search areas during the search process.

Specifically, for the nPSO algorithm, the position vector and velocity vector of the $i$th particle in a $D$-dimensional search space can be represented as

$$x_i = [x_{i1}, x_{i2}, x_{i3}, ..., x_{iD}]^T$$

and

$$v_i = [v_{i1}, v_{i2}, v_{i3}, ..., v_{iD}]^T$$

respectively. According to a predefined fitness function, the best previous position of the $i$th particle is

$$p_i = [p_{i1}, p_{i2}, p_{i3}, ..., p_{iD}]^T$$

(its corresponding fitness value is known as the ‘personal best’ $p_{best}$) and the fittest position among all the particles found so far is

$$p_d = [p_{d1}, p_{d2}, p_{d3}, ..., p_{dD}]^T$$

(its corresponding fitness value is called the ‘global best’ $g_{best}$). The velocities and positions of the particles are updated according to the following equations [2]:

$$v_{id}(t+1) = w_{id}(t+1) \times r_{1id}(t+1) \times v_{max,d} + c_1 \times r_{2id}(t+1)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$$
\[ x_d(t+1) = x_d(t) + v_d(t+1) \]

where \( d = 1, 2, \ldots, D; t \) is the index of the iteration; \( w_d \) is the momentum weight of the \( i \)th particle in the \( d \)th search space dimension; \( r_{1,d}(t), r_{2,d}(t) \) and \( r_{3,d}(t) \) are three uniformly distributed random variables in the range \([0, 1]\); \( v_{max} = [v_{max1}, v_{max2}, v_{max3}, \ldots, v_{maxD}] \) is an upper limit for the achievable velocity of the particles; \( c_1 \) and \( c_2 \) are positive constants known as acceleration coefficients.

The momentum term, which is the first component of Equation (1), aims at giving one particle extra adjustable momentum in its optimal search process to keep a balance between exploration (global search) and exploitation (local search). When a particle converges to one solution, which is judged by whether the velocity of the particle is too small or not, it may be trapped in a local minimum. In this situation, the momentum weight \( w_{id} \) is set at a big value ‘1’ to encourage the particle to jump out from the local minimum. When a particle does not converge, the momentum weight is dynamically adjusted according to the particle’s own search experience: More specifically, if the particle can find a better solution in the previous generation, it may be in a local optimum space and the momentum should be increased to enhance the global search ability; if the particle cannot find a better solution in the previous generation, it may wander in a large space and the momentum should be reduced to enhance the local search ability. This mechanism is proved to be effective in balancing the global search and local search and makes the optimisation search more adaptive.

For minimisation problems, the momentum weight \( w_{id} \) is varied in the optimisation process as follows:

\[
\begin{align*}
 w_{id}(t+1) &= \left\{ \begin{array}{ll}
 1, & \text{if } v_{id}(t) \leq \epsilon \times v_{max,d} \text{ and } pos_{id}(t+1) = 1; \\
 w_{id}(t) \times m_1, & \text{if } v_{id}(t) > \epsilon \times v_{max,d} \text{ and } f(x_i(t)) \geq f(p_i(t-1)); \\
 w_{id}(t) \times m_2, & \text{if } v_{id}(t) > \epsilon \times v_{max,d} \text{ and } f(x_i(t)) < f(p_i(t-1)); \\
 w_{id}(t+1) = c_2 \times r_3 \times (p_{id}(t) - x_{id}(t)) + c_1 \times r_1 \times (p_{id}(t) - x_{id}(t)), & \text{if } v_{id}(t) \leq \epsilon \times v_{max,d} \text{ and } pos_{id}(t+1) = 0; \\
 w_{id}(t+1) = c_2 \times r_3 \times (p_{id}(t) - x_{id}(t)) + c_1 \times r_1 \times (p_{id}(t) - x_{id}(t)) & \text{otherwise.}
\end{array} \right.
\]

where \( \epsilon \) is a small positive coefficient; \( m_1 \) and \( m_2 \) are two scaling parameters used to control the range of magnitudes of the momentum weight varying, where \( m_1 \) is a positive coefficient smaller than 1 and \( m_2 \) is a positive coefficient larger than 1; \( pos_{id} \) is a discrete variable used to control the direction in which the velocity of one particle should be refreshed, where its value is 0 or 1; \( f(x) \) is the fitness function of the minimisation problem.

Normally, \( \epsilon \) is set to \( 10^{-10} \) without any loss of generality, which means that if the velocity of a particle is smaller than \( V_{max} \times 10^{-10} \), then the particle is assumed to be converging. In addition, \( m_1 \) is generally set to 0.5 and \( m_2 \) is set to 2, which means that the momentum weight will reduce to half of the previous value or increase up to double of the previous value.

The position parameter \( pos_{id} \) is used to control the direction in which the velocity of the particle should be refreshed. In this work, three strategies for setting the value for \( Pos_{i} = (pos_{i1}, pos_{i2}, pos_{i3}, \ldots, pos_{id}) \) are proposed as follows:

1. **One-directional refresh mechanism:** At every iteration, and for each particle, only one of the elements in the position parameter vector \( Pos \), will be randomly chosen and set to the value ‘1’; other elements will be set to the value ‘0’.

