PARTICLE SWARM-GROUP SEARCH ALGORITHM AND ITS APPLICATION TO SPATIAL STRUCTURAL DESIGN WITH DISCRETE VARIABLES

S.K. Zeng and L.J. Li*

School of Civil and Transportation Engineering, Guangdong University of Technology, Guangzhou, China

ABSTRACT

Based on introducing two optimization algorithms, group search optimization (GSO) algorithm and particle swarm optimization (PSO) algorithm, a new hybrid optimization algorithm which named particle swarm-group search optimization (PS-GSO) algorithm is presented and its application to optimal structural design is analyzed. The PS-GSO is used to investigate the spatial truss structures with discrete variables and is tested by truss optimization problems. The optimization results are compared with that of the HPSO and GSO algorithm. The results show that the PS-GSO is able to accelerate the convergence rate effectively and has the fastest convergence rate among these three algorithms. The research shows the proposed PS-GSO algorithm can be effectively applied to optimal design of spatial structures with discrete variables.

Received: 5 August 2012; Accepted: 10 September 2012

KEYWORDS: particle swarm optimization; group search optimization; hybrid optimization algorithm; discrete variables; spatial structures

1. INTRODUCTION

Bionic optimization algorithms, notably Evolutionary Algorithms (EAs) had been widely used to solve various scientific and engineering problems and have been extensively used in structural optimization problems recently. Among them, ant colony optimizer (ACO),
particle swarm optimizer and group search optimization inspired by Dorigo [1], Kennedy and Eberhart [2], Barnard and He [3, 4] respectively, are three typical representatives. The ‘individual behavior’ of group is mainly considered by ACO and PSO, in which evolution behavior is based on the evolutionary theory. These two algorithms belong to ‘evolutionary strategies’ areas in a way. ACO is good at solving discrete variable tasks and combination optimization problems but shows a low evolutionary velocity [5]. PSO suits for continuous and discrete variables optimization problems but is easy to entrap into local minima [6-9]. Also they are time consuming in optimizing complex structures. As we know, gregariousness is a common phenomenon in animality, ‘information communication’ and ‘mutual cooperation’ are two important aspects of group behavior. Group search optimizer (GSO) is such an optimization algorithm which is based upon this group specialty and also has been successfully used in structural optimal design with continuous variables [10-12]. However, as the bars, the areas of cross-sections of practical engineering structures are produced according to specifications, structural optimization with discrete variables is of value of practical application and shows more obvious significance.

To improve the efficiency of PSO for structural optimization, a new hybrid optimization algorithm named Particle Swarm-Group Search Optimization (PS-GSO) is presented in this paper, which is based on the combination of the particle swarm optimizer (PSO) and group search optimizer (GSO). Compared with the heuristic particle warm algorithm [7], the basic GSO [13] algorithm, the genetic algorithm [14], the hybrid genetic algorithm [15], and the harmony search heuristic algorithm [16, 17], this hybrid PS-GSO algorithm has preferable convergence rate and accuracy.

2. MATHEMATICAL MODEL OF OPTIMIZATION DESIGN

The mathematical model of optimization design can be express as follows:

\[
\text{min } f(X) \\
\text{subject to } g_i(X) \leq 0 , \ i = 1, 2, ..., m
\]

where \( f(X) \) is the objective function, and \( g_i(X) \) is the inequality constraints. The variables vector \( X \) represents a set of the design variables, and \( m \) is the number of constraints.

