A Hybrid Simplex Search and Particle Swarm Optimization for Nonlinear Programming

Erwie Zahara\textsuperscript{1,}\textsuperscript{†} , Yi-Tung Kao\textsuperscript{2} and Chia-Hsin Hu\textsuperscript{3}

\textsuperscript{1,3} Department of Industrial Engineering and Management, St. John’s University, Tamsui, Taiwan 251, Republic of China

\textsuperscript{2} Department of Computer Science and Engineering, Tatung University, Taipei City, Taiwan 104, Republic of China

Abstract: Nonlinear programming models often arise in science and engineering. A nonlinear programming model consists of the optimization of a function subject to constraints, in which both the function and constraints may be nonlinear. Constraint handling is one of the major concerns when solving nonlinear programming problems by hybrid Nelder-Mead simplex search method and particle swarm optimization, denoted as NM-PSO. This paper proposes embedding constraint handling methods, which include the gradient repair method and constraint fitness priority-based ranking method, in NM-PSO as a special operator to deal with satisfying constraints. Experiments using 6 benchmark problems are presented and compared with the best known solutions reported in the literature. The comparison results with three different metaheuristics demonstrate that NM-PSO with the embedded constraint operator proves to be extremely effective and efficient at locating optimal solutions.

Keywords: nonlinear programming, Nelder-Mead simplex search method, particle swarm optimization, constraint handling.

1. INTRODUCTION

Nonlinear programming (NLP) is a mathematical programming technique which attempts to solve objective functions that are nonlinear, or deals with constraints that have a nonlinear relationship, or handles both at the same time. It has become an important branch of operations research and has a wide variety of applications in such areas as the military, economics, engineering optimization, and management science (Nash and Sofer, 1996). NLP problems with \( n \) variables and \( m \) constraints may be written in the following canonical form:

\[
\begin{align*}
\text{Minimize} & \quad f(x) \\
\text{subject to} & \quad g_j(x) \leq 0, \quad j = 1, \ldots, q \\
& \quad h_j(x) = 0, \quad j = q+1, \ldots, m \\
& \quad l_i \leq x_i \leq u_i, \quad i = 1, \ldots, n
\end{align*}
\]

where \( x = (x_1, x_2, \ldots, x_n) \) is an \( n \) dimensional vector of decision variables, \( f(x) \) is an objective function, \( g_j(x) \leq 0 \) are \( q \) inequality constraints, and \( h_j(x) = 0 \) are \( m-q \) equality constraints. The values \( u_i \) and \( l_i \) are the upper and the lower bounds of \( x_i \), respectively.

Traditionally, most of the methods found in the literature search for approximate solutions to constraint optimization problems and assume that the goal and constraints are differentiable (Fung \textit{et al}., 2002). In case that the goal and constraints are not differentiable, heuristic methods are employed. Optimization methods using heuristics, which have helped researchers formulate such new stochastic algorithms as the genetic algorithm (Chootinan and Chen, 2006), evolutionary optimization (Runarsson and Yao, 2005), and particle swarm optimization (Dong \textit{et al}., 2005), are of little practical use in solving constrained optimization problems due to slow convergence rates. Fan and Zahara (2007) have demonstrated the hybrid Nelder-Mead simplex method and particle swarm optimization method (NM-PSO) to be a promising and viable tool for solving unconstrained nonlinear optimization problems. The very method can also be adopted to solving constrained nonlinear optimization problems in general, which is the focus of this paper. A constraint

\textsuperscript{†}Correspondence should be addressed to: erwi@mail.sju.edu.tw
fitness priority-based ranking method (Dong et al., 2005) and a repair method using the gradient information derived from the constraint set (Chootinan and Chen, 2006) are embedded in NM-PSO to evaluate the particles during the search when solving constraint optimization problems. The superior performance of the proposed NM-PSO will be demonstrated by solving 6 benchmark constraint optimization problems.

2. CONSTRAINT HANDLING METHODS

This section will introduce two constraint-handling methods: the gradient repair method and constraint fitness priority-based ranking method, which will be embedded in NM-PSO.