2. **Multiple-directional refresh mechanism:** The number of elements and the positions of elements that will be set to 1 are both randomly generated; other elements in the position parameter vector will be set to 0.

3. **All-directional refresh mechanism:** All the elements of \( Pos \) will be set to the value ‘1’ at all times.

For a broad adaptation to various problems, it is recommended to use both the one-directional and multiple-directional refresh mechanisms simultaneously.

The second part of Equation (1) is viewed as the ‘memory’ component. It represents the personal thinking of one particle. This component attracts the particle to fly towards its own best positions found so far. The third part of Equation (1) is known as the ‘cooperation’ component, which represents the cooperative effect of the particles in the optimisation search. This component always encourages the particle to move towards the global best position.

Generally, a maximum velocity vector \( v_{max} = [v_{max1}, v_{max2}, v_{max3}, \ldots, v_{maxD}] \) is defined and acts as the upper limit for the achievable velocity of the particles, where \( v_{max,d} \), \( d = 1, 2, \ldots, D \), are positive numbers. It works as follows:

\[
\begin{align*}
 \text{If } v_{id} > v_{max,d}, \text{ then } v_{id} &= v_{max,d} \\
 \text{Else if } v_{id} < -v_{max,d}, \text{ then } v_{id} &= -v_{max,d}.
\end{align*}
\]

However, the particles may still occasionally fly to a position beyond the predefined search space and produce an invalid solution. In this case, a simple handling method is used in most of the PSO algorithms, which works as follows:

\[
\begin{align*}
 \text{If } x_{id} > x_{max,d}, \text{ then } x_{id} &= x_{max,d} \\
 \text{Else if } x_{id} < x_{min,d}, \text{ then } x_{id} &= x_{min,d}.
\end{align*}
\]

where \( x_{max,d} \) and \( x_{min,d} \) are the maximum and minimum bands of the \( d \)th dimension of the search space.

**B. The Method of Dynamically Varying the Population**

The population size is very important for PSOs’ performance in terms of robustness and computational efficiency. In the standard PSO, the population size is set to be a fixed value and remains constant throughout the entire optimisation search. The drawbacks of this method are as follows: First, the optimal population size depends on the difficulty of the optimisation problem. Thus different optimisation problems need different population sizes. More often than not, one does not have the necessary knowledge/expertise for setting an optimisation problem with an adequate population size. Second, even in one single optimisation process, different search stages need different population sizes. For instance, the global search needs a relatively big population size while the local search only needs a relatively small one.

To overcome the above problems, a new search mechanism is proposed which consists of dynamically varying the population size of PSO during the search process. The basic ‘rationale’ behind this new method is as follows: First, if the
PSO cannot find new better solutions in the previous search iteration, then this would suggest that the search ability of the algorithm is weak and thus should be enhanced by increasing the population size; Second, if the PSO can keep finding new better solutions, then this would also suggest that the population size is big enough or even too big. In this case, the population size may be kept the same as before or may be reduced.

Specifically, the population varying method used in this paper works as follows (for minimisation problems, for instance):

1. If \( g_{\text{best}}(t) < g_{\text{best}}(t-1) \) and \( N_p > N_{l_\text{lower}} \), then remove the particle with the worst performance from the population; If \( g_{\text{best}}(t) \geq g_{\text{best}}(t-1) \) and \( N_p < N_{l_\text{upper}} \), then copy the particle with the best performance and add it to the new population. (6)

where \( N_p \) is the number of particles; \( N_{l_\text{lower}} \) and \( N_{l_\text{upper}} \) are the lower and upper limits for the particle size.

In summary, the entire nPSO algorithm, which includes the Varying Population, is named throughout as 'nPSO-VP'. It can be described via the following sequential procedure:

1. Initialise the swarm by assigning a random position to each particle within the problem hyperspace.
2. According to the predefined objective function, evaluate the fitness for each particle.
3. For each individual particle, compare the particle’s fitness value with its \( p_{\text{best}} \). If the current value is better than the \( p_{\text{best}} \), then set the current value as the new \( p_{\text{best}} \) and set the current particle’s position \( x_i \) as the new \( p_i \).
4. Identify the particle that has the best fitness value. The value of its \( p_{\text{best}} \) is identified as \( g_{\text{best}} \) and its \( p_i \) is identified as \( p_g \).
5. Apply Method (6) to vary the population size.
6. Update the velocities and positions of all the particles using Equations (1) and (2).
7. If the velocity of one particle exceeds the upper limit \( v_{\text{max}} \), implement Method (4).
8. If the position of one particle exceeds the search bounds, then implement Method (5).
9. Repeat Step 2 to Step 8 until a stopping criterion (e.g., a maximum number of iterations or a sufficiently good fitness value) is achieved.

