A novel modified differential evolution algorithm for constrained optimization problems

Dexuan Zou\textsuperscript{a,}\textsuperscript{*}, Haikuan Liu\textsuperscript{a}, Liqun Gao\textsuperscript{b}, Steven Li\textsuperscript{c}

\textsuperscript{a} School of Electrical Engineering and Automation, Xuzhou Normal University, Xuzhou, Jiangsu 221116, PR China
\textsuperscript{b} School of Information Science and Engineering, Northeastern University, Shenyang, Liaoning 110004, PR China
\textsuperscript{c} Division of Business University of South Australia GPO Box 2471, Adelaide, SA 5001, Australia

\begin{abstract}
A novel modified differential evolution algorithm (NMDE) is proposed to solve constrained optimization problems in this paper. The NMDE algorithm modifies scale factor and crossover rate using an adaptive strategy. For any solution, if it is at a standstill, its own scale factor and crossover rate will be adjusted in terms of the information of all successful solutions. We can obtain satisfactory feasible solutions for constrained optimization problems by combining the NMDE algorithm and a common penalty function method. Experimental results show that the proposed algorithm can yield better solutions than those reported in the literature for most problems, and it can be an efficient alternative to solving constrained optimization problems.
\end{abstract}

\section{Introduction}
Constrained optimization problems are very important and frequently appear in many science and engineering disciplines, such as pressure vessel design problem \cite{1,2}, welded beam design problem \cite{3,4}, reliability optimization problems \cite{5,6} and so on. The general constrained optimization problem with inequality, equality, upper bound, and lower bound constraints is defined as Eq. (1).

\begin{equation}
\begin{aligned}
\min (f(x)), x &= (x_1, x_2, \ldots, x_N) \\
\text{s.t.} \quad & g_p(x) \leq 0, \quad p = 1, 2, \ldots, N_g \\
& h_q(x) = 0, \quad q = 1, 2, \ldots, N_h \\
& x_L \leq x_i \leq x_U, \quad i = 1, 2, \ldots, N
\end{aligned}
\end{equation}

where \(x_L\) and \(x_U\) are the lower bound and the upper bound of variable \(x_i\), respectively. \(g_p(x)\) is the \(p\)th inequality constraint, and \(N_g\) is the total number of inequality constraints. \(h_q(x)\) is the \(q\)th equality constraint, and \(N_h\) is the total number of equality constraints.

In order to find satisfactory feasible solutions for constrained optimization problems, researchers had devised a large number of efficient methods: Srivastava and Fahim \cite{7} presented a heuristic approach for minimizing nonlinear mixed discrete–continuous problems with nonlinear mixed discrete–continuous constraints. The approach was an extension of the boundary tracking optimization that was developed by the authors to solve the minimum of nonlinear pure discrete programming problems with pure discrete constraints. Liu et al. \cite{8} proposed a novel hybrid algorithm named PSO-DE,
which integrates particle swarm optimization (PSO) with differential evolution (DE) to solve constrained numerical and engineering optimization problems. Traditional PSO is easy to fall into stagnation when no particle discovers a position that is better than its previous best position for several generations. DE is incorporated to update the previous best positions of particles to force PSO jump out of stagnation, because of its strong searching ability. Due to the utilization of DE, each pbest. communicates and collaborates with its neighbors belonging to pbest in order to improve its performance. PSO-DE allows only half a part of particles to be evolved by PSO. Those particles with higher degree of constraint violation fly throughout the search space according to the information delivered by their pbest and gbest to search for better positions. The hybrid algorithm speeds up the convergence and improves the algorithm’s performance. Ichida [9] described an interval analysis method for finding the global maximum of a multimodal multivariable function subject to equality and/or inequality constraints. By discarding subregions where the global solution cannot exist and applying the interval Newton method to solve the Lagrange equation, one can always find the solution with the rigorous error bound. The feature of this method is that the optimal solution can be obtained with the rigorous error bound without failure. Moreover, as the number of constraints increases, the possibility that unnecessary subregions will be discarded increases. It is important to develop effective algorithms to prevent expansions of interval width. Evolutionary computation techniques have been receiving increasing attention regarding their potential as optimization techniques for complex problems. Based on this consideration, Michalewicz et al. concentrated on constraint handling heuristics for evolutionary computation techniques [10]. Furthermore, this general discussion was followed by three test case studies: truss structure optimization problem, design of a composite laminated plate, and the unit commitment problem. These are typical highly constrained engineering problems and the methods discussed in [10] are directly transferable to industrial engineering problems. Runarsson and Yao [11] proposed a new constraint handling technique, stochastic ranking approach, to balance objective and penalty functions stochastically, and presented a new view on penalty function methods in terms of the dominance of penalty and objective functions. The technique does not introduce any specialized variation operators. It does not require any priori knowledge about a problem since it does not use any penalty coefficient in a penalty function. Stochastic ranking is motivated by their analysis of penalty methods from the point of view of dominance. The balance between the objective and penalty functions is achieved through a ranking procedure based on the stochastic bubble-sort algorithm. Takahama and Sakai [12] proposed the $\alpha$ constrained method to solve constrained optimization problems. This $\alpha$ constrained method adopts a satisfaction level which indicates how well a search point satisfies the constraints. They also made some modifications [13] of the nonlinear simplex method to search around the boundary of the feasible region and to control the convergence speed of the method.

In this paper, we propose a novel modified differential evolution algorithm (NMDE) to solve constrained optimization problems. The proposed algorithm modifies two important parameters of the original differential evolution algorithm, and they are scale factor and crossover rate, respectively. The NMDE adopts an adaptive strategy to adjust scale factor and crossover rate. Specially, each solution has its own scale factor and crossover rate. If stagnation happens to this solution, it will adjust these parameters according to the information of all successful solutions. In addition, we adopt a common penalty function method to balance objective and constraint violations.

The paper is organized as follows. In Section 2, the general procedure of the original differential evolution algorithm (DE) is briefly summarized. In Section 3, a novel modified differential evolution algorithm (NMDE) is proposed, and the procedure of the NMDE is adequately described. In Section 5, a large number of experiments are carried out to test the performance of the NMDE on solving constrained optimization problems. In Section 6, we end this paper with some conclusions and comments for further research.

2. The original differential evolution algorithm (DE)

As an advanced computing technique, differential evolution algorithm [14] was first proposed by Price and Storn in 1995. This technique includes three important operations: mutation, crossover and selection, and it utilizes the three operators to evolve from randomly generated initial population to final individual solution. Mutation and crossover are used to generate new vectors (trial vectors), and selection is then used to determine whether the new generated vectors can survive the next generation. In short, the procedure of the DE works as follows:

Step 1: Initial algorithm parameters.
They are: scale factor $F$, crossover rate $CR$, the population size $M$ and the maximum number of iterations $K$.

Step 2: Randomly generate $M$ candidate solutions.
The initial candidate solutions are generated from a uniform distribution in the ranges $[x_{iL}, x_{iU}]$ ($j = 1, 2, \ldots, N$), where, $N$ is the number of variables.