3. GROUP SEARCH OPTIMIZER (GSO) ALGORITHM

GSO [10] is inspired by the food searching behavior and group living theory of social animals, such as birds, fish and lions. The foraging strategies of these animals mainly include: producing, e.g., searching for food; and joining (scrounging), e.g., joining resources uncovered by others. GSO also employs ‘rangers’ which perform random walks to avoid entrapment in local minima. Therefore, in GSO, a group consists of three kinds of members:
producers, scroungers and rangers. At each iteration, a group member, located in the most promising area, conferring the best fitness value, is chosen as the producer. It locates in the most promising area and stays still. The other group members are selected as scroungers or rangers by random. Then, each scrounger makes a random walk towards the producer, and the rangers make a random walk in arbitrary direction. It is also assumed that the producer, scroungers and rangers do not differ in their relevant phenotypic characteristics. Therefore, they can switch among the three roles. The GSO behaves as follows [18]:

In an n-dimensional search space, the \( i_{th} \) member at the \( k_{th} \) searching bout (iteration) has a current position \( X_i^k \in \mathbb{R}^n \), a head angle \( \varphi_i = (\varphi_i^1, ..., \varphi_i^{n-1}) \in \mathbb{R}^{n-1} \) and a head direction \( D_i^k (\varphi_i) = (d_i^1, ..., d_i^{n-1}) \in \mathbb{R}^{n-1} \) which can be calculated from \( \varphi_i^k \) via a Polar to Cartesian coordinate transformation:

\[
d_i^k = \prod_{p=1}^{n-1} \cos(\varphi_p^k) \tag{1}
\]

\[
d_y^k = \sin(\varphi_{y(i-1)}^k) \cdot \prod_{p=1}^{n-1} \cos(\varphi_{y_p}^k) \tag{2}
\]

\[
d_{in}^k = \sin(\varphi_{i(n-1)}^k) \tag{3}
\]

In the GSO algorithm, the group consists of three individuals: producer, scroungers and rangers. At the \( k_{th} \) iteration the producer \( X_p \) behaves as follows:

1. The producer will scan at zero degree and then scan laterally by randomly sampling three points in the scanning field: one point at zero degree:

\[
X_z = X_p^k + r_{l_{\max}}^k D_p^k (\varphi^k) \tag{4}
\]

One point in the left hand side hypercube:

\[
X_l = X_p^k + r_{l_{\max}}^k D_p^k (\varphi^k - r_2 \theta_{\max} / 2) \tag{5}
\]

And one point in the right hand side hypercube:

\[
X_r = X_p^k + r_{l_{\max}}^k D_p^k (\varphi^k + r_2 \theta_{\max} / 2) \tag{6}
\]

where, \( r_1 \in \mathbb{R}^n \) is a normally distributed random number with mean 0 and standard deviation 1 and \( r_2 \in \mathbb{R}^{n-1} \) is a random sequence in the range (0, 1). \( \theta_{\max} \in \mathbb{R}^{n-1} \) is maximum pursuit.
angle. The maximum pursuit distance $l_{\text{max}}$ is calculated from:

$$l_{\text{max}} = \sum_{i=1}^{n} (U_i - L_i)^2$$

where, $L_i$ and $U_i$ are the lower and upper bounds for the $i_{th}$ dimension.

(2) The producer will then find the best point with the best resource (fitness value). If the best point has a better resource than its current position, then it will fly to this point. Or it will stay in its current position and turn its head to a new angle:

$$\phi^{k+1} = \phi^k + r_2 \alpha_{\text{max}}$$  \hspace{1cm} (7)

where, $\alpha_{\text{max}}$ is the maximum turning angle.

(3) If the producer cannot find a better area after $a$ iterations, it will turn its head back to zero degree:

$$\phi^{k+a} = \phi^k$$  \hspace{1cm} (8)

where, $a$ is a constant.

At the $k_{th}$ iteration, the area copying behavior of the $i_{th}$ scrounger can be modelled as a random walk towards the producer:

$$X_i^{k+1} = X_i^k + r_j (X_p^k - X_i^k)$$  \hspace{1cm} (9)

where, $r_j \in \mathbb{R}^n$ is a uniform random sequence in the range (0, 1).