C. Comparison between nPSO-VP and Other Evolutionary Algorithms

For a comprehensive comparison, experiments were carried-out using nPSO-VP, two versions of the standard PSO, which apply a linearly decreasing inertia weight (PSO-LD) [4] and a randomly varying inertia weight (PSO-RV) [5], and other three salient Evolutionary Algorithms, which are the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [6], the Differential Evolution (DE) [7] and the Generalised Generation Gap model with the Parent-Centric Recombination operator (G3+PCX) [8].

The parameter settings for these algorithms were as follows:

1. **nPSO**: \( N_p = 10; N_{l_\text{lower}} = 2; N_{l_\text{upper}} = 100; c_1 = c_2 = 1.8; m_1 = 0.5 \) and \( m_2 = 2; c = 10^{-10}; pos_{\text{init}} \) was updated using both the one-directional refresh mechanism (with the 50% probability of usage) and the multiple-directional refresh mechanism (with the 50% probability of usage).
2. **PSO-LD**: Population size \( N = 10 \times D \), where \( D \) is the dimension of the test problem; \( c_1 = c_2 = 1.8 \); the inertia weight \( w \) varied from 0.9 at the beginning of the search to 0.4 at the end of the search [4].
3. **PSO-RV**: \( N = 10 \times D; c_1 = c_2 = 1.8; w \) changed according to the equation: \( w = 0.4 + \text{rand()} \times 0.5 \), where \( \text{rand()} \) is a uniformly distributed random number within the range \([0, 1]\) [5].
4. **CMA-ES**: There are 8 parameters to be predefined for this algorithm. All settings followed the instructions given in [9]. For instance, the population size \( \lambda = 4 + \text{floor}(3 \times \ln D) \), the parent number \( \mu = \text{floor}(\lambda/2) \), etc., where \( \text{floor}(x) \) is the function that allows to round-off \( x \) to the nearest integer towards \(-\infty\).
5. **DE**: The DE/rand/1 scheme was employed. The parameter settings followed the instructions in [10]. The population size \( N = 10 \times D \); the crossover probability \( CR = 0.9 \) and the weighting factor \( F = 0.8 \).
6. **G3+PCX**: Following the papers by [8], [11], the population size \( N = 10 \times D \); the parent size was set to 3; the offspring size was set to 2 and the replacement size was set to 2. For the PCX operator, the distribution parameter \( \sigma_c = 0.1 \) and \( \sigma_g = 0.1 \).

In these experiments, 9 well-characterised benchmark functions were employed as the test set. All these functions represent minimisation problems; their expressions and the summary of their features about separability and multimodality are as follows:

1. **Sphere function** (Unimodal, Separable and \( D\)-dimensional):
\[
    f_{\text{sphere}}(x) = \sum_{i=1}^{D} x_i^2, x_i \in [-10,10],
\]
\[
    \min(f_{\text{sphere}}) = f_{\text{sphere}}(0,...,0) = 0.
\]
2. **Schwefel's function 1.2** (Unimodal, Non-separable and \( D\)-dimensional):
\[
    f_{\text{schwefel_1.2}}(x) = \sum_{i=1}^{D} \left( \sum_{j=1}^{D} x_j \right)^2, x_i \in [-10,10],
\]
\[
    \min(f_{\text{schwefel_1.2}}) = f_{\text{schwefel_1.2}}(0,...,0) = 0.
\]
3. **Rosenbrock's function** (Multimodal, Non-separable and \( D\)-dimensional):
\[
    f_{\text{rosenbrock}}(x) = \sum_{i=1}^{D-1} \left[ 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right], x_i \in [-2,2],
\]
\[
    \min(f_{\text{rosenbrock}}) = f_{\text{rosenbrock}}(1,...,1) = 0.
\]
4. **Schwefel's function 2.26** (Multimodal, Separable and \( D\)-dimensional):
\[
    f_{\text{schwefel_2.26}}(x) = -\sum_{i=1}^{D} \left[ x_i \sin(\sqrt{|x_i|}) \right], x_i \in [-500,500],
\]
\[
    \min(f_{\text{schwefel_2.26}}) = f_{\text{schwefel_2.26}}(420.9687,...,420.9687).
\]
5) **Rastrigin’s function** (Multimodal, Separable and D-dimensional):
\[
 f_{\text{Ras}}(x) = \sum_{i=1}^{D} \left( x_i^2 - 10 \cos(2\pi x_i) + 10 \right), \quad x_i \in [-5, 5],
\]
\[
 \min(f_{\text{Ras}}) = f_{\text{Ras}}(0,\ldots,0) = 0.
\]

6) **Ackley’s function** (Multimodal, Non-separable and D-dimensional):
\[
 f_{\text{Ack}}(x) = -20 \exp \left( -0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2} \right) - \exp \left( \frac{1}{D} \sum_{i=1}^{D} \cos (2\pi x_i) \right) + 20 + e
\]
\[
 x_i \in [-30, 30], \quad \min(f_{\text{Ack}}) = f_{\text{Ack}}(0,\ldots,0) = 0.
\]