Step 3: Mutation.
The mutation operator can not only increase the diversity of solution vectors, but also enhance the exploration capability of solution space for the DE algorithm. For each parent, $x_i^k$ ($i = 1, 2, \ldots, M$) of generation $k$ ($k = 1, 2, \ldots, K$), a trial vector, $u_i^{k+1}$ is created by mutating a target vector. According to the mutation operator, the trial vector is then calculated using one of the following equations:

$$ u_i^{k+1} = x_i^k + F(x_i^k - x_j^k), $$

(2)
Here, \( F \), \( F_1 \), and \( F_2 \) are scale factors, and they are used to control the amplification of the differential variation between two individuals so as to avoid search stagnation. \( x_{i,1}^k \) represents the best solution vector at generation \( k \), \( i_1, i_2, i_3, i_4 \) and \( i_5 \) are different integers, randomly selected from the set \( \{1, 2, \ldots, M\} \). Here the choice of Eqs. (2)–(6) leads to different variants of DE, such as DE/rand/1/bin, DE/best/1/bin, DE/current-to-best/1/bin, DE/best/2/bin, and DE/rand/2/bin, respectively.

Step 4: Crossover. DE follows a discrete recombination approach where elements from the parent vector \( x_i^k \), are combined with elements from the trial vector \( u_i^k \), to produce the offspring \( u_i^k \).

\[
u_{i,j}^{k+1} = \begin{cases} u_{i,j}^{k+1}, & \text{if } \text{rand} < \text{CR} \text{ or } j = r_{1-D}; \\ x_{i,j}^k, & \text{otherwise}, \end{cases}
\]

where, \( r_{1-D} \) is a random integer in \( [1, D] \). CR represents crossover rate, and \( CR \in [0, 1] \).

Step 5: Selection. The generated offspring \( u_{i,j}^{k+1} \) replaces the parent \( x_i^k \), only if the fitness of the offspring is better than that of the parent.

Step 6: Check the stopping criterion.

If the stopping criterion (maximum number of iterations \( K \)) is satisfied, computation is terminated. Otherwise, Steps 3–5 are repeated.

3. A novel modified differential evolution algorithm (NMDE)

The recent development of intelligent optimization algorithms has provided more alternatives for finding the optimal solutions of constrained optimization problems, such as: genetic algorithm [15], particle swarm optimization algorithm [16], differential evolution algorithm [14] and so on. DE is a competitive and potential algorithm compared to the other intelligent optimization algorithms, and the excellent performance of the DE has drawn much attention from researchers.

In order to apply the DE algorithm to more complex optimization problems, a further performance improvement is necessary. In this paper, we propose an efficient algorithm named novel modified differential evolution algorithm (NMDE) to get feasible solutions of high quality for constrained optimization problems. In this study, we adopt the DE/rand/1/bin scheme (Eq. (2)). In the DE algorithm, scale factor \( F \) and crossover rate are set to fixed values for all solutions. However, this parameter setting lacks self-adaptation, which may lead to the convergence premature of the DE algorithm. Each solution has its own characteristic, some may be improved step by step, but some may be at a standstill. To enable all solutions to get rid of stagnation easily, an adaptive strategy of modifying scale factor and crossover rate is devised. The new strategy is described as follows:

A predominant advantage of the NMDE algorithm is that each solution has its own scale factor and crossover rate. Specially, for the \( i \)-th \((i = 1, 2, \ldots, M) \) solution, its own scale factor \( F_i \) and crossover rate \( CR_i \) are adaptively adapted to a suitable range in terms of the corresponding values of all successful solutions. With regard to the \( F_i \) (\( CR_i \)) value of the \( i \)-th solution, we assume that it is uniformly distributed in the range of \([F_m - \delta_F, F_m + \delta_F]([CR_m - \delta_{CR}, CR_m + \delta_{CR})\) with mean \( F_m \) (\( CR_m \)) and standard deviation \( \delta_F/\sqrt{3} \) (\( \delta_{CR}/\sqrt{3} \)). Initially, \( F_1 \) (\( CR_1 \)) value is uniformly generated in the range of \([0,2][0,1])\). If the \( i \)-th solution is continuously at a standstill during a fixed number of iterations \( SP \) (Stagnation period), \( F_i \) (\( CR_i \)) value will be uniformly regenerated in the range of \([F_m - \delta_F, F_m + \delta_F]([CR_m - \delta_{CR}, CR_m + \delta_{CR})\). Especially, \( \delta_F \) (\( \delta_{CR} \)) value is predetermined, and \( F_m \) (\( CR_m \)) is calculated as follows:

\[
F_m = \frac{\sum_{F \in S_F} F}{N_{S_F}}.
\]

\[
CR_m = \frac{\sum_{CR \in S_{CR}} CR}{N_{S_{CR}}}.
\]

Here, \( S_F \) represents the set of all successful scale factors at current iteration, and \( S_{CR} \) represents the set of all successful crossover rates at current iteration. \( N_{S_F} \) represents the number of successful scale factors at current iteration, and \( N_{S_{CR}} \) represents the number of successful crossover rates at current iteration. Eqs. (8) and (9) are the computational formulas of arithmetic means of successful scale factors and crossover rates. It should be noticed that the \( F_i \) (\( CR_i \)) value using uniform
regulated formulas after Eqs. (8) and (9) are described as follows:

\[
F_m = \begin{cases} 
2, & \text{if } F_m > 2; \\
0.2, & \text{if } F_m < 0; \\
F_m, & \text{otherwise.}
\end{cases} \quad (10)
\]

\[
CR_m = \begin{cases} 
1, & \text{if } CR_m > 1; \\
0.1, & \text{if } CR_m < 0; \\
CR_m, & \text{otherwise.}
\end{cases} \quad (11)
\]

In case no successful scale factors and crossover rates are found at current iteration, \( F^i \) and \( CR^i \) are uniformly generated in the ranges of \([0, 2]\) and \([0, 1]\), respectively. It should be emphasized that the adaptive strategy using uniform distribution happens if and only if a solution is at a standstill judged by stagnation period \( SP \). Above all, the procedure of the NMDE works as Table 1:

Here, \( U(\cdot) \) represents uniform distribution. \( \text{flag}^i \) is used to record the number of continuous stagnations for the \( i \)th \( (i = 1, 2, \ldots, M) \) solution. If \( \text{flag}^i \) reaches the stagnation period \( SP \), \( F^i \) and \( CR^i \) will be adjusted using uniform distribution.

4. The constraint handling technique

It is well known that penalty function method is an effective constraint handling technique, and it can guide unfeasible solutions to move to feasible regions. In this paper, we employ a common penalty function method to handle constrained