Besides the producer and the scroungers, a small number of rangers have been also introduced into our GSO algorithm. Random walks, which are thought to be the most efficient searching method for randomly distributed resources, are employed by rangers. If the $i_{th}$ group member is selected as a ranger, at the $k_{th}$ iteration, firstly, it generates a random head angle $\phi_i$:

$$\phi^{k+1} = \phi^k + r_2 \alpha_{\text{max}}$$  \hspace{1cm} (10)

where, $\alpha_{\text{max}}$ is the maximum turning angle; and secondly, it chooses a random distance:

$$l_i = a \cdot l_{\text{max}}$$  \hspace{1cm} (11)

and move to the new point:

$$X_i^{k+1} = X_i^k + l_i D_i^k (\phi^{k+1})$$  \hspace{1cm} (12)
4. PARTICLE SWARM OPTIMIZER (PSO) ALGORITHM

The PSO has been inspired by the social behavior of animals such as fish schooling, insects swarming and birds flocking Kennedy and Eberhart [19]. It involves a number of particles, which are initialized randomly in the search space of an objective function. These particles are referred to as swarm. Each particle of the swarm represents a potential solution of the optimization problem. The particles fly through the search space and their positions are updated based on the best positions of individual particles for each iteration. The objective function is evaluated for each particle and the fitness values of particles are obtained to determine which position in the search space is the best Bergh and Engelbrecht [20] Using neighborhood with the guaranteed convergence PSO.

In each iteration, the swarm is updated using the following equations:

\[ V_{i}^{(k+1)} = \omega V_{i}^{(k)} + c_1 r_1(P_i^{(k)} - X_i^{(k)}) + c_2 r_2(gP_g^{(k)} - X_i^{(k)}) \]  
\[ X_{i}^{(k+1)} = X_i^{(k)} + V_{i}^{(k)} \]  

where \( X_i \) and \( V_i \) represent the current position and the velocity of the \( i_{th} \) particle, respectively; \( P_i \) is the best previous position of the \( i_{th} \) particle (called \( P_{best} \)) and \( gP_g \) is the best global position among all the particles in the swarm (called \( G_{best} \)); \( r_1 \) and \( r_2 \) are two uniform random sequences generated from \( U(0,1) \); and \( \omega \) is the inertia weight used to discount the previous velocity of the particle perseverance [21].

5. PARTICLE SWARM HYBRID GROUP SEARCH OPTIMIZER (PS-GSO)

The PS-GSO algorithm continues the classification model of PSO and GSO. Random search, angle search and step search are used at the same time. The PS-GSO is an improved PSO or GSO. The improvements are as follows: 1) Use the step search mechanism of PSO when the algorithm does not move forward. 2) Adopt angle search mechanism of GSO and step search mechanism PSO at the same time. 3) Deal with the boundary problem with the harmony search (HS) algorithm [22].

The realization of this hybrid optimization algorithm is as follows:

In an n-dimensional search space, the \( i_{th} \) member at the \( k_{th} \) searching bout (iteration) has a current position \( X_i^k \in \mathbb{R}^n \), the position of each member are initialized by random value before the start of the iterative search.

At the \( k_{th} \) searching iteration, calculate the fitness of each member, take the position of the best members as the producer and denoted it as \( X_p^k \), and scan at zero degree first time, then search for a new position in front, left and right directions, respectively, by Eqs. (4-6)
to seek a better position. The remaining members are selected as the scroungers by certain probability randomly, which will participate in search with random steps and follow the producer by Eq. (9). The remaining group members are selected as the rangers, which will select search angle and search distance randomly by Eq. (10) and Eq. (11), and then move to the new position by Eq. (12).