7) **Griewank’s function** (Multimodal, Non-separable and D-dimensional):
\[
 f_{\text{Gri}}(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1,
\]
\[
 x_i \in [-600, 600], \quad \min(f_{\text{Gri}}) = f_{\text{Gri}}(0,\ldots,0) = 0.
\]

8) **Bohachevsky’s function** (Multimodal, Separable and D-dimensional):
\[
 f_{\text{Boh}}(x) = \sum_{i=1}^{D-1} \left( x_i^2 + 2 x_{i+1}^2 - 3 \cos(3\pi x_i) - 4 \cos(4\pi x_{i+1}) + 7 \right)
\]
\[
 x_i \in [-15, 15], \quad \min(f_{\text{Boh}}) = f_{\text{Boh}}(0,\ldots,0) = 0.
\]

9) **Schaffer’s function** (Multimodal, Non-separable and D-dimensional):
\[
 f_{\text{Sch}}(x) = \sum_{i=1}^{D} \left( x_i^2 + x_i^2 \right)^{0.25} \left( \sin^2 \left( 50 (x_i^2 + x_i^2)^{0.1} \right) + 1 \right),
\]
\[
 x_i \in [-100, 100], \quad \min(f_{\text{Sch}}) = f_{\text{Sch}}(0,\ldots,0) = 0.
\]

For every individual experiment, D was set to be 30 and the result was obtained after 20 runs. For each run, the maximum function evaluations number was set to 10^6 and the optimisation process was regarded as successful and stopped when the best solution Fb satisfied: Fb < 10^{-5} if the true global minimum Gb = 0 or (Fb - Gb) / Gb < 10^{-5} if Gb ≠ 0.

Fig. 1 gives an example that shows the situation of the population size varying during the optimisation process when optimising \( f_{\text{Ras}} \) and \( f_{\text{Gri}} \). Table I shows the optimisation results for different algorithms on various problems. From this table, one can observe the following:

1) For unimodal problems, nPSO-VP performs best in optimising \( f_{\text{Boh}} \) and CMA-ES performs best in optimising \( f_{\text{Sch}1.2} \) since they use the fewest function evaluations. For \( f_{\text{Sch}1.2} \), nPSO-VP performs the third best among all the six algorithms.

2) For high-dimensional multimodal problems, nPSO-VP performs better than other algorithms in most of the situations. For instance, as far as \( f_{\text{Ras}}, f_{\text{Ack}} \) and \( f_{\text{Gri}} \) are concerned, nPSO-VP is able to locate the global optimum with the fewest function evaluations; for \( f_{\text{Ras}}, f_{\text{Sch}2.26} \) and \( f_{\text{Sch}} \), though nPSO-VP cannot find the global optimum, it still performs well with the minimal fitness values.

### III. EXTENSION OF nPSO-VP TO MULTI-OBJECTIVE OPTIMISATION PROBLEMS

To extend the nPSO-VP algorithm for optimising multi-objective problems, the Random Weighted Aggregation (RWA) technique [3] is employed and an archiving approach is also included to preserve the Pareto-optimal solutions.

**A. The Random Weighted Aggregation Approach**

Assume a multi-objective problem that consists of finding a vector
\[
 X^* = (x_1^*, x_2^*, \ldots, x_D^*)
\]
that will optimise the following vector function:
\[
 \bar{f}(X) = (f_1(X), f_2(X), \ldots, f_D(X)).
\]

**TABLE I**

<table>
<thead>
<tr>
<th>Function</th>
<th>nPSO-VP</th>
<th>PSO-LD</th>
<th>PSO-RV</th>
<th>CMA-ES</th>
<th>DE</th>
<th>G3+PCX</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_{\text{Boh}} )</td>
<td>(N/A)</td>
<td>(N/A)</td>
<td>(N/A)</td>
<td>(N/A)</td>
<td>(N/A)</td>
<td>(N/A)</td>
</tr>
<tr>
<td>( f_{\text{Sch}1.2} )</td>
<td>(N/A)</td>
<td>(79.1666)</td>
<td>(20.0000)</td>
<td>(N/A)</td>
<td>(1.8527)</td>
<td>(N/A)</td>
</tr>
<tr>
<td>( f_{\text{Ack}} )</td>
<td>(N/A)</td>
<td>(0.0158)</td>
<td>(N/A)</td>
<td>(N/A)</td>
<td>(N/A)</td>
<td>(N/A)</td>
</tr>
<tr>
<td>( f_{\text{Sch}2.26} )</td>
<td>(N/A)</td>
<td>(10793)</td>
<td>(-9341)</td>
<td>(10202)</td>
<td>(6665)</td>
<td>(N/A)</td>
</tr>
<tr>
<td>( f_{\text{Ras}} )</td>
<td>(N/A)</td>
<td>(6.6162e-4)</td>
<td>(24.4833)</td>
<td>(31.2336)</td>
<td>(0.0158)</td>
<td>(N/A)</td>
</tr>
<tr>
<td>( f_{\text{Ack}} )</td>
<td>(N/A)</td>
<td>(137040)</td>
<td>(535500)</td>
<td>(93650)</td>
<td>(8575)</td>
<td>(694560)</td>
</tr>
<tr>
<td>( f_{\text{Sch}} )</td>
<td>(N/A)</td>
<td>(4209)</td>
<td>(493425)</td>
<td>(146983)</td>
<td>(5586)</td>
<td>(58674)</td>
</tr>
<tr>
<td>( f_{\text{Gri}} )</td>
<td>(N/A)</td>
<td>(84377)</td>
<td>(485389)</td>
<td>(141330)</td>
<td>(510180)</td>
<td>(N/A)</td>
</tr>
<tr>
<td>( f_{\text{Sch}} )</td>
<td>(N/A)</td>
<td>(228900)</td>
<td>(30.3990)</td>
<td>(21.8394)</td>
<td>(31.2336)</td>
<td>(31.2336)</td>
</tr>
</tbody>
</table>