---

Table 1

<table>
<thead>
<tr>
<th>Line</th>
<th>Procedure of NMDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Begin</td>
</tr>
<tr>
<td>2</td>
<td>Set ( \delta_S = 0.2; \delta_{CR} = 0.1; SP = 50; \text{flag} = 0 ) for the ( i )th ( (i = 1, 2, \ldots, M) ) solution</td>
</tr>
<tr>
<td>3</td>
<td>Uniformly generate ( F^i ) value in the range of ([0,2][0,1]) for the ( i )th ( (i = 1, 2, \ldots, M) ) solution</td>
</tr>
<tr>
<td>4</td>
<td>Initialize a random population ( A )</td>
</tr>
<tr>
<td>5</td>
<td>For ( k = 1 ) to ( K )</td>
</tr>
<tr>
<td>6</td>
<td>( p_S = \emptyset; p_{CR} = \emptyset )</td>
</tr>
<tr>
<td>7</td>
<td>For ( i = 1 ) to ( M )</td>
</tr>
<tr>
<td>8</td>
<td>Randomly generate three integers ( i_1, i_2 ) and ( i_3 ) in the range ([1, M]), and ( i_1 \neq i_2 \neq i_3 \neq i ).</td>
</tr>
<tr>
<td>9</td>
<td>( u_{i}^{k+1} = u_{i}^{k} + F^i \times (x_{i}^{k} - x_{i}^{k}) )</td>
</tr>
<tr>
<td>10</td>
<td>Randomly generate an integer ( j_{rand} ) in the range ([1, N])</td>
</tr>
<tr>
<td>11</td>
<td>For ( j = 1 ) to ( N )</td>
</tr>
<tr>
<td>12</td>
<td>If ( \text{rand} &lt; CR^j ) or ( j = j_{rand} )</td>
</tr>
<tr>
<td>13</td>
<td>( u_{i,j}^{k+1} = u_{i,j}^{k} )</td>
</tr>
<tr>
<td>14</td>
<td>Else</td>
</tr>
<tr>
<td>15</td>
<td>( u_{i,j}^{k+1} = x_{i,j} )</td>
</tr>
<tr>
<td>16</td>
<td>End If</td>
</tr>
<tr>
<td>17</td>
<td>End For</td>
</tr>
<tr>
<td>18</td>
<td>If ( f(u_{i,j}^{k+1}) &lt; f(x_{i,j}) )</td>
</tr>
<tr>
<td>19</td>
<td>( x_{i,j}^{k+1} = u_{i,j}^{k+1}; F^i \rightarrow S_F; CR^i \rightarrow S_{CR}; \text{flag} = 0 )</td>
</tr>
<tr>
<td>20</td>
<td>Else</td>
</tr>
<tr>
<td>21</td>
<td>( x_{i,j}^{k+1} = x_{i,j}; \text{flag} = \text{flag} + 1 )</td>
</tr>
<tr>
<td>22</td>
<td>End If</td>
</tr>
<tr>
<td>23</td>
<td>End For</td>
</tr>
<tr>
<td>24</td>
<td>For ( i = 1 ) to ( M )</td>
</tr>
<tr>
<td>25</td>
<td>If ( \text{flag}^i = SP )</td>
</tr>
<tr>
<td>26</td>
<td>( F_m = \frac{\sum F_m}{n_F}; CR_m = \frac{\sum CR_m}{n_{CR}} )</td>
</tr>
<tr>
<td>27</td>
<td>( F^i \leftarrow U(F_m - \delta_S, F_m + \delta_S); CR^i \leftarrow U(CR_m - \delta_{CR}, CR_m + \delta_{CR}) )</td>
</tr>
<tr>
<td>28</td>
<td>Judge whether ( F^i ) (( CR^i )) has gotten out of the range ([0, 2]) (([0, 1])), and regulate it if necessary</td>
</tr>
<tr>
<td>29</td>
<td>Else</td>
</tr>
<tr>
<td>30</td>
<td>( F^i \leftarrow U(0, 2); CR^i \leftarrow U(0, 1) )</td>
</tr>
<tr>
<td>31</td>
<td>End If</td>
</tr>
<tr>
<td>32</td>
<td>( \text{flag}^i = 0 )</td>
</tr>
<tr>
<td>33</td>
<td>End If</td>
</tr>
<tr>
<td>34</td>
<td>End For</td>
</tr>
<tr>
<td>35</td>
<td>End For</td>
</tr>
<tr>
<td>36</td>
<td>End For</td>
</tr>
<tr>
<td>37</td>
<td>End</td>
</tr>
</tbody>
</table>
optimization problems, and it is described as follows:

\[
\min F(x) = f(x) + \lambda \left[ \sum_{p=1}^{N_p} \max\{0, g_p(x)\}^2 + \sum_{q=1}^{N_q} \max\{0, |h_q(x)| - \varepsilon\}^2 \right].
\]  

(12)

Here, \( F(x) \) represents penalty function, \( f(x) \) represents objective function. \( g_p(x), (p = 1, 2, \ldots, N_p) \) represents the \( p \)th inequality constraint, and \( h_q(x), (q = 1, 2, \ldots, N_q) \) represents the \( q \)th equality constraint. \( N_p \) is the number of inequality constraints, and \( N_q \) is the number of equality constraints. It should be noticed that Eq. (12) only meets the case that the problems to be solved are minimization problems. For maximization problem \( f(x) \), a useful method is to replace \( f(x) \) with \(-f(x)\) in Eq. (12). Unfortunately, we encounter an intractability, and it is hard to find feasible solutions that satisfy the equality constraint \( h_q(x) = 0 \) exactly, so it is necessary to convert it into the inequality constraint \(|h_q(x)| - \varepsilon \leq 0\). Here, \( \varepsilon \) is a small positive constant, and it represents the tolerated violation. In addition, \( \lambda \) is a large positive constant which imposes penalty on unfeasible solutions, and it is defined as penalty coefficient.

5. Experimental results and analysis

In this section, three differential evolution algorithms are selected to solve fourteen constrained optimization problems, and these algorithms include the differential evolution algorithm based on self-adapting control parameters (SADE) [17], opposition-based differential evolution (ODE) [18] and novel modified differential evolution algorithm (NMDE). For the SADE and ODE algorithms, the parameters were fixed as those in the original papers. Especially, for the SADE, probability \( \tau_1 = \tau_2 = 0.1, F_1 = 0.1, F_0 = 0.9, \) population size \( M = 40 \). For the ODE, jumping rate \( J = 0.3, \) scale factor \( F = 0.5, \) crossover rate \( CR = 0.9, \) population size \( M = 40. \) On the other hand, the parameters of the NMDE is set as follows: \( \delta_F = 0.2, \delta_{CR} = 0.1, SP = 50, \) population size \( M = 40. \) The numbers of iterations performed are set to 1000 for problems \( f_1 - f_{14}. \) Penalty coefficient \( \lambda \) is set to \( 10^{30} \) to punish constraint violations. In the experiments, each constrained optimization problem is run for 30 independent replications. The best, median, mean, and worst objective function values over these 30 replications are reported in Table 2.

From Table 2, it can be observed that the NMDE outperforms the other two DE algorithms on solving problems \( f_5, f_6 \) and \( f_{13}. \) The best objective function values obtained using the NMDE are 680.6300573776, 24.30622005 and 7049.24808782, respectively, for problems \( f_5, f_6 \) and \( f_{13}. \) The SADE, ODE and NMDE can find the same optimal values for the other problems. In addition, the NMDE obtains better mean values than those obtained by the SADE and the ODE on problems \( f_6, f_7, f_8, f_{12}, f_{13} \) and \( f_{14}. \) The SADE obtains better mean values on problems \( f_5 \) and \( f_{11}. \) The NMDE and the SADE obtain the same mean value for problem \( f_4, \) and this value is better than the one obtained by the ODE. The SADE and the ODE obtain the same mean value for problem \( f_{10}, \) and this value is better than the one obtained by the NMDE. With regard to the other problems, almost no significant difference was observed on the compared algorithms. Above all, all the three DE algorithms can produce the same optimal values for most problems, and the NMDE has demonstrated stronger convergence on solving six problems according to the term “mean”. Compared with the other two DE algorithms, the NMDE algorithm is found to be better than, or at least comparable to them considering the quality of the solution obtained.