At the end of $k_{th}$ searching iteration, the algorithm will use PSO search mechanism if the producer does not update, meaning satisfy Eq. (15). Meanwhile, the producer will be considered as the best position $P_g^k$ of PSO algorithm. If the algorithm enters the PSO algorithm for the first time, consider the scroungers and rangers as personal best position $P_i^k$. Next time the personal best position $P_g^k$ will be selected by Eq. (17), which means the personal best position $P_i^{k-1}$ at previous iteration will be replaced by the better position $X_i^k$ at current iteration, then receive the latest personal best position $P_i^k$.

$$f(X_p^k) \geq f(X_p^{k-1})$$  \hspace{1cm} (15) \\
$$P_g^k = X_p^k$$  \hspace{1cm} (16) \\
$$P_i^k = M(X_i^k, P_i^{k-1})$$  \hspace{1cm} (17)

In the PSO search mechanism, the selection of particle inherits the group of GSO. The position and the velocity of each particle will be calculated by Eqs. (13 and 14). At the end of $k_{th}$ searching iteration, it will calculate the latest global best position $P_g^k$. If $f(P_g^k) < f(X_p^k)$, at next iteration, the producer will be replaced by the latest global best position $P_g^k$, as follow equation,

$$X_{p}^{k+1} = P_g^k$$  \hspace{1cm} (18)

After the $(k+1)_{th}$ iterations, if $X_i^d < X_i^{d(L)} (LowerBound)$ or $X_i^d > X_i^{d(U)} (UpperBound)$, the scalar $X_i^d$ is regenerated by selecting the corresponding component of the vector from $p_{best}$ swarm randomly, which can be described as follows:

$$X_i^d = (P_j)^d_t, \quad t = \text{int}(rand(1,n))$$  \hspace{1cm} (19)

where $(P_t)^d$ denotes the $d_{th}$ dimension scalar of $p_{best}$ swarm of the $t_{th}$ Particle, $t$ denotes a random integer number and $n$ denotes the swarm sizes. The pseudo code for structural optimization by PS-GSO is listed in Table 1.
Table 1. Pseudo code for structural optimization by PS-GSO algorithm

Set k = 0;
Randomly initialize positions $X_i$, the velocity $V_i$ and head angles $\phi_i$ of all members;
FOR (each member $i$ in the group)
WHILE (the constraints are violated)
Randomly re-generate the current member $X_i$
END WHILE
END FOR
WHILE (the termination conditions are not met)
FOR (each members $i$ in the group)
Calculate fitness: Calculate the fitness value of current member: $f(X_i)$
Choose producer: Find the producer $X_p$ of the group;
The producer, scroungers and rangers update their position by the equations of GSO algorithm.
Calculate the fitness of each individual; update the producer and the search angle.
Check whether the algorithm move forward. If not, enter the PSO algorithm. If it does, skip the PSO algorithm.
PSO algorithm: Update the best position $P_{sg}$ and the personal best position $P_i$ by Eq. (13) and Eq. (14);
Update the position and velocity via equations of PSO algorithm.
Check whether each member of the group violates the variable boundary. If it does, calculated by Eq. (19).
Calculate the fitness value of current member, and then update the best position $P_{sg}$ (the producer $X_p$) and the personal best position $P_i$.
END FOR
Set k = k + 1;
END WHILE

6. PS-GSO ALGORITHM WITH DISCRETE VARIABLES

When using the PS-GSO algorithm in optimizing with discrete variables, because of the areas of cross-sections aren’t continuum, before optimization there will be a mapping function which makes the discrete section areas correspond to the continuum integer from small to large. Suppose a discrete set $A_n$ with $n$ discrete variables, after arranging from small to large:

$$A_n = \{X_1, X_2, \ldots, X_j, \ldots X_n\}, 1 \leq j \leq n$$

Employ a mapping function to replace the discrete values of $A_n$ with its serial numbers like:

$$h(j) = X_j$$

The serial numbers replace the discrete values to keep searching with continuum values and avoid efficiency of search declining. Suppose that there are $p$ members the search space
with D dimension. And the position of the \(i\)th member is denoted with vector \(x_i\) like:

\[
x_i = (x_{i1}^1, x_{i2}^2, \ldots, x_{id}^d, \ldots, x_{ip}^p), 1 \leq d \leq D, i = 1, 2, \ldots, p
\]

in which, \(x_{id}^d \in \{1, 2, \ldots, j, \ldots, n\}\) corresponds to the discrete variables \(\{X_1, X_2, \ldots, X_j, \ldots, X_n\}\) by mapping function \(h(j)\). After that, all of the members search in the continuum space which is the integer space, as each component of vector \(x_i\) is integer. Accordingly, the producer, scroungers, rangers and particles become:

\[
X_i = \text{floor}(X_i)
\]

in which floor is a function rounding to negative infinity. After that, there is no change with the objective function and constraints, which just before being substitution into, the iterated integer are turn into areas of cross-sections correspondingly by the mapping function.