2.1 The Gradient Repair Method

The gradient repair method was proposed by Chootinan and Chen (2006), and it utilizes gradient information derived from the constraint set to systematically repair infeasible solutions. Basically, the gradient information is used to direct infeasible solutions toward the feasible region. The procedure is summarized below:

1. For any solution, determine the degree of constraint violation \( \Delta V \) by Equation (2).

\[
V = \begin{bmatrix} g_{eql} \\ h_{(neq-g)=0} \end{bmatrix}_{n \times 1} \Rightarrow \Delta V = \begin{bmatrix} \text{Min}[0, -g(x)] \\ -h(x) \end{bmatrix}
\]  

(2)

where \( V \) consist of vectors of inequality constraints (\( g \)) and equality constraints (\( h \)) for the problem.

2. Compute \( \nabla_x V \), where \( \nabla_x V \) are the derivatives of these constraints with respect to the solution vector (n decision variables) and \( e \) is a small scalar for perturbation.

\[
\nabla_x V = \begin{bmatrix} \nabla_x g \\ \nabla_x h \end{bmatrix}_{n \times 1} = \frac{1}{e} \begin{bmatrix} g(x | x_i = x_i + e) - g(x), \forall i = 1, 2, \ldots n \\ h(x | x_i = x_i + e) - h(x), \forall i = 1, 2, \ldots n \end{bmatrix}
\]  

(3)

Therefore, the relationship between changes of constraint violation \( \Delta V \) and solution vector \( \Delta x \) is expressed by

\[
\Delta V = \nabla_x V \times \Delta x \Rightarrow \Delta x = \nabla_x V^{-1} \times \Delta V
\]  

(4)

3. Compute the Moore-Penrose inverse or pseudoinverse \( \nabla_x V^+ \) which is the approximate inverse of \( \nabla_x V \) and is to be used instead in Equation (4).

4. Update the solution vector by \( x^{t+1} = x^t + \Delta x = x^t + \nabla_x V^{-1} \times \Delta V \approx x^t + \nabla_x V^+ \times \Delta V \).

5. Loop through steps 1 to 4 at most five times.

2.2 Constraint Fitness Priority-Based Ranking Method

Y Dong et al. (2005) proposed this method and introduced constraint fitness function \( C_f(x) \) for constraints, and it is computed from both inequality and equality constraints as follows:

For inequality constraints \( g_j(x) \leq 0 \),

\[
C_j(x) = \begin{cases} 
1, & \text{If } g_j(x) \leq 0 \\
1 - \frac{g_j(x)}{g_{\text{max}}(x)}, & \text{If } g_j(x) > 0
\end{cases}
\]  

(5)

where \( g_{\text{max}}(x) = \max[g_j(x), j = 1, 2, \cdots q] \) and \( C_j(x) \) is the fitness level of point \( x \) for constrained condition (j).
For equality constraints \( h_j(x) = 0 \),

\[
C_j(x) = \begin{cases} 
1, & \text{If } h_j(x) = 0 \\
1 - \frac{|h_j(x)|}{h_{\text{max}}(x)}, & \text{If } h_j(x) \neq 0 
\end{cases}
\]  

(6)

where \( h_{\text{max}}(x) = \max\{h_j(x), \, j = q+1, \, q+2, \ldots, m\} \).

The constraint fitness function evaluated at point \( x \) is equal to the weighted sum of the \( C_j \)'s, as defined below:

\[
C_f(x) = \sum_{j=1}^{m} w_j C_j(x), \quad \sum_{j=1}^{m} w_j = 1, \quad 0 \leq w_j \leq 1, \quad \forall j.
\]  

(7)

where \( w_j \) is a randomly generated weight for constraint \( j \). The sum signifies the fitness level of point \( x \) as related to the feasible domain \( Q \); that is, if \( C_f(x) = 1 \), it is an indication that \( x \in Q \), and on the contrary, if \( 0 < C_f(x) < 1 \), the smaller \( C_f(x) \) is, the less likely that \( x \) is in the feasible domain \( Q \) or the further away \( x \) is from \( Q \).