In fact, all the 14 constrained optimization problems in Table 2 had been solved previously, and the detailed information of these problems can be described as follows:

The first constrained problem was originally introduced by Bracken and McCormick [19], and the problem can be stated as:

\[
\min f_1(x) = (x_1 - 2)^2 + (x_2 - 1)^2
\]

s.t. \( g_1(x) = x_1^2/4 + x_2^2 - 1 \leq 0 \)

\( h_1(x) = x_1 - 2x_2 + 1 = 0 \)

\( -10 \leq x_1, x_2 \leq 10. \)  

(13)

Homaifar et al. [20] and Fogel [21] had solve this problem previously, and the optimal values obtained by them are 1.4339 and 1.377192, respectively. Table 3 shows the optimum using the NMDE algorithm and also provides the results obtained by Homaifar et al. [20] and Fogel [21]. The optimal solution found using the NMDE algorithm is \( x^* = (0.8228756624, 0.9114378262), \) and the corresponding objective function value is \( f(x^*) = 1.39346496, \) It is clear that the optimal value found using the NMDE algorithm is better than the one obtained by Homaifar et al. [20]. On the other hand, it seems that the NMDE value is worst than the one produced by Fogel [21]. However, the tolerated violation of the NMDE is more stringent than his tolerated violation.

The second problem is a minimization problem with two design variables and two inequality constraints [22].

\[
\min f_2(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2
\]

s.t. \( g_1(x) = (x_1 - 0.05)^2 + (x_2 - 2.5)^2 - 4.84 \leq 0 \)

\( g_2(x) = -x_1^2 - (x_2 - 2.5)^2 + 4.84 \leq 0 \)

\( 0 \leq x_1, x_2 \leq 6. \)  

(14)
Table 2
The optimization results of 14 test problems obtained by three differential evolution algorithms.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Algorithm</th>
<th>Best</th>
<th>Median</th>
<th>Mean</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_1</td>
<td>SADE</td>
<td>703.946947</td>
<td>703.946947</td>
<td>703.946947</td>
<td>703.946947</td>
</tr>
<tr>
<td></td>
<td>ODE</td>
<td>703.946947</td>
<td>703.946947</td>
<td>703.946947</td>
<td>703.946947</td>
</tr>
<tr>
<td></td>
<td>NMDE</td>
<td>703.946947</td>
<td>703.946947</td>
<td>703.946947</td>
<td>703.946947</td>
</tr>
<tr>
<td>f_2</td>
<td>SADE</td>
<td>13.59084169</td>
<td>13.59084169</td>
<td>13.59084169</td>
<td>13.59084169</td>
</tr>
</tbody>
</table>

Table 3
Optimal results of the first constrained problem.

<table>
<thead>
<tr>
<th>x^∗ and f(x^∗)</th>
<th>Homaifar et al. [20]</th>
<th>Fogel [21]</th>
<th>Zou</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_1^*</td>
<td>0.8080</td>
<td>0.834963</td>
<td>0.8228756624</td>
</tr>
<tr>
<td>x_2^*</td>
<td>0.88544</td>
<td>0.912514</td>
<td>0.9114378262</td>
</tr>
<tr>
<td>g_1</td>
<td>–0.052</td>
<td>–0.006973</td>
<td>–0.2926e–010</td>
</tr>
<tr>
<td></td>
<td>h_1</td>
<td></td>
<td>3.7e–002</td>
</tr>
<tr>
<td>f(x^∗)</td>
<td>1.4339</td>
<td>1.377192</td>
<td>1.3934649468</td>
</tr>
</tbody>
</table>

The optimum solution is x^∗ = (2.246826, 2.381865) where f(x^∗) = 13.59085. Deb [22] solved this problem by using the hybrid GA-based method, which was combined by tournament selection (TS-R method) and Powell and Skolnick’s constraint handling technique. The best objective function value using the hybrid GA-based method is equal to 13.59085, which seems to be the same as the optimal value. Lee and Geem [23] employed harmony search algorithm (HS) to solve this problem, and the best objective function value obtained by them was equal to 13.590845. The NMDE algorithm is also used to solve this problem, and the best solution using the NMDE was x^∗ = (2.2468258369, 2.3818634593) with a corresponding function value of f(x^∗) = 13.5908416924. A closer look at these results in Table 4 shows that the NMDE solution is better than those found by Deb [22] and Lee and Geem [23].

The third problem is a minimization problem with five design variables and six inequality constraints, and this problem can be stated as:
and this constrained problem can be stated as:

\[
\begin{align*}
\text{min} f_3(x) &= 5.3578547x_1^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \\
s.t. \\
g_1(x) &= 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0 \\
g_2(x) &= -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0 \\
g_3(x) &= 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_1^2 - 110 \leq 0 \\
g_4(x) &= -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_1^2 + 90 \leq 0 \\
g_5(x) &= 9.300961 + 0.0047026x_2x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0 \\
g_6(x) &= -9.300961 - 0.0047026x_2x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0
\end{align*}
\]

where 78 \leq x_1 \leq 102, 33 \leq x_2 \leq 45 and 27 \leq x_i \leq 45 (i = 3, 4, 5). The optimum solution is \(x^* = (78, 33, 29.9953, 45, 36.7758)\) where the optimal objective function value \(f(x^*) = -30665.539\). Two constraints are active \((g_1\) and \(g_6)\). Lee and Geem [23] used the HS algorithm to yield a best objective function value of \(f(x^*) = -30665.500\). On the other hand, the NMDE algorithm is also used to solve this problem. The best solution found by the NMDE is \(x^* = (78, 33, 29.9952560257, 45, 36.7758129058)\), and its corresponding objective function value is equal to -30665.53867178 which is a little better than the value obtained by Lee and Geem [23] (as Table 5).

The fourth constrained problem is a maximization problem with three design variables and one inequality constraint, and this constrained problem can be stated as:

\[
\begin{align*}
\text{max} f_4(x) &= (100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100 \\
s.t. g(x) &= (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0
\end{align*}
\]

where 0 \leq x_i \leq 10 (i = 1, 2, 3) and \(p, q, r = 1, \ldots, 9\). The feasible region of the search space consists of 9³ disjointed spheres. A point \((x_1, x_2, x_3)\) is feasible if and only if there exist \(p, q, r\) such that the above inequality holds. The optimum is located at \(x^* = (5, 5, 5)\) where \(f(x^*) = 1\). The solution lies within the feasible region.

Coello [24] used a GA-based method to solve problem \(f_4\). Koziel and Michalewicz [25] used a GA variant to solve this problem, additionally, Mahdavi et al. [26] used the IHS algorithm to solve the same problem. Table 6 compares the best solution obtained using the previous best solutions reported by Mahdavi et al. [26], Coello [24], and Koziel and Michalewicz [25]. According to Table 6, the best objective function value obtained by our method is better than those obtained by Mahdavi et al. [26] and Koziel and Michalewicz [25], and it is comparable to the result obtained by Coello [24].