\begin{tabular}{c|c|c|c|c|c|c}
<table>
<thead>
<tr>
<th>Function</th>
<th>nPSO-VP</th>
<th>PSO-LD</th>
<th>PSO-RV</th>
<th>CMA-ES</th>
<th>DE</th>
<th>G3+PCX</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_{\text{Boh}} )</td>
<td>(N/A)</td>
<td>(N/A)</td>
<td>(N/A)</td>
<td>(N/A)</td>
<td>(N/A)</td>
<td>(N/A)</td>
</tr>
<tr>
<td>( f_{\text{Sch}1.2} )</td>
<td>(N/A)</td>
<td>(100%)</td>
<td>(100%)</td>
<td>(100%)</td>
<td>(100%)</td>
<td>(100%)</td>
</tr>
<tr>
<td>( f_{\text{Ack}} )</td>
<td>(N/A)</td>
<td>(940560)</td>
<td>(940560)</td>
<td>(142.8754)</td>
<td>(N/A)</td>
<td>(N/A)</td>
</tr>
<tr>
<td>( f_{\text{Sch}2.26} )</td>
<td>(N/A)</td>
<td>(137040)</td>
<td>(535500)</td>
<td>(93650)</td>
<td>(8575)</td>
<td>(694560)</td>
</tr>
<tr>
<td>( f_{\text{Ras}} )</td>
<td>(N/A)</td>
<td>(137040)</td>
<td>(535500)</td>
<td>(93650)</td>
<td>(8575)</td>
<td>(694560)</td>
</tr>
<tr>
<td>( f_{\text{Ack}} )</td>
<td>(N/A)</td>
<td>(137040)</td>
<td>(535500)</td>
<td>(93650)</td>
<td>(8575)</td>
<td>(694560)</td>
</tr>
<tr>
<td>( f_{\text{Sch}} )</td>
<td>(N/A)</td>
<td>(137040)</td>
<td>(535500)</td>
<td>(93650)</td>
<td>(8575)</td>
<td>(694560)</td>
</tr>
<tr>
<td>( f_{\text{Ack}} )</td>
<td>(N/A)</td>
<td>(137040)</td>
<td>(535500)</td>
<td>(93650)</td>
<td>(8575)</td>
<td>(694560)</td>
</tr>
<tr>
<td>( f_{\text{Sch}} )</td>
<td>(N/A)</td>
<td>(137040)</td>
<td>(535500)</td>
<td>(93650)</td>
<td>(8575)</td>
<td>(694560)</td>
</tr>
<tr>
<td>( f_{\text{Ack}} )</td>
<td>(N/A)</td>
<td>(137040)</td>
<td>(535500)</td>
<td>(93650)</td>
<td>(8575)</td>
<td>(694560)</td>
</tr>
<tr>
<td>( f_{\text{Sch}} )</td>
<td>(N/A)</td>
<td>(137040)</td>
<td>(535500)</td>
<td>(93650)</td>
<td>(8575)</td>
<td>(694560)</td>
</tr>
</tbody>
</table>
The Weighted Aggregation is one of the most common approaches for solving multi-objective problems. In this type of approach, all the objectives are summed to a weighted combination as follows:

\[ F = \sum_{i=1}^{k} w_i f_i(X), \quad \sum_{i=1}^{k} w_i = 1 \]  

where \( w_i \), \( i = 1, 2, \ldots, k \), are non-negative weights.

In the Conventional Weighted Aggregation (CWA) method, the above weights are fixed during the optimisation process. By using CWA, only a single Pareto-optimal solution can be obtained in every optimisation run. If one wishes to obtain different Pareto solutions, the algorithm has to be repeated several times with different weights settings. In addition, this method cannot locate the Pareto solutions when there are concave regions in the true Pareto front.