3. HYBRID NELDER-MEAD AND PARTICLE SWARM OPTIMIZATION

The goal of integrating Nelder-Mead (NM) simplex method and particle swarm optimization (PSO) is to combine their advantages and avoid disadvantages. For example, NM simplex method is a very efficient local search procedure but its convergence is excessively sensitive to the selected starting point; PSO belongs to the class of global search procedures but requires much computational effort (Fan and Zahara, 2007). This section starts by introducing the procedures of NM and PSO, followed by a description of the hybrid Nelder-Mead and particle swarm optimization method.

3.1 The Nelder-Mead Simplex Search Method (NM)

The Nelder-Mead simplex search method is proposed by Nelder and Mead (1965), which is a local search method designed for unconstrained optimization without using gradient information. The operations of this method are to rescale the simplex based on the local behavior of the function by using four basic procedures: reflection, expansion, contraction and shrinkage. Through these procedures, the simplex can successively improve itself and zero in on the optimum. To solve a constrained optimization problem, the steps of NM are described details below:

1. Initialization. An initial \( N+1 \) vertex points are randomly generated according to their search space. Evaluate fitness of the objective function and the constraint fitness at each vertex point of the simplex. For the maximization case, it is convenient to transform the problem into the minimization case by pre-multiplying the objective function by \(-1\).

2. Reflection. Determine \( X_{\text{high}} \) and \( X_{\text{low}} \), vertices with the highest and the lowest function values, respectively. Let \( f_{\text{high}}, \, f_{\text{low}} \) and \( C_f_{\text{high}}, \, C_f_{\text{low}} \) represent the corresponding observed function values and constraint function values, respectively. Find \( X_{\text{cent}} \), the center of the simplex without \( X_{\text{high}} \) in the minimization case. Generate a new vertex \( X_{\text{refl}} \) by reflecting the worst point according to the following equation

\[
X_{\text{refl}} = (1 + \alpha)X_{\text{cent}} - \alpha X_{\text{high}} \quad (\alpha > 0)
\]  

(8)

Nelder and Mead suggested that \( \alpha = 1 \). If \( C_f_{\text{refl}} < 1 \) and \( C_f_{\text{refl}} > C_f_{\text{low}} \) or \( C_f_{\text{refl}} = 1 \) and \( f_{\text{refl}} < f_{\text{low}} \) then proceed with expansion; otherwise proceed with contraction.

3. Expansion. After reflection, two possible expansion cases need to be considered, as described below:
3.1 If $C_{r_{refl}} < 1$ and $C_{f_{refl}} > C_{f_{low}}$, the simplex is expanded in order to extend the search space in the same direction and the expansion point is calculated by Equation (9):

$$X_{exp} = \gamma X_{refl} + (1 - \gamma) X_{cent} \quad (\gamma > 1) \quad (9)$$

Nelder and Mead suggested $\gamma = 2$. If $C_{f_{exp}} > C_{f_{low}}$ or $C_{f_{exp}} = 1$, the expansion is accepted by replacing $X_{high}$ with $X_{exp}$; otherwise, $X_{refl}$ replaces $X_{high}$.

3.2 If $C_{f_{refl}} = 1$ and $f_{refl} < f_{low}$, the expansion point is calculated by Equation (9).

If $C_{f_{exp}} = 1$ and $f_{exp} < f_{low}$, the expansion is accepted by replacing $X_{high}$ with $X_{exp}$; otherwise, $X_{refl}$ replaces $X_{high}$.

3.3 Exit the algorithm if the stopping criteria are satisfied; otherwise go to step 2.

4. **Contraction.** After reflection, there are two possible contraction cases to consider:

4.1 If $C_{f_{refl}} < 1$ and $C_{f_{refl}} \leq C_{f_{low}}$, the contraction vertex is calculated by Equation (10).

$$X_{cont} = \beta X_{high} + (1 - \beta) X_{cent} \quad (0 < \beta < 1) \quad (10)$$

Nelder and Mead suggested $\beta = 0.5$. If $C_{f_{cont}} > C_{f_{low}}$ or $C_{f_{cont}} = 1$, the contraction is accepted by replacing $X_{high}$ with $X_{cont}$; otherwise do shrinking in step 5.1.