The fifth problem can be stated as Eq. (17).

\[
\begin{align*}
\text{min} f_5(x) &= (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \\
s.t. \\
g_1(x) &= -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0 \\
g_2(x) &= -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0 \\
g_3(x) &= -196 + 23x_1 + x_2^2 + 6x_3 - 8x_7 \leq 0 \\
g_4(x) &= 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0
\end{align*}
\]
where \(-10 \leq x_i \leq 10 \quad (i = 1, \ldots, 7)\). The optimum solution is \(x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)\) where the optimal objective function value \(f(x^*) = 680.6300573\). Two constraints are active (\(g_1\) and \(g_4\)). Table 7 presents the best solution found using the NMDE algorithm for the fifth problem, and compares the NMDE result with the previous best solutions reported in the literature. Michalewicz [27] obtained a best objective function value of \(f(x^*) = 680.6462\). Deb [22] obtained a best objective function value of \(f(x^*) = 680.6413574\). The NMDE is also used to solve this problem, and the optimal solution is obtained at \(x^* = (2.330499, 1.95137236, -0.4775413141, 4.3657262536, -0.6244869364, 1.0381310065, 1.5942266781)\) with a corresponding function value equal to \(f(x^*) = 680.6300573776\). The NMDE solution is better than the above results.

The sixth constrained problem is a minimization problem with ten design variables and eight inequality constraints, and this problem can be stated as Eq. (18).

\[
\begin{align*}
\text{min } f_6(x) &= x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 \\
&\quad + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45
\end{align*}
\]

s.t.
\[
\begin{align*}
&g_1(x) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_9 \leq 0 \\
&g_2(x) = 10x_1 - 8x_2 - 17x_9 + 2x_8 \leq 0 \\
&g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0 \\
&g_4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0 \\
&g_5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0 \\
&g_6(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_3 - 6x_6 \leq 0 \\
&g_7(x) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_3^2 - x_6 - 30 \leq 0 \\
&g_8(x) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0 \\
&-10 \leq x_i \leq 10 \quad (i = 1, \ldots, 10).
\end{align*}
\]

The optimum solution is \(x^* = (2.171996, 2.363683, 8.773962, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828762, 8.280092, 8.375927)\) where \(f(x^*) = 24.3062091\). Six constraints are active (\(g_1, g_2, g_3, g_4, g_5\) and \(g_6\)). Table 8 lists the best solution of the sixth constrained problem obtained by the NMDE algorithm, and compares the NMDE solution with the previous best solutions reported by Michalewicz [27], Deb [22] and Lee and Geem [23]. The best solution found by Michalewicz [27] had an objective function value equal to \(f(x^*) = 24.690\). Deb [22] solved this constrained problem, and obtained a best objective function value equal to \(f(x^*) = 24.37248\). Lee and Geem [23] used the HS algorithm to solve this problem, and the best objective function value obtained using the HS is equal to \(f(x^*) = 24.3667946\). In addition, the NMDE algorithm is used to solve the same problem. The best objective function value obtained using the NMDE algorithm is \(f(x^*) = 24.30622005\) which is better than those obtained by Michalewicz [27], Deb [22] and Lee and Geem [23].

The seventh constrained problem is minimization of the weight of the spring. It consists of minimizing the weight of a tension/compression spring subject to constraints on shear stress, surge frequency and minimum deflection as shown in Fig. 1.
Table 8
Optimal results of the sixth constrained problem.

<table>
<thead>
<tr>
<th>x^* and f (x^*)</th>
<th>Michalewicz [27]</th>
<th>Deb [22]</th>
<th>Lee and Geem [23]</th>
<th>Zou</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_1^*</td>
<td>Unavailable</td>
<td>Unavailable</td>
<td>2.155225</td>
<td>2.1720512173</td>
</tr>
<tr>
<td>x_2^*</td>
<td>2.407687</td>
<td>2.3635260785</td>
<td>8.7738518481</td>
<td></td>
</tr>
<tr>
<td>x_3^*</td>
<td>8.778069</td>
<td>5.0557491689</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_4^*</td>
<td>5.102078</td>
<td>0.9909029900</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_5^*</td>
<td>1.357685</td>
<td>1.4312257120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_6^*</td>
<td>1.287760</td>
<td>1.3217623165</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_7^*</td>
<td>9.800438</td>
<td>9.8288278401</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_8^*</td>
<td>8.187803</td>
<td>8.2808428341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_9^*</td>
<td>8.256297</td>
<td>8.3774285038</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_10^*</td>
<td>0.967625</td>
<td>0.9909029900</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g_1</td>
<td>Unavailable</td>
<td>-0.00000113</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g_2</td>
<td>-0.00000016</td>
<td>-0.00000042</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g_3</td>
<td>-0.0000042</td>
<td>-0.000000960</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g_4</td>
<td>-0.00000318</td>
<td>-0.00000289</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g_5</td>
<td>-6.14697221</td>
<td>-50.03052434</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g_6</td>
<td>-5.108x_1^2</td>
<td>-1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>g_7</td>
<td>1 - x_1^2</td>
<td>0</td>
<td>0.01266523</td>
<td></td>
</tr>
<tr>
<td>g_8</td>
<td>x_1 + x_2</td>
<td>1.5</td>
<td>0.01266523</td>
<td></td>
</tr>
<tr>
<td>f (x^*)</td>
<td>24.690</td>
<td>24.37248</td>
<td>24.3667946</td>
<td>24.30622005</td>
</tr>
</tbody>
</table>

Fig. 1. Tension/compression spring.

The design variables are the mean coil diameter $D(=x_1)$; the wire diameter $d(=x_2)$ and the number of active coils $N(=x_3)$. The problem formulation can be stated as:

$$\begin{align*}
\text{min } f_7(x) &= (x_3 + 2)x_2x_1^2 \\
\text{s.t. } g_1(x) &= 1 - \frac{x_3^2 x_1}{71785x_1^4} \leq 0 \\
g_2(x) &= \frac{4x_2^3 - x_1 x_2}{12566(x_2 x_1^2 - x_1^3)} + \frac{1}{5108x_1^2} - 1 \leq 0 \\
g_3(x) &= 1 - \frac{x_1}{x_2 x_3} \leq 0 \\
g_4(x) &= \frac{x_1 + x_2}{1.5} - 1 \leq 0.
\end{align*}$$ (19)

Belegundu [29] used eight different mathematical optimization techniques to solve this problem. Arora [28] used a numerical optimization technique called constraint correction at constant cost to solve this problem. Coello [24] used a GA-based method to solve this problem. In addition to previous research, the application of the NMDE algorithm to the minimization of the weight of the spring is also important. The NMDE obtains the best solution at $x^* = (0.05168928572142, 0.35672314403848, 11.28864892515349)$, and its corresponding best objective function value is $f(x^*) = 0.01266523$. Table 9 presents the best solution of this problem obtained using the NMDE algorithm and compares the NMDE results with solutions reported by other researchers. It is obvious from Table 9 that the best solution obtained using the NMDE algorithm is better than those reported previously in the literature.