Random Weighted Aggregation (RWA) can overcome the limitations of CWA. In the RWA method, the weights are modified after every certain number of iterations during the optimisation. The weights are defined by the following equation:

\[ w_i(t) = \begin{cases} \frac{\text{rand}_i(t)}{\sum_{j=1}^{k} \text{rand}_j(t)}, & \text{if } \text{rem}(t, H) = 1; \\ w_i(t-1), & \text{else.} \end{cases} \]  

where \( t \) is the index of iteration and \( H \) is the frequency parameter that allows to round-off the weight changing; \( \text{rand}(t) \) is a function to create a uniformly distributed random value in the range \([0, 1]\); \( \text{rem}(t, H) \) is a function to obtain the remainder from dividing \( t \) by \( H \).

In this paper, the frequency parameter is calculated using the following equation:

\[ H = \text{round}(E_{\text{max}} / (4 * \text{Obj})) \]  

where \( \text{round}(x) \) is the function that allows to round-off \( x \) to the nearest integer, \( E_{\text{max}} \) is the maximum function evaluation number and \( \text{Obj} \) is the number of objectives.

B. Archive Design

In the RWA method, the population cannot keep all the found Pareto solutions. Thus, an archive is used to record the Pareto solutions found so far during the optimisation search. To update the archive with appropriate Pareto solutions, a non-dominated selection and a diversity selection mechanism are employed. The non-dominated selection aims to obtain the Pareto-optimal solutions from the candidates. This is easy to implement. The diversity selection tends to obtain the solutions with a good diversity from the candidates. In this paper, a simple method is proposed to achieve this purpose, which works as follows:

1) If the number of solutions in the present archive is more than the predefined maximum number, go to Step 2; else terminate this selection and return.

2) For every solution in the archive, calculate the value of its closeness criterion. The closeness criterion of the \( i \)th solution is defined as follows:

\[ \text{cri}_i = d_{i1} + d_{i2} \]  

where \( d_{i1} \) is the distance between the \( i \)th solution to its closest neighbour and \( d_{i2} \) is the distance between the \( i \)th solution to its second closest neighbour.

3) Find the solution with the minimum criterion value and remove it from the archive.

4) Go to Step 1.

C. Algorithm Formulation

By applying the RWA method and by maintaining an archive for preserving the Pareto-optimal solutions, the nPSO-VP is thus extended to a multi-objective optimisation algorithm, named as ‘MO-nPSO-VP’. In summary, the entire MO-nPSO-VP can be described via the following procedure:

1) Randomly generate the initial weights for the optimisation objectives.
2) Optimise the related problem, whose objective is the weighted sum of the multiple objectives, using nPSO-VP for one iteration.
3) Add the particles’ present positions to the archive as the candidate solutions.
4) Execute the non-dominated selection to the archive.
5) Execute the diversity selection to the archive.
6) Vary the weights of the objectives using the method RWA.
7) Repeat Step 2 to Step 6 until a stopping criterion (e.g., a maximum number of iterations or a sufficiently good fitness value) is achieved.

D. Experiments Based on Benchmark Test Problems

To validate the effectiveness of the proposed multi-objective optimisation algorithm, a set of experimental tests were carried-out using the well-known multi-objective optimisation problems - the ZDT series benchmark problems [12] and DTLZ series problems [13].

The ZDT series benchmark functions include 2 minimisation objectives and they are considered to be difficult to optimise, especially ZDT2, ZDT3 and ZDT4 [12]. The maximal function evaluation for every experiment was set to 25000, which is the same as the experiments configuration referred to the experiments in [14]. The configuration of the algorithm was the same as the one used in Section 2.3.

Fig. 2 shows the graphical results produced by MO-nPSO-VP. The ‘round’ dots are the solutions obtained using the new algorithm. It can be observed that the algorithm possesses very good convergence properties while maintaining a good diversity among the Pareto solutions. Compared with the optimisation results in [14], which used PAES, SPEA and NSGA-II, MO-nPSO-VP performs as well as and sometimes better than the other three salient EAs in terms of both accuracy and diversity.

In the second experiment, MO-nPSO-VP was used to optimise the DTLZ series problems [13]. All the DTLZ problems were set so as to include three objectives. For a meaningful comparison, MO-nPSO-VP used the same numbers of function evaluations as the experiments in [13]. The parameters of the algorithm were set the same as the previous experiments. Fig. 3 shows the 3-D Pareto fronts obtained by MO-nPSO-VP. It can be seen that, in most of the situations, the algorithm can convergence to the real
Pareto-optimal front with a good diversity among the solutions. Compared with the optimisation results in [13], MO-nPSO-VP performs as well as and more often than not better than the salient EAs, SPEA2 and NSGA-II, both in terms accuracy and diversity.

Fig. 2. Pareto fronts obtained by MO-nPSO-VP based on ZDT series problems

Fig. 3. Pareto fronts obtained by MO-nPSO-VP based on DTLZ series problems

Fig. 4. An example of the population size varying: (a) Optimising ZDT1; (b) Optimising DTLZ1

IV. OPTIMAL ALLOY DESIGN CONSIDERING BOTH THE MECHANICAL PROPERTIES AND THE ECONOMICAL FACTORS

In the steel industry, determining the optimal heat treatment regime and the required weight percentages for the chemical composites to obtain the desired mechanical properties of the steel is always a challenging multi-objective optimisation problem. Usually, some objectives may conflict with each other, such as the ultimate tensile strength (UTS) and the Reduction of Area (ROA).