7. NUMERICAL EXAMPLES

In this section, two spatial truss structures commonly used in literature are selected as benchmark structures to test the PS-GSO. A double layer reticulated shell structure is chosen to see the efficiency of the hybrid PS-GSO algorithm. For all the three algorithms, the population size is set to at 50. The inertia weight \(\omega\) decrease linearly from 0.9 to 0.4; and the value of acceleration constants \(c_1\) and \(c_2\) are set to be the same and equal to 0.8. For the GSO algorithm, 20% of the population is selected as rangers; the initial head angle \(\phi_0\) of each individual is set to be \(\pi / 4\). The constant \(a\) is given by round \(\sqrt{n + 1}\). The maximum pursuit angle \(\theta_{\text{max}}\) is \(\pi / a^2\). The maximum turning angle \(\alpha_{\text{max}}\) is set to be \(\pi / 2a^2\). For the new hybrid algorithm (PS-GSO), the parameters are set to be the same as GSO and HPSO, except that the inertia weight \(\omega\) is set to be 0.9; and the value of acceleration constants \(c_1\) and \(c_2\) are set to be the same and equal to 2. Different iteration numbers are used for different optimization structures, with smaller iteration number for smaller variable number structures and larger one for large variable number structures.

7.1. Example 1: The 25-bar spatial truss structure

The 25-bar spatial truss structure is shown in Figure 1. The material density is 0.1 lb/in\(^3\) and the modulus of elasticity is 10,000 ksi. The stress limits of the members are \(\pm 40,000\) psi. All nodes in all directions are subjected to the displacement limits of \(\pm 0.35\) in. Load case is listed in Table 2. There are 25 members, which are divided into 8 groups, as follows: (1) \(A_1\), (2) \(A_2\sim A_5\), (3) \(A_6\sim A_{10}\), (4) \(A_{10}\sim A_{11}\), (5) \(A_{12}\sim A_{13}\), (6) \(A_{14}\sim A_{17}\), (7) \(A_{18}\sim A_{21}\) and (8) \(A_{22}\sim A_{25}\). The discrete variables are selected from the set \(D = \{0.01, 0.4, 0.8, 1.2, 1.6, 2.0, 2.4, 2.8, 3.2, 3.6, 4.0, 4.4, 4.8, 5.2, 5.6, 6.0\}\) (in\(^2\)). The maximum number of iterations is 500.
Figure 1. The 25-bar spatial truss structure

Table 2. Load cases for the 25-bar spatial truss structure

<table>
<thead>
<tr>
<th>Node</th>
<th>Load case</th>
<th>( P_X ) (kips)</th>
<th>( P_Y ) (kips)</th>
<th>( P_Z ) (kips)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>0.0</td>
<td>20.0</td>
<td>−5.0</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0.0</td>
<td>−20.0</td>
<td>−5.0</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Figure 2. Comparison of convergence rates for the 25-bar truss structure