The eighth constrained problem is pressure vessel design. A cylindrical vessel is capped at both ends by hemispherical heads as shown in Fig. 2. The objective is to minimize the total cost, including the cost of material, forming and welding. There are four design variables: $T_s$ (thickness of the shell, $x_1$), $T_h$ (thickness of the head, $x_2$), $R$ (inner radius, $x_3$) and $L$ (length of cylindrical section of the vessel, not including the head, $x_4$). $T_s$ and $T_h$ are integer multiples of 0.0625 inch, which are the available thicknesses of rolled steel plates, and $R$ and $L$ are continuous. The problem formulation can be stated as follows:
Deb and Gene [30] solved this problem by using a genetic adaptive search technique, and Coello [31] solved this problem by using a GA-based method. On the other hand, Mahdavi et al. [26] used an improved harmony search algorithm (IHS) to solve the same problem. They found the best solution at $x^* = (0.75, 0.375, 38.86010, 221.36553)$, and its corresponding objective function value was reported to be $f(x^*) = 5849.76169$. However, this value was wrongly calculated, and it is approximately equal to $f(x^*) = 5850.38363$ in terms of the above best solution. In addition, the NMDE algorithm is also used to solve the problem of pressure vessel design. The best NMDE solution is at $x^* = (0.75, 0.375, 38.860103626943, 221.365471356008)$, and its corresponding objective function value is equal to $f(x^*) = 5850.38306033$. Table 10 lists the comparisons of results, and it is clear that the result obtained using the NMDE algorithm is better than those reported previously in the literature.

Problem $f_8$ has another variation, and the variation is defined as problem $f_9$ in this paper. Problem $f_9$ has two extra inequalities, and they are $g_5(x) = 1.1 - x_1 \leq 0$ and $g_6(x) = 0.6 - x_2 \leq 0$, respectively. The methods applied to this problem include a GA-based approach [2] and the HS algorithm [23]. In addition, the best solution was obtained at $x^* = (1.125, 0.625, 58.29015, 43.69268)$ by Mahdavi et al. [26] using the HS algorithm, and its corresponding best value was reported to be $f(x^*) = 7197.730$. However, this value was wrongly calculated, and it is approximately equal to $f(x^*) = 7198.0054775$ according to the solution reported by Mahdavi et al. [26]. On the other hand, the constraints obtained using the HS are $g_1 = -0.000000105$, $g_2 = -0.068911969$, $g_3 = 0.065715899$ and $g_4 = -196.307$, respectively. Strictly speaking, the best solution obtained by Mahdavi et al. [26] is not a feasible solution, because $g_4$ does not satisfy the constraint. In addition to the above methods, the NMDE is also applied to solve problem $f_9$. The best solution obtained using the NMDE is $x^* = (1.125, 0.625, 58.29015544041451, 43.69265623882458)$, and its corresponding objective function value is equal to $f(x^*) = 7198.00542037$. Table 11 shows the best solution from the NMDE algorithm and also provides the results reported previously in the literature. The NMDE algorithm shows better results than the other methods.

The tenth constrained problem is welded beam design. The welded beam structure (as in Fig. 3) is a practical design problem that has been often used as a benchmark for testing different optimization methods. The objective is to find the minimum fabricating cost of the welded beam subject to constraints on shear stress ($\tau$), bending stress ($\sigma$), buckling load ($P_c$), end deflection ($\delta$), and side constraint. There are four design variables: $l(t=-x_1)$, $l(t=-x_2)$, $t(t=-x_3)$ and $b(t=-x_4)$. The mathematical formulation of the objective function $f(x)$, which is the total fabricating cost mainly comprised of the setup,
Table 11
Optimal results for pressure vessel design (six inequalities).

<table>
<thead>
<tr>
<th>$x^<em>$ and $f(x^</em>)$</th>
<th>Mahdavi et al. [26]</th>
<th>Wu and Chow [2]</th>
<th>Lee and Geem [23]</th>
<th>Zou</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1^*$</td>
<td>1.125</td>
<td>1.125</td>
<td>1.125</td>
<td>1.125</td>
</tr>
<tr>
<td>$x_2^*$</td>
<td>0.625</td>
<td>0.625</td>
<td>0.625</td>
<td>0.625</td>
</tr>
<tr>
<td>$x_3^*$</td>
<td>58.29015</td>
<td>58.1978</td>
<td>58.2789</td>
<td>58.29015544041451</td>
</tr>
<tr>
<td>$x_4^*$</td>
<td>43.69268</td>
<td>44.2930</td>
<td>43.7549</td>
<td>43.69265623882458</td>
</tr>
<tr>
<td>$g_1(x^*)$</td>
<td>-0.000000105</td>
<td>-0.00178</td>
<td>-0.00022</td>
<td>0</td>
</tr>
<tr>
<td>$g_2(x^*)$</td>
<td>-0.068911969</td>
<td>-0.06979</td>
<td>-0.06902</td>
<td>-0.06891192</td>
</tr>
<tr>
<td>$g_3(x^*)$</td>
<td>0.065715899</td>
<td>-974.3</td>
<td>-3.71629</td>
<td>0</td>
</tr>
<tr>
<td>$g_4(x^*)$</td>
<td>-196.307</td>
<td>-195.707</td>
<td>-196.245</td>
<td>-196.30734376</td>
</tr>
<tr>
<td>$g_5(x^*)$</td>
<td>-0.025</td>
<td>-0.025</td>
<td>-0.025</td>
<td>-0.025</td>
</tr>
<tr>
<td>$g_6(x^*)$</td>
<td>-0.025</td>
<td>-0.025</td>
<td>-0.025</td>
<td>-0.025</td>
</tr>
<tr>
<td>$f(x^*)$</td>
<td>7198.0054775</td>
<td>7207.494</td>
<td>7198.433</td>
<td>7198.00542037</td>
</tr>
</tbody>
</table>

Fig. 2. Schematic of the pressure vessel.

Fig. 3. Welded beam structure.

welding labor, and material costs, is as follows:

$$
\text{min}_{x} f_{10}(x) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2)
$$

s.t.

$$
\begin{align*}
g_1(x) &= \tau(x) - 13600 \leq 0 \\
g_2(x) &= \sigma(x) - 30000 \leq 0 \\
g_3(x) &= x_1 - x_4 \leq 0 \\
g_4(x) &= \delta(x) - 0.25 \leq 0 \\
g_5(x) &= 6000 - P_t(x) \leq 0 \\
0.125 \leq x_1 &\leq 10, \ 0.1 \leq x_2, x_3, x_4 \leq 10.
\end{align*}
$$

(21)

The terms $\tau(x)$, $\sigma(x)$, $P_t(x)$, and $\delta(x)$ are given below:

$$
\tau(x) = \sqrt{(\tau'(x))^2 + (\tau''(x))^2 + x_2 \tau'(x) \tau''(x)} / \sqrt{0.25(x_2^2 + (x_1 + x_3)^2)},
$$

$$
\sigma(x) = \frac{504000}{x_3^3 x_4}, \quad P_t(x) = 64746.022(1 - 0.0282346x_3)x_3 x_4^3, \quad \delta(x) = \frac{2.1952}{x_3^2 x_4}
$$

where

$$
\tau'(x) = \frac{6000}{\sqrt{2}x_1 x_2}, \quad \tau''(x) = \frac{6000(14 + 0.5x_3)}{2(0.707x_1 x_2(x_2^2/12 + 0.25(x_1 + x_3)^2))}.
$$
Table 12

Optimal results for the welded beam design.