In the previously carried-out work [15], intelligent models based on fuzzy systems were developed to predict the mechanical test results for the steels covered by a wide range of training data. Fig. 5 shows the prediction results of one UTS model and one ROA model. In the following studies, all alloy design experiments are conducted using the proposed algorithms, nPSO-VP and MO-nPSO-VP, based on these two developed fuzzy models. The parameter configurations of the algorithms were set the same as the experiments in previous sections.

Fig. 5. The prediction performance of the UTS model and the ROA models used in this paper

In this study, not only the mechanical properties but also the economical factors will be considered, which aims to minimise the production costs. The production costs of heat-treated steels include the costs of the addition of alloying elements, such as Mn, Cr, Mo, etc. and the costs of energy consumption during the heat-treatment process. The factors contributing to the cost of heat treatment operation are summarised in Tables II, III [16].

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>CONTRIBUTION OF TEMPERING (ANNEALING) TO THE COST OF HEAT TREATMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Cost (US$: 1.3GJ/tonne at 600° C)</td>
</tr>
<tr>
<td>Annealing (tempering)</td>
<td>4.88</td>
</tr>
</tbody>
</table>
A. The Optimal Design Considering both UTS and the Cost

According to the contribution of the chemical composites and the tempering process to the cost of heat-treated steels, an objective function to reflect such costs was designed as follows:

$$\text{Cost} = \frac{18\text{Mn} + 42\text{Cr} + 52\text{Mo} + 4.88\text{Temp}/600}{100}$$ (13)

By taking into account such economic consideration, the problem of designing an alloy steel with a predefined target UTS property becomes a two-objective optimisation problem described as follows:

Objective 1: Minimise $J_1 = \left(\frac{\text{UTS} - \text{UTS}_{\text{Target}}}{900}\right)^2$

Objective 2: Minimise $J_{\text{cost}}$

Fig. 6 displays the obtained Pareto-optimal solutions in the objective space with the UTS target value $\text{UTS}_{\text{Target}} = 900$ (MPa). 10 various solutions around the UTS target value are selected from the Pareto-optimal solutions and listed in Table IV.

B. The Optimal Design Considering UTS, ROA and the Cost

Taking into account three factors, i.e. UTS, ROA and the cost of the heat treatment, the problem of designing an alloy steel can be described as follows:

Objective 1: Minimise $J_1 = \left(\frac{\text{UTS} - \text{UTS}_{\text{Target}}}{900}\right)^2$

Objective 2: Minimise $J_2 = \left(\frac{\text{ROA} - \text{ROA}_{\text{Target}}}{60}\right)^2$

Objective 3: Minimise $J_{\text{cost}}$

Fig. 6. The performance of the Pareto-optimal solutions for the design problem of $\text{UTS}_{\text{Target}} = 900$ (MPa) and minimising the heat treatment cost with respect to (a) Objective 1 and Objective 2; (b) UTS and Cost

### Table III

<table>
<thead>
<tr>
<th>Composite</th>
<th>Cost (US$ per tonne)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manganese</td>
<td>18</td>
</tr>
<tr>
<td>Chromium</td>
<td>42</td>
</tr>
<tr>
<td>Molybdenum</td>
<td>52</td>
</tr>
</tbody>
</table>

### Table IV

<table>
<thead>
<tr>
<th>Solutions</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>C (wt%)</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
</tr>
<tr>
<td>Mn (wt%)</td>
<td>1.658</td>
<td>0.818</td>
<td>0.800</td>
<td>0.830</td>
<td>0.789</td>
<td>0.805</td>
<td>0.869</td>
<td>0.900</td>
<td>0.780</td>
<td>0.685</td>
</tr>
<tr>
<td>Cr (wt%)</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>Mo (wt%)</td>
<td>0.010</td>
<td>0.171</td>
<td>0.156</td>
<td>0.132</td>
<td>0.136</td>
<td>0.096</td>
<td>0.045</td>
<td>0.010</td>
<td>0.010</td>
<td>0.010</td>
</tr>
<tr>
<td>Tempering Temperature (°C)</td>
<td>820</td>
<td>821</td>
<td>820</td>
<td>829</td>
<td>841</td>
<td>820</td>
<td>820</td>
<td>820</td>
<td>820</td>
<td>820</td>
</tr>
<tr>
<td>UTS (MPa)</td>
<td>899.9</td>
<td>892.3</td>
<td>883.6</td>
<td>878.0</td>
<td>871.8</td>
<td>861.2</td>
<td>851.0</td>
<td>841.3</td>
<td>824.8</td>
<td>810.1</td>
</tr>
<tr>
<td>Cost (US$)</td>
<td>39.08</td>
<td>32.36</td>
<td>31.26</td>
<td>30.65</td>
<td>30.22</td>
<td>28.28</td>
<td>26.78</td>
<td>25.50</td>
<td>23.32</td>
<td>21.63</td>
</tr>
</tbody>
</table>