4.2 If $C_{f_{refl}} = 1$ and $f_{refl} \geq f_{low}$, and if $f_{refl} \leq f_{high}$, then $X_{refl}$ replaces $X_{high}$ and contraction by Equation (10) is carried out. If $f_{refl} > f_{high}$, then direct contraction without the replacement of $X_{high}$ by $X_{refl}$ is performed. If $C_{f_{cont}} = 1$ and $f_{cont} < f_{low}$, the contraction is accepted by replacing $X_{high}$ with $X_{cont}$; otherwise do shrinking in step 5.2.

4.3 Exit the algorithm if the stopping criteria are satisfied; otherwise go to step 2.

5. **Shrinkage.**

5.1 Following step 4.1 in which $C_{f_{cont}} \leq C_{f_{low}}$ and contraction has failed, shrinkage attempts to all points except $X_{low}$ by Equation (11):

$$X_{i} \leftarrow \delta X_{i} + (1 - \delta) X_{low} \quad (0 < \delta < 1) \quad (11)$$

Nelder and Mead suggested $\delta = 0.5$.

5.2 Contraction has failed in step 4.2 where $C_{f_{cont}} < 1$ or $C_{f_{cont}} = 1$ but $f_{cont} \geq f_{low}$. Shrink the entire simplex except $X_{low}$ by Equation (11).

5.3 Exit the algorithm if the stopping criteria are satisfied; otherwise go to step 2.

3.2 **Particle Swarm Optimization (PSO)**

Particle swarm optimization (PSO) is one of the latest evolutionary optimization techniques developed by Kennedy and Eberhart (1995). PSO concept is based on a metaphor of social interaction such as bird flocking and fish schooling. Similar to genetic algorithms, PSO is also population-based and evolutionary in nature, with one major difference from genetic algorithms that it does not implement filtering, i.e., all members in the population survive through the entire search process. PSO simulates a commonly observed social behavior, where members of a group tend to follow the lead of the best of the group. The procedure of PSO is reviewed below.
1. Initialization. Randomly generate a swarm of the potential solutions, called “particles” and assign a random velocity to each.

2. Velocity Update. The particles are then “flown” through hyperspace by updating their own velocity. The velocity update of a particle is dynamically adjusted, subject to its own past flight and those of its companions. The particle’s velocity and position are updated by the following equations:

$$V_{id}^{\text{New}}(t + 1) = c_0 \times V_{id}^{\text{old}}(t) + c_1 \times \text{rand}() \times (p_{id}(t) - x_{id}^{\text{old}}(t)) + c_2 \times \text{rand}() \times (p_{gd}(t) - x_{id}^{\text{old}}(t))$$  

(12)

$$x_{id}^{\text{New}}(t + 1) = x_{id}^{\text{old}}(t) + V_{id}^{\text{New}}(t + 1)$$  

(13)

where $c_1$ and $c_2$ are two positive constants; $c_0$ is an inertia weight and $\text{rand}()$ is a random value inside (0, 1). Eberhart and Shi (2001) and Hu and Eberhart (2001) suggested $c_1 = c_2 = 2$ and $c_0 = [0.5 + (\text{rand}() / 2.0)]$. Equation (12) illustrates the calculation of a new velocity for each individual. The velocity of each particle is updated according to its previous velocity ($V_{id}$), the particle’s previous best location ($p_{id}$) and the global best location ($p_{gd}$). Particle’s velocities on each dimension are clamped to a maximum velocity $V_{max}$, where the maximum velocity $V_{max}$ is a fraction of the domain search space in each dimension. Equation (13) shows how each particle’s position is updated in the search space.