<table>
<thead>
<tr>
<th>$x^<em>$ and $f(x^</em>)$</th>
<th>Lee and Geem [23]</th>
<th>Deb [22]</th>
<th>Deb [4]</th>
<th>Zou</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1^*$</td>
<td>0.2442</td>
<td>Unavailable</td>
<td>0.2489</td>
<td>0.24500538884006</td>
</tr>
<tr>
<td>$x_2^*$</td>
<td>6.2231</td>
<td>6.1730</td>
<td>6.28451091444737</td>
<td></td>
</tr>
<tr>
<td>$x_3^*$</td>
<td>8.2915</td>
<td>8.1789</td>
<td>8.19911044972404</td>
<td></td>
</tr>
<tr>
<td>$x_4^*$</td>
<td>0.2443</td>
<td>0.2533</td>
<td>0.24500538886305</td>
<td></td>
</tr>
<tr>
<td>$g_1(x^*)$</td>
<td>Unavailable</td>
<td>Unavailable</td>
<td>-0.00000046</td>
<td></td>
</tr>
<tr>
<td>$g_2(x^*)$</td>
<td></td>
<td></td>
<td>-0.000002621</td>
<td></td>
</tr>
<tr>
<td>$g_3(x^*)$</td>
<td></td>
<td></td>
<td>-0.000000000023</td>
<td></td>
</tr>
<tr>
<td>$g_4(x^*)$</td>
<td></td>
<td></td>
<td>-0.00000427</td>
<td></td>
</tr>
<tr>
<td>$g_5(x^*)$</td>
<td></td>
<td></td>
<td>-0.23374458</td>
<td></td>
</tr>
<tr>
<td>$f(x^*)$</td>
<td>2.3807</td>
<td>2.38</td>
<td>2.43</td>
<td>2.37713468</td>
</tr>
</tbody>
</table>

Fig. 4. The schematic diagram of the complex (bridge) system.

The approaches applied to this problem include the HS method [23] and GA-based methods [22,4]. In addition, the NMDE algorithm is also used to solve the same problem. The NMDE obtains the best solution at $x^* = (0.24500538884006, 6.28451091444737, 8.19911044972404, 0.24500538886305)$, and its corresponding objective function value is equal to $f(x^*) = 2.37713468$. The comparisons of results are shown in Table 12. The NMDE result is superior to those obtained using the HS method [23] and a binary GA [4], and this result is comparable to that obtained using the GA based on a penalty function approach [22].

The eleventh problem is a nonlinear mixed integer programming problem for a complex (bridge) system with five subsystems (as in Fig. 4), and the problem formulation can be stated as Eq. (22):

$$
\begin{align*}
\max f(\mathbf{r}, \mathbf{n}) &= R_1 R_2 + R_3 R_4 + R_3 R_5 + R_2 R_3 R_5 - R_1 R_2 R_3 R_4 - R_1 R_2 R_3 R_5 - R_1 R_2 R_4 R_5 \\
&- R_1 R_3 R_4 R_5 - R_2 R_3 R_4 R_5 + 2R_1 R_2 R_3 R_4 R_5 \\
\text{s.t.} \quad &g_1(\mathbf{r}, \mathbf{n}) = \sum_{i=1}^{m} w_i n_i^2 - V \leq 0 \\
&g_2(\mathbf{r}, \mathbf{n}) = \sum_{i=1}^{m} \alpha_i \left( -\frac{1000}{\ln(r_i)} \right)^{\beta_i} n_i + \exp(0.25n_i) - C \leq 0 \\
&g_3(\mathbf{r}, \mathbf{n}) = \sum_{i=1}^{m} w_i n_i \exp(0.25n_i) - W \leq 0 \\
&0 \leq r_i \leq 1, \quad n_i \in Z^+, \quad 1 \leq i \leq m.
\end{align*}
$$

Here, $m$ is the number of subsystems in the system; $n_i$ is the number of components in subsystem $i$ (1 ≤ $i$ ≤ $m$); $r_i$ is the reliability of each component in subsystem $i$; $q_i = 1 - r_i$ is the failure probability of each component in subsystem $i$; $R_i(n_i) = 1 - q_i^{n_i}$ is the reliability of subsystem $i$. $f(\mathbf{r}, \mathbf{n})$ is the system reliability. $w_i$ is the weight of each component in subsystem $i$; $v_i$ is the volume of each component in subsystem $i$; and $c_i$ is the cost of each component in subsystem $i$. Furthermore, $V$ is the upper limit on the sum of the subsystems’ products of volume and weight; $C$ is the upper limit on the cost of the system; $W$ is the upper limit on the weight of the system. The parameters $\beta_i$ and $\alpha_i$ are physical features of system components. Constraint $g_1(\mathbf{r}, \mathbf{n})$ is a combination of weight, redundancy allocation and volume. $g_2(\mathbf{r}, \mathbf{n})$ is a cost constraint, while $g_3(\mathbf{r}, \mathbf{n})$ is a weight constraint. The input parameters of the complex (bridge) system are shown in Table 13.

Table 13 compares the best solution obtained using the NMDE algorithm with those obtained by the other approaches. It is clear that the best solution found using the NMDE algorithm is better than the recent studies presented in the literature. For measuring the improvement, MPI (maximum possible improvement) can be used to measure the amount of improvement of the solutions found by the proposed approach to the previous best known solutions, and it is described as: $\text{MPI}(\%) = (f_{\text{Zou}} - f_{\text{other}}) / (1 - f_{\text{other}})$, where $f_{\text{Zou}}$ represents the best system reliability obtained by the proposed algorithm and $f_{\text{other}}$ represents the best system reliability obtained by any other method in the literature. Slack is the unused resource. By using MPI, it shows that the proposed approach made improvements for the complex (bridge) system. With regard to the best
relabilities obtained by Hsieh et al. [32], Chen [33] and Coelho [34], the corresponding improvements made by the proposed approach are 8.6726%, 0.3881% and 0.0634%, respectively. It should be emphasized that even very small improvements in reliability are critical and beneficial to system security and system efficiency.

The twelfth problem is the reliability-redundancy optimization problem of the overspeed protection system for a gas turbine (as in Fig. 5). Overspeed detection is continuously provided by the electrical and mechanical systems. When an overspeed occurs, it is necessary to cut off the fuel supply. For this purpose, four control valves (V1–V4) must close. The control system is modeled as a 4-stage series system. The objective is to determine an optimal level of $r_i$ and $n_i$ at each stage $i$ such that the system reliability is maximized. This reliability problem is formulated as Eq. (23):

$$\text{Max } f(r, n) = \prod_{i=1}^{m} [1 - (1 - r_i)^{n_i}]$$

s.t.

$$g_1(r, n) = \sum_{i=1}^{m} v_i n_i^2 - V \leq 0$$

$$g_2(r, n) = \sum_{i=1}^{m} C(r_i)[n_i + \exp(0.25n_i)] - C \leq 0$$

$$g_3(r, n) = \sum_{i=1}^{m} w_i n_i \exp(0.25n_i) - W \leq 0$$

$$0.5 \leq r_i \leq 1 - 10^{-6}, \ r_i \in R^+, \ 1 \leq n_i \leq 10, \ n_i \in Z^+$$

$r_i$ is reliability of component in stage $i$, and $n_i$ is the number of redundant components in stage $i$. $v_i$ is the product of weight and volume per element at stage $i$. $w_i$ is the weight of each component at stage $i$. The exp($n_i/4$) accounts for the interconnecting hardware. $C(r_i) = \alpha_i \left( - \frac{r_i}{\ln(r_i)} \right)^{\beta_i}$ is the cost of each component with reliability $r_i$ at subsystem $i$. $\alpha_i$ and $\beta_i$ are constants representing the physical characteristics of each component at stage $i$. $T$ is the operating time during which the component must not fail. The input parameters defining the overspeed protection system for a gas turbine are shown in Table 15.