### Table V

<table>
<thead>
<tr>
<th>Solutions</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>C (wt%)</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
</tr>
<tr>
<td>Mn (wt%)</td>
<td>1.133</td>
<td>1.021</td>
<td>1.031</td>
<td>0.931</td>
<td>1.004</td>
<td>1.035</td>
<td>0.875</td>
<td>0.818</td>
<td>0.653</td>
<td>0.515</td>
</tr>
<tr>
<td>Cr (wt%)</td>
<td>0.285</td>
<td>0.276</td>
<td>0.249</td>
<td>0.244</td>
<td>0.168</td>
<td>0.050</td>
<td>0.143</td>
<td>0.050</td>
<td>0.050</td>
<td>0.050</td>
</tr>
<tr>
<td>Mo (wt%)</td>
<td>0.012</td>
<td>0.018</td>
<td>0.014</td>
<td>0.017</td>
<td>0.012</td>
<td>0.010</td>
<td>0.010</td>
<td>0.010</td>
<td>0.010</td>
<td>0.010</td>
</tr>
<tr>
<td>Tempering Temperature (°C)</td>
<td>942.2</td>
<td>919.9</td>
<td>915.6</td>
<td>900</td>
<td>900</td>
<td>855.2</td>
<td>947.0</td>
<td>900</td>
<td>825.5</td>
<td>832.1</td>
</tr>
<tr>
<td>UTS (MPa)</td>
<td>886.5</td>
<td>882.0</td>
<td>877.2</td>
<td>870.82</td>
<td>862.62</td>
<td>850.9</td>
<td>837.1</td>
<td>820.3</td>
<td>804.3</td>
<td>781.2</td>
</tr>
<tr>
<td>ROA (%)</td>
<td>60.51</td>
<td>61.23</td>
<td>61.35</td>
<td>61.80</td>
<td>61.92</td>
<td>63.38</td>
<td>61.01</td>
<td>62.36</td>
<td>64.54</td>
<td>64.40</td>
</tr>
<tr>
<td>Cost (US$)</td>
<td>40.67</td>
<td>38.36</td>
<td>37.17</td>
<td>35.22</td>
<td>33.07</td>
<td>28.21</td>
<td>30.02</td>
<td>24.66</td>
<td>21.09</td>
<td>18.67</td>
</tr>
</tbody>
</table>
An optimisation experiment has been conducted based on the above objectives where the target values $UTS_{Target} = 900$ (MPa) and $ROA_{Target} = 60$ (%). The result of this experiment is shown in Fig. 7. 10 solutions out of all the obtained Pareto-optimal solutions are selected and listed in Table V.

From the above experiments, it can be seen that, for the optimal design problems that consider both the mechanical properties and the economical factors, MO-nPSO-VP is able to obtain a set of optional solutions (Pareto-optimal solutions), which are close to the predefined UTS and/or ROA targets while providing various levels of heat treatment costs.

![Fig. 7](image)

**Fig. 7.** The performance of the Pareto-optimal solutions for the design problem of $UTS_{Target} = 900$ (MPa), $ROA_{Target} = 60$ (%) and minimising the heat treatment cost with respect to (a) Objective 1, Objective 2 and Objective 3; (b) UTS, ROA and Cost

V. CONCLUSION

In this paper, a new PSO algorithm ‘nPSO-VP’ is proposed, whose population size can be dynamically varied according to the algorithm’s search performance in the optimisation process. The new algorithm was compared with some salient EAs, i.e. CMA-ES, DE and G3+PCX, based on a set of well-known benchmark test problems. The results show that nPSO-VP outperforms the other algorithms in most cases.

The new algorithm was later extended to the multi-objective optimisation case, ‘MO-nPSO-VP’, by applying the RWA method and by preserving the suitable Pareto-optimal solutions in an archive. MO-nPSO-VP was tested using two sets of difficult multi-objective optimisation problems - the ZDT series problems and DTLZ series problems. Preliminary results show that the new algorithm performs as well as and in some cases better than other salient EAs, i.e. PAES, SPEA, SPEA2 and NSGA-II, in terms of both accuracy and diversity.

Finally, the proposed algorithm was also applied to a multi-objective optimal design problem of alloy steels. It aims at determining the optimal heat treatment regime and the required weight percentages for chemical composites to obtain the desired mechanical properties of steel and, at the same time, to minimise the production costs. It is shown that the proposed algorithm is able to obtain the Pareto-optimal solutions for the problem, which are close to the predefined UTS and/or ROA targets while providing various levels of heat treatment costs.

REFERENCES


[15] Q. Zhang and M. Mahfouf, “Mamdani-Type Fuzzy Modelling via Hierarchical Clustering and Multi-Objective Particle Swarm Optimisation (FM-HCPSO)” International Journal of Computational Intelligence Research (IJCIR), Accepted