3.3 Hybrid NM-PSO Method

We now proceed to present embedding the constraint handling methods in NM-PSO. The population size of this hybrid NM-PSO approach is set at $21N + 1$ when solving an $N$-dimensional problem. The initial population is randomly generated in the problem search space. For every particle that violates the constraints, use the gradient repair method to direct the infeasible solution toward the feasible region. In most cases, the repair method does move the solution to the feasible region. However, there may be times when the repair method fails; in such cases, the infeasible solutions are left as they are and the process continues with the next step. The constraint fitness priority-based ranking method is then evaluates the constraint fitness value of each particle and ranks them according their constraint fitness value with the particle having the greatest (constraint fitness) value computed by (7) placed at the top. If two particles have the same constraint fitness value, then their objective fitness values are compared in order to determine their relative position. The one with better objective fitness value is positioned in front of the other. The top $N + 1$ particles are then fed into the Nelder-Mead simplex search method to improve the $(N + 1)$\textsuperscript{th} particle. The PSO method adjusts the other $20N$ particles by taking into account the positions of the $N + 1$ best particles. This procedure for adjusting the $20N$ particles involves selection of the global best particle, selection of the neighborhood best particles, and finally velocity updates. The global best particle of the population is determined according to the sorted fitness values. Next, the $20N$ particles are evenly divided into $10N$ neighborhoods with 2 particles per neighborhood. The particle with the better fitness value in each neighborhood is denoted as the neighborhood best particle. By equations (12) and (13), a velocity update for each of the $20N$ particles is then carried out. The $21N + 1$ particles are sorted in preparation for repeating the entire run. The process terminates when a certain convergence criterion is satisfied. Figure 1 summarizes the algorithm of NM-PSO.

4. EXPERIMENTAL RESULTS

In this study, we applied the proposed NM-PSO to solve the same 6-benchmark problems as studied by Becerra and Coello (2006). This test suite includes two maximization problems (G02 and G03) and the others are minimization problems. They are a set of nonlinear functions for difficult constrained optimization problems. The forms for the functions are described completely in the appendix. The algorithm is coded in Matlab 7.0 and the simulations are run on a Pentium IV 2.4GHz with 512 MB memory capacity.

The task of optimizing each of the test functions was executed 20 times by NM-PSO, and the search process was iterated 5000 times, or until the reference or a better solution was found. Table 1 contains a summary of the execution results of NM-PSO, and includes the worst and the best solutions obtained, the means, the medians and the standard deviations of the solutions, averaged numbers of function evaluations and average numbers of iteration. For the sake of comparison, the table also gives the reference optimal values. For every problem, the best solutions are almost equal to the optimal solutions. For problems G05 the optimal solutions are better than the reference. For problems G01, G03, G04, G05 and G06 the optimal solutions are found consistently in all 20 runs. For problems G02 the optimal solutions were not
consistently found. In term of the efficiency of NM-PSO, Table 1 shows that more than half of test problems converged in less than 300 iterations. However, test problems G02 never completely converged; they all stopped at the preset maximum number of iterations, which is 5000, with a high standard deviation, signifying that the solutions found thus far are still quite far apart.

1. **Initialization.** Generate a population of size $(N+1)$.

Repeat

2. **Constraint handling methods**

2.1 **The Gradient Repair Method.** Repair particles that violate the constraints by directing the infeasible solution toward the feasible region. Leave unrepairable solutions as they are.

2.2 **Constraint Fitness Priority-Based Ranking Method.** Evaluate the constraint fitness, the objective fitness of each particle and rank them.

3. **Simplex Method.** Apply NM operator to the top $N+1$ particles and update the $(N+1)$th particle.

4. **PSO Method.** Apply PSO operator for updating the remaining $20N$ particles with worst fitness.

4.1 **Selection.** From the population select the global best particle and the neighborhood best particles.

4.2 **Velocity Update.** Apply velocity update to the $20N$ particles with worst fitness according to Equations (12) and (13).

Until some termination condition is met.