Table 16 compares the best solution found using the NMDE algorithm with those obtained by the other approaches. From Table 16, it can be easily seen that the best solution found using the NMDE algorithm has surpassed recent studies presented in the literature. For the best reliabilities obtained by Yokota et al. [35], Chen [33] and Coelho [34], the corresponding improvements made by the NMDE algorithm are 91.4793%, 21.8448% and 3.5532%, respectively. Based on the above comparison, the proposed approach has demonstrated higher efficiency than the other methods in finding a better solution for problem $f_{12}$.
Table 15
Data used in overspeed protection system.

<table>
<thead>
<tr>
<th>Stage</th>
<th>$a_i$</th>
<th>$\beta_i$</th>
<th>$v_i$</th>
<th>$w_i$</th>
<th>$V$</th>
<th>$C$</th>
<th>$W$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>1.5</td>
<td>1</td>
<td>6</td>
<td>250</td>
<td>400</td>
<td>500</td>
<td>1000h</td>
</tr>
<tr>
<td>2</td>
<td>2.3</td>
<td>1.5</td>
<td>2</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
<td>1.5</td>
<td>3</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2.3</td>
<td>1.5</td>
<td>2</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 16
Comparison of the best result for the overspeed protection system for a gas turbine with other results presented in the literature.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Yokota et al. [35]</th>
<th>Chen [33]</th>
<th>Coelho [34]</th>
<th>Zou</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(r, n)$</td>
<td>0.999468</td>
<td>0.999942</td>
<td>0.999953</td>
<td>0.9995467</td>
</tr>
<tr>
<td>$n_1$</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$n_2$</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$n_3$</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$n_4$</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$r_1$</td>
<td>0.965593</td>
<td>0.903800</td>
<td>0.902231</td>
<td>0.9016148</td>
</tr>
<tr>
<td>$r_2$</td>
<td>0.760592</td>
<td>0.874992</td>
<td>0.856325</td>
<td>0.8499211</td>
</tr>
<tr>
<td>$r_3$</td>
<td>0.972646</td>
<td>0.919898</td>
<td>0.948145</td>
<td>0.9481413</td>
</tr>
<tr>
<td>$r_4$</td>
<td>0.804660</td>
<td>0.890609</td>
<td>0.883156</td>
<td>0.8882286</td>
</tr>
<tr>
<td>MPI(%)</td>
<td>91.4793</td>
<td>21.8448</td>
<td>3.5532</td>
<td>–</td>
</tr>
<tr>
<td>Slack($g_1$)</td>
<td>92</td>
<td>50</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>Slack($g_2$)</td>
<td>70.733576</td>
<td>0.002152</td>
<td>0.975465</td>
<td>0.0001057</td>
</tr>
<tr>
<td>Slack($g_3$)</td>
<td>127.583189</td>
<td>28.803701</td>
<td>24.801882</td>
<td>24.80188272</td>
</tr>
</tbody>
</table>

Fig. 5. Schematic diagram for the overspeed protection system of a gas turbine.

The thirteenth problem is a minimization problem with eight design variables and six inequality constraints, and this problem can be stated as:

$$\min f_{13}(x) = x_1 + x_2 + x_3$$

s.t.

$$g_1(x) = -1 + 0.0025(x_4 + x_6) \leq 0$$
$$g_2(x) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$$
$$g_3(x) = -1 + 0.01(x_8 - x_5) \leq 0$$
$$g_4(x) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83 333.333 \leq 0$$
$$g_5(x) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$$
$$g_6(x) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$$

(24)

where 100 ≤ $x_i$ ≤ 10000, 1000 ≤ $x_i$ ≤ 10000 (i = 2, 3), and 10 ≤ $x_i$ ≤ 1000 (i = 4, . . . , 8). The optimum solution is $x^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.98799, 286.4162, 395.5979)$ where $f(x^*) = 7049.3307$. Three constraints are active ($g_1$, $g_2$ and $g_3$).

This problem has been solved previously by Michalewicz [27], Deb [22] and Lee and Geem [23]. On the other hand, the NMDE is also applied to this problem. The best solution found by the NMDE is $x^* \approx (579.26903752, 1360.10034881, 5109.87870149, 182.01455315, 295.60485271, 217.98544311, 286.40970009, 395.60485246)$, and its corresponding objective function value is equal to $f(x^*) = 7049.24808782$. Table 17 lists the comparisons of results, and it is clear that the best objective function value obtained using the NMDE algorithm is superior to those reported previously in the literature.
The fourteenth optimization problem is a process synthesis and design problem [36,37], and it is described as follows:

\[
\begin{align*}
\min f_{14}(\mathbf{x}) &= (x_4 - 1)^2 + (x_5 - 1)^2 + (x_6 - 1)^2 - \ln(x_7 + 1) + (x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2 \\
\text{s.t.} \\
g_1(\mathbf{x}) &= x_4 + x_5 + x_6 + x_1 + x_2 + x_3 - 5 \leq 0 \\
g_2(\mathbf{x}) &= x_6^2 + x_2^2 + x_3 - 5.5 \leq 0 \\
g_3(\mathbf{x}) &= x_4 + x_1 - 1.2 \leq 0 \\
g_4(\mathbf{x}) &= x_5 + x_2 - 1.8 \leq 0 \\
g_5(\mathbf{x}) &= x_6 + x_3 - 2.5 \leq 0 \\
g_6(\mathbf{x}) &= x_7 + x_1 - 1.2 \leq 0 \\
g_7(\mathbf{x}) &= x_5^2 + x_6^2 - 1.64 \leq 0 \\
g_8(\mathbf{x}) &= x_4^2 + x_2^2 - 4.25 \leq 0 \\
g_9(\mathbf{x}) &= x_3^2 + x_4^2 - 4.64 \leq 0 \\
x_1 \in [0, 1.2], \, x_2 \in [0, 1.8], \, x_3 \in [0, 2.5], \, x_4, x_5, x_6, x_7 \in [0, 1].
\end{align*}
\] (25)

This problem is a nonlinear mixed integer programming problem. The global best solution reported by Angira and Babu [36] and Liao [37] was \(x^* = (0.2, 1.28062, 1.95448, 1.0, 0, 0, 1)\), and its corresponding objective function value was \(f(x^*) = 3.557473\). In addition, the NMDE is also used to solve this problem. The best solution found by the NMDE is \(x^* = (0.2, 1.28062484, 1.95448203, 1.0, 0, 0, 1)\), and its corresponding objective function value is equal to \(f(x^*) = 3.557461\). Table 18 shows the comparisons of results, and it can be observed that the NMDE solution is better than those reported by Angira and Babu [36] and Liao [37].

6. Conclusions

In this paper, a novel modified differential evolution algorithm (NMDE) is proposed to solve constrained optimization problems. For any solution, the NMDE adaptively adjusts its scale factor and crossover rate using uniform distribution if
and only if stagnation happens to this solution. The adaptive scale factors and crossover rates of all solutions are beneficial to balancing the global search and the local search of the NMDE algorithm, and enable it to explore the solution space sufficiently. Experimental results show that the NMDE algorithm has higher efficiency than the other methods in the literature on finding better feasible solutions of most constrained problems.

Acknowledgement

This work was supported by National Science Foundation of PR China under the Grant 60674021.

References