Table 3 shows the optimization solutions of 25-bar truss structure. Figure 2 gives the convergence rate of the three algorithms. The three algorithms can achieve the optimal solution after 500 iterations. But it can be seen from Figure 2 that it takes about 160 and 400 iterations for the GSO and HPSO algorithms to converge respectively, while the PS-GSO algorithm only needs less than 50 iterations.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Optimal cross-sectional areas (in(^2))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PS-GSO</td>
</tr>
<tr>
<td>1 A(_1)</td>
<td>0.01</td>
</tr>
<tr>
<td>2 A(_2)~A(_5)</td>
<td>2.00</td>
</tr>
<tr>
<td>3 A(_6)~A(_9)</td>
<td>3.60</td>
</tr>
<tr>
<td>4 A(<em>{10})~A(</em>{11})</td>
<td>0.01</td>
</tr>
<tr>
<td>5 A(<em>{12})~A(</em>{13})</td>
<td>0.01</td>
</tr>
<tr>
<td>6 A(<em>{14})~A(</em>{17})</td>
<td>0.80</td>
</tr>
<tr>
<td>7 A(<em>{18})~A(</em>{21})</td>
<td>1.60</td>
</tr>
<tr>
<td>8 A(<em>{22})~A(</em>{25})</td>
<td>2.40</td>
</tr>
<tr>
<td>Weight (lb)</td>
<td>560.59</td>
</tr>
</tbody>
</table>

7.2. Example 2: The 72-bar spatial truss structure

The 72-bar spatial truss structure is shown in Figure 3. The material density is 0.1 lb/in\(^3\) and the modulus of elasticity is 10\(^7\)psi. The stress limits of all the members are ±25ksi. All nodes in all directions are subjected to the displacement limits of ±0.25 in. The load cases are listed in Table 4. There are 72 bars, which are divided into 16 groups, as follows: (1) A\(_1\)~A\(_4\), (2) A\(_5\)~A\(_{12}\), (3) A\(_{13}\)~A\(_{16}\), (4) A\(_{17}\)~A\(_{18}\), (5) A\(_{19}\)~A\(_{22}\), (6) A\(_{23}\)~A\(_{30}\), (7) A\(_{31}\)~A\(_{34}\), (8) A\(_{35}\)~A\(_{36}\), (9) A\(_{37}\)~A\(_{40}\), (10) A\(_{41}\)~A\(_{48}\), (11) A\(_{49}\)~A\(_{52}\), (12) A\(_{53}\)~A\(_{54}\), (13) A\(_{55}\)~A\(_{58}\), (14) A\(_{59}\)~A\(_{66}\), (15) A\(_{67}\)~A\(_{70}\), (16) A\(_{71}\)~A\(_{72}\). The optional discrete variables are: D={0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2}. The maximum number of iterations is 1000.

<table>
<thead>
<tr>
<th>Node</th>
<th>Load cases</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P(_X)(kips)</td>
</tr>
<tr>
<td>17</td>
<td>5.0</td>
</tr>
<tr>
<td>18</td>
<td>0.0</td>
</tr>
<tr>
<td>19</td>
<td>0.0</td>
</tr>
<tr>
<td>20</td>
<td>0.0</td>
</tr>
</tbody>
</table>
It can be seen from Table 5 that the PS-GSO algorithms achieve the best solution after 1000 iterations, its result is statistically better than that of GSO and HPSO. Moreover, it can be seen from Figure 4 that the PS-GSO algorithm provides best convergence rates than that of the GSO and HPSO.
Table 5. Design results for the 72-bar truss structure