![Figure 1. The hybrid NM-PSO algorithm embedded with constraint handling methods.](image)

**Table 1. Results of test problems using NM-PSO**

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Reference optimal value</th>
<th>Best objective value</th>
<th>Median objective value</th>
<th>Mean objective value</th>
<th>Standard deviation</th>
<th>Worst objective value</th>
<th>Average of fune_evaluation</th>
<th>Average of Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>G01</td>
<td>Min</td>
<td>-15</td>
<td>-15</td>
<td>-15</td>
<td>-15</td>
<td>0.000000</td>
<td>-15</td>
<td>41,959</td>
<td>142</td>
</tr>
<tr>
<td>G02</td>
<td>Max</td>
<td>0.803619</td>
<td>0.803619</td>
<td>0.796464</td>
<td>0.796431</td>
<td>0.002522</td>
<td>0.796422</td>
<td>2,229,858</td>
<td>5000</td>
</tr>
<tr>
<td>G03</td>
<td>Max</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.000000</td>
<td>1</td>
<td>64,108</td>
<td>282</td>
</tr>
<tr>
<td>G04</td>
<td>Min</td>
<td>-30665.5386</td>
<td>-30665.5386</td>
<td>-30665.5386</td>
<td>-30665.5386</td>
<td>0.000000</td>
<td>-30665.5386</td>
<td>19,658</td>
<td>169</td>
</tr>
<tr>
<td>G05</td>
<td>Min</td>
<td>5126.4981</td>
<td>5126.359</td>
<td>5126.359</td>
<td>5126.359</td>
<td>0.000000</td>
<td>5126.359</td>
<td>25,253</td>
<td>269</td>
</tr>
<tr>
<td>G06</td>
<td>Min</td>
<td>-6961.8138</td>
<td>-6961.824</td>
<td>-6961.824</td>
<td>-6961.824</td>
<td>0.000000</td>
<td>-6961.824</td>
<td>9,856</td>
<td>198</td>
</tr>
</tbody>
</table>

The same 6 benchmark problems are solved by the cultural differential evolution (CDE) algorithm (Becerra and Coello, 2006), the filter simulated annealing (FSA) method (Hedar and Fukushima, 2006), and the genetic algorithm (GA) (Chootinan and Chen, 2006). Table 2 lists the results of solving these problems by the various methods in 20 runs in terms of the best, the mean and the worst of the solutions, the standard deviation and the average of function evaluations of each of the methods. Better values in each category are highlighted in boldface. It is seen in Table 2 that NM-PSO obtained better solutions in all respects for test problems G01, G03, G04, G05 and G06. For test problems G02 NM-PSO obtained better global optimal solutions than the other methods but required more function evaluations. NM-PSO surpassed the other three methods in all 6 problems. We therefore conclude that NM-PSO is a suitable tool for solving constraint optimization problems.

**4. CONCLUSIONS**

In this paper, NM-PSO with embedded constraint handling methods is proposed for solving the constraint optimization problems. Experimental results clearly illustrate the attractiveness of the method for handling several types of constraint. It can produce competitive, if not better, solutions as compared to CDE, FSA and GA, which appear to be the promising constraint-handling techniques reported thus far in the literature. In the future, we will apply NM-PSO to various problems found in the real world. Additionally, we want to explore the benefits of using a belief space in other type of problems. Specifically, we are interested in extending our approach so that it can deal with multi-objective problems.
Table 2. Comparison results of test problems with CDE, FSA and GA

<table>
<thead>
<tr>
<th>No</th>
<th>Type</th>
<th>Algorithm</th>
<th>Best objective value</th>
<th>Mean objective value</th>
<th>Worst objective value</th>
<th>Standard deviation</th>
<th>Average of func evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>G01</td>
<td>Min</td>
<td>CDE</td>
<td>-15.000000</td>
<td>-14.999996</td>
<td>-14.999993</td>
<td>0.000002</td>
<td>100,100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FSA</td>
<td>-14.999105</td>
<td>-14.993316</td>
<td>-14.979977</td>
<td>0.004813</td>
<td>205,748</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GA</td>
<td>-15.000000</td>
<td>-15.000000</td>
<td>-15.000000</td>
<td>0.000000</td>
<td>95,512</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NM-PSO</td>
<td>-15.000000</td>
<td>-15.000000</td>
<td>-15.000000</td>
<td>0.000000</td>
<td>41,959</td>
</tr>
<tr>
<td>G02</td>
<td>Max</td>
<td>CDE</td>
<td>0.803619</td>
<td>0.724886</td>
<td>0.590908</td>
<td>0.070125</td>
<td>100,100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FSA</td>
<td>0.754912</td>
<td>0.371708</td>
<td>0.271311</td>
<td>0.098023</td>
<td>227,832</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GA</td>
<td>0.801119</td>
<td>0.785746</td>
<td>0.745329</td>
<td>0.013700</td>
<td>331,972</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NM-PSO</td>
<td>0.803619</td>
<td>0.796431</td>
<td>0.796422</td>
<td>0.002522</td>
<td>2,229,858</td>
</tr>
<tr>
<td>G03</td>
<td>Max</td>
<td>CDE</td>
<td>0.995413</td>
<td>0.788635</td>
<td>0.639920</td>
<td>0.115214</td>
<td>100,100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FSA</td>
<td>1.000001</td>
<td>0.999187</td>
<td>0.991518</td>
<td>0.001653</td>
<td>314,938</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GA</td>
<td>0.999980</td>
<td>0.999920</td>
<td>0.999790</td>
<td>0.000060</td>
<td>399,804</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NM-PSO</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>64,108</td>
</tr>
<tr>
<td>G04</td>
<td>Min</td>
<td>CDE</td>
<td>-30665.5387</td>
<td>-30665.5387</td>
<td>-30665.5387</td>
<td>0.000000</td>
<td>100,100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FSA</td>
<td>-30665.5380</td>
<td>-30665.4665</td>
<td>-30664.6880</td>
<td>0.173218</td>
<td>86,154</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GA</td>
<td>-30665.5386</td>
<td>-30665.5386</td>
<td>-30665.5386</td>
<td>0.000000</td>
<td>26,981</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NM-PSO</td>
<td>-30665.5386</td>
<td>-30665.5386</td>
<td>-30665.5386</td>
<td>0.000000</td>
<td>19,658</td>
</tr>
<tr>
<td>G05</td>
<td>Min</td>
<td>CDE</td>
<td>5126.5709</td>
<td>5207.4107</td>
<td>5327.3905</td>
<td>69.225796</td>
<td>100,100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FSA</td>
<td>5126.4981</td>
<td>5126.4981</td>
<td>5126.4981</td>
<td>0.000000</td>
<td>47,661</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GA</td>
<td>5126.4981</td>
<td>5126.4981</td>
<td>5126.4981</td>
<td>0.000000</td>
<td>39,459</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NM-PSO</td>
<td>5126.3590</td>
<td>5126.3590</td>
<td>5126.3590</td>
<td>0.000000</td>
<td>25,253</td>
</tr>
<tr>
<td>G06</td>
<td>Min</td>
<td>CDE</td>
<td>-6961.8139</td>
<td>-6961.8139</td>
<td>-6961.8139</td>
<td>0.000000</td>
<td>100,100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FSA</td>
<td>-6961.8139</td>
<td>-6961.8139</td>
<td>-6961.8139</td>
<td>0.000000</td>
<td>44,538</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GA</td>
<td>-6961.8139</td>
<td>-6961.8139</td>
<td>-6961.8139</td>
<td>0.000000</td>
<td>13,577</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NM-PSO</td>
<td>-6961.8240</td>
<td>-6961.8240</td>
<td>-6961.8240</td>
<td>0.000000</td>
<td>9,856</td>
</tr>
</tbody>
</table>

REFERENCES


**APPENDIX**

\[ G01 \]