<table>
<thead>
<tr>
<th>Variables</th>
<th>Optimal cross-sectional areas (in²)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PS-GSO</td>
</tr>
<tr>
<td>1 A₁~A₄</td>
<td>2.1</td>
</tr>
<tr>
<td>2 A₅~A₁₂</td>
<td>0.5</td>
</tr>
<tr>
<td>3 A₁₃~A₁₆</td>
<td>0.1</td>
</tr>
<tr>
<td>4 A₁₇~A₁₈</td>
<td>0.1</td>
</tr>
<tr>
<td>5 A₁₉~A₂₂</td>
<td>1.5</td>
</tr>
<tr>
<td>6 A₂₃~A₃₀</td>
<td>0.5</td>
</tr>
<tr>
<td>7 A₃₁~A₃₄</td>
<td>0.1</td>
</tr>
<tr>
<td>8 A₃₅~A₃₆</td>
<td>0.1</td>
</tr>
<tr>
<td>9 A₃₇~A₄₀</td>
<td>0.5</td>
</tr>
<tr>
<td>10 A₄₁~A₄₈</td>
<td>0.5</td>
</tr>
<tr>
<td>11 A₄₉~A₅₂</td>
<td>0.1</td>
</tr>
<tr>
<td>12 A₅₃~A₅₄</td>
<td>0.1</td>
</tr>
<tr>
<td>13 A₅₅~A₅₈</td>
<td>0.2</td>
</tr>
<tr>
<td>14 A₅₉~A₆₆</td>
<td>0.5</td>
</tr>
<tr>
<td>15 A₆₇~A₇₀</td>
<td>0.5</td>
</tr>
<tr>
<td>16 A₇₁~A₇₂</td>
<td>0.6</td>
</tr>
<tr>
<td>Weight (lb)</td>
<td>386.81</td>
</tr>
</tbody>
</table>

7.2. Example 3: A double layer spherical shell

There are 6761 nodes and 1834 bars in the double layer shell structure and it shows in Figure 5. All bars are divided into three groups which are upper chords, lower chords and web members. The span of the shell is 83.6 meters long and the rise is 14 meters high. The layer highness between the upper and the lower chord is 1.5 meters. The shell structure is made of steel. And the elasticity modulus is 210 GPa, the density 7850 kg/m³. The periphery nodes of the lower chord layer are joined with hinges.

The target is the weight of the structure. The areas of cross-sections of bar members are selected as the design variables. Furthermore, all of the members are thin-walled circular bar which are selected from national standard GB8162-87 of China with 379 kinds of cross-sections. The structure bears a single load case. Each node of upper chord layer bears a
vertical downward load of 50kN. All the nodes in all directions are subjected to the displacement limits of ±0.209m which is the span of 1/400. Every bar is subjected to the stress limits of ±215MPa. Meanwhile, the restriction of stability of bar members is considered in accordance with criterion. The maximum allowable slenderness ratio of pressure and pull bar is 180 and 300 respectively. The maximum number of iteration is limited to 200.

Table 6 shows the optimization solutions of double layer spherical shell. Figure 6 gives the convergence rate of the three algorithms. The three algorithms can achieve the optimal solution after 200 iterations. But it can be seen from Figure 6 that it takes about 130 and 100 iterations for the GSO and HPSO algorithms to converge respectively, while the PS-GSO algorithm only needs less than 15 iterations. It can be seen from Table 6 that the PS-GSO algorithms achieve the lightest structure, its result is statistically better than that of GSO and HPSO. Moreover, it can be seen from Figure 6 that the PS-GSO algorithm provides best convergence rates than those of the GSO and HPSO.
8. CONCLUSIONS

In this paper, a new hybrid optimization algorithm named particle swarm-group search optimization (PS-GSO) handling discrete variables is presented based on the GSO, PSO and harmony search method. The PS-GSO algorithm for discrete variables has all the advantages that belong to the PS-GSO algorithm for continuous variables, and has faster convergence rate than the GSO and HPSO algorithms for discrete variables. It is the most efficient one of these three algorithms. The PS-GSO algorithm is tested by three spatial truss structures optimization problems. All the results show that the PS-GSO algorithm converges much quickly than the HPSO and the GSO. It is proved that the PS-GSO can be used for optimum problems with discrete variables efficiently. Compared with GSO and HPSO algorithm, the PS-GSO algorithm achieves convergent results during the early iterations, which illustrates the rapidly converging to the optimal solution by using the PS-GSO algorithm.

Acknowledgements: The Corresponding author, Prof. Lijuan Li would like to thank the financially supporting by the National Natural Science Foundation of China (project numbers: 10772052, 51178121) and the Natural Science Foundation of Guangdong Province. (Project numbers: 8151009001000042, 9151009001000059).

REFERENCES