**Minimize**

\[
 f(x) = 5 \sum_{i=1}^{4} x_i - 5 \sum_{i=1}^{4} x_i^3 - \sum_{i=1}^{13} x_i
\]

s.t.

\[
 g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0
\]

\[
 g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0
\]

\[
 g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0
\]

\[
 g_4(x) = -8x_1 + x_{10} \leq 0
\]

\[
 g_5(x) = -8x_2 + x_{10} \leq 0
\]

\[
 g_6(x) = -8x_3 + x_{12} \leq 0
\]

\[
 g_7(x) = -2x_4 - x_5 + x_{10} \leq 0
\]

\[
 g_8(x) = -2x_6 - x_7 + x_{10} \leq 0
\]

\[
 g_9(x) = -2x_8 - x_9 + x_{12} \leq 0
\]

\[
 0 \leq x_i \leq 1, \quad i = 1, 2, \ldots, 9,
\]

\[
 0 \leq x_i \leq 100, \quad i = 10, 11, 12,
\]

\[
 0 \leq x_i \leq 1, \quad i = 13,
\]

\[ G02 \]

**Maximize**

\[
 f(x) = \left| \sum_{i=1}^{n} \cos^4(x_i) - 2 \prod_{i=1}^{n} \cos^2(x_i) \right| \sqrt{\sum_{i=1}^{n} i x_i^2}
\]

s.t.

\[
 g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \leq 0
\]

\[
 g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \leq 0
\]

\[
 n = 20
\]

\[
 0 \leq x_i \leq 10, \quad i = 1, \ldots, n
\]
**G03**

Maximize

\[ f(x) = (\sqrt{n})^n \prod_{j=1}^{n} x_j \]

s.t.

\[ h_i(x) = \sum_{j=1}^{n} x_j^2 - 1 = 0 \]

\[ n = 10 \]

\[ 0 \leq x_i \leq 1, \ i = 1, \ldots, n \]

---

**G04**

Minimize

\[ f(x) = 5.3578547x_1^2 + 0.8356891x_1x_2 + 37.293239x_1 - 40792.141 \]

s.t.

\[ g_1(x) = 85.334407 + 0.0056858x_2x_5 - 92 \leq 0 \]

\[ g_2(x) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0 \]

\[ g_3(x) = 80.51249 + 0.0071317x_2x_4 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0 \]

\[ g_4(x) = -80.51249 - 0.0071317x_2x_4 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0 \]

\[ g_5(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0 \]

\[ g_6(x) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 + 0.0019085x_3x_4 + 20 \leq 0 \]

\[ 78 \leq x_1 \leq 102 \]

\[ 33 \leq x_2 \leq 45 \]

\[ 27 \leq x_3 \leq 45, \ i = 3, 4, 5 \]

---

**G05**

Minimize

\[ f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002 / 3)x_2^3 \]

s.t.

\[ g_1(x) = -x_4 + x_3 - 0.55 \leq 0 \]

\[ g_2(x) = -x_3 + x_4 - 0.55 \leq 0 \]

\[ h_3(x) = 1000 \sin(-x_3 - 0.25) + 1000 \sin(-x_4 - 0.25) + 894.8 - x_1 = 0 \]

\[ h_4(x) = 1000 \sin(x_3 - 0.25) + 1000 \sin(x_3 - x_4 - 0.25) + 894.8 - x_1 = 0 \]

\[ h_5(x) = 1000 \sin(x_4 - 0.25) + 1000 \sin(x_4 - x_3 - 0.25) + 1294.8 = 0 \]

\[ 0 \leq x_1 \leq 1200 \]

\[ 0 \leq x_2 \leq 1200 \]

\[ 0.55 \leq x_i \leq -0.55, \ i = 3, 4, \]
\[
\begin{align*}
\text{Minimize} & \quad f(x) = (x_1 - 10)^3 + (x_2 - 20)^3 \\
\text{s.t.} & \quad g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \\
& \quad g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0 \\
& \quad 13 \leq x_1 \leq 100 \\
& \quad 0 \leq x_2 \leq 100
\end{align*}
\]