Struct Multidisc Optim (2010) 41:179–187
DOI 10.1007/s00158-009-0417-x

RESEARCH PAPER

Ply stacking sequence optimization of composite laminate by permutation discrete particle swarm optimization

Nan Chang · Wei Wang · Wei Yang · Jian Wang

Received: 24 May 2008 / Revised: 26 April 2009 / Accepted: 5 July 2009 / Published online: 7 August 2009

Abstract Stacking sequence optimization (SSO) of laminate will greatly improve its mechanical properties without weight penalty. In this paper, a novel permutation discrete particle swarm optimization (PDPSO) method was proposed to perform SSO. To improve the efficiency of the algorithm, the concepts and techniques of valid/invalid exchange, checking memory and Self-escape were introduced into the PDPSO. In total 11 examples were presented. First, eight examples were carried out by employing the proposed method. The results show that the computational efficiency of PDPSO is greatly improved compared with standard discrete particle swarm optimization (SDPSO), and is comparable with that of gene rank crossover (GR) and partially mapped crossover (PMX). Then, three extra examples were presented, in which the outermost plies in the optimum design are not ±45° plies. The results show that the PDPSO has better stability and potential which demonstrate the better performance of PDPSO for laminates.

Keywords Composite · Stacking sequence · PSO · Self-escape idea · Valid/invalid exchange

N. Chang · W. Wang
School of Aeronautics, Northwestern Polytechnical University, Xi'an 710072, China

W. Yang
Chengdu Aircraft Design & Research Institute, Chengdu 610041, China

J. Wang (✉)
School of Aerospace Engineering and Mechanical, Queen's University of Belfast, Belfast, BT7 1NN, UK
e-mail: j.wang@qub.ac.uk

1 Introduction

Composite materials are preferred in aircraft structures because of their high strength to weight ratio, high stiffness to weight ratio and other designable properties compared with metals. As composites are increasingly used in next generation aircraft, optimization technique becomes more and more important in the design of laminate because of the complexity and enormous design space. Thickness of laminates, fibber orientation angles and stacking sequence are usually taken as common design variables in the design of the laminates. For the composite wing box optimization problem, the global–local optimization strategy was adopted by many researchers (Faggiani and Falzon 2007; Herencia et al. 2007; Seresta et al. 2006). However, the stacking sequence optimization was always conducted independently because it is a discrete combinatorial optimization problem.

Composite laminates consist of layers of one or more materials stacked at different orientations. The layer thickness for each material is usually fixed and fibre orientation angles are often limited to discrete values set as 0, ±45, and 90° due to manufacturing constraints. Hence, the stacking sequence optimization should be treated as a discrete optimization problem to which the conventional optimisation methodologies are difficult to apply. Intelligent optimization methods, for example, genetic algorithm (GA), have been applied to composite stacking sequence optimization problem because of its excellent ability and better chances to find global optima (Park et al. 2001; Sciuva et al. 2003; Muc and Gurba 2001; Walker and Smith 2003; Liu et al. 2000a; Kameyama and Fukunaga 2007). GA is an evolutionary algorithm based on Darwin's principle of survival of the fittest and mimics the process of natural selection. Nowadays, the particle swarm optimization (PSO) which is also an evolutionary algorithm has become more and more popular. It

was first proposed by Kennedy and Eberhart (Eberhart and Kennedy 1995; Shi and Eberhart 1998) and could be used to carry out continuous global optimization problem with non-linear objective functions. PSO algorithm is based on a simplified social model and it mimics the behaviour of a group of fish or birds in searching for food. PSO has been successfully applied to some engineering and structural optimization problems (Fourie and Groenwold 2001; Venter and Sobieszczanski-Sobieski 2004). PSO is similar to GA in some aspects. For example, both PSO and GA start with a randomly generated population, evaluate the population by fitness values, update the population and use random methods to search for the optimum. Although neither of the two methods guarantees an optimal result, they are both capable of finding the "global minimum" by jumping across the given design space. In addition, PSO may obtain the same high quality solution as a GA at lower computational cost (Suresh et al. 2007). Very recently, PSO has been used in composite optimization. Mark W. Bloomfield, J. Enrique Herencia and Paul M. Weaver developed a two-level optimization approach for the composite optimization problem, in which a particle swarm optimization algorithm is used at the second level to determine laminate stacking sequences (Bloomfield et al. 2008).

To solve the discrete combinatorial optimization problem, a new PSO named discrete particle swarm optimization (DPSO) (Kennedy and Eberhart 1997; Rameshkumar et al. 2005) has been employed to deal with the typical travelling salesman problems (TSP) (Li et al. 2006). However, stacking sequence optimization (SSO) is not equivalent to TSP. For example, the exchange between two plies with the same fibber orientation will not change the properties of laminated panels. Hence, if we handle the SSO problem in the same way with TSP, there will be a large amount of unnecessary evaluation or simulation, thus expensive cost in computational time will occur, especially for thicker laminates.

Although the genetic algorithm (GA) could be used to solve this optimisation problem, GA has drawbacks of complex operation procedures and low efficiency. Only if certain complicated repair strategies are implemented, can the efficiency of GA be improved. In this paper, As the difference between SSO and TSP mentioned above, the definitions of both valid and invalid exchanges are introduced in DPSO. If the exchange occurs between two plies with different fibber orientations, the exchange is valid. Then, the result of exchange is counted as a new design and will be evaluated. Otherwise, when the exchange happens between two plies with the same fibber angles it is invalid, and this design is discarded. Furthermore, the concepts of self-escape and memory checking are introduced to the permutation DPSO in this paper to prevent premature convergence and improve the ability of global searching. To validate the algorithm, 11 cases under different loading cases are calculated to achieve the maximum combined buckling load.

2 Problem formulation

In laminated composite design, ply orientations are generally restricted to 0, 90, ±45°. In this paper, the stacking sequence of an orthotropic laminate was optimized for maximum buckling load. The numbers of plies with 0_, ±45_, and 90_° respectively in the laminate were fixed. And the laminate was assumed to be symmetric and balanced. Furthermore, more than four contiguous plies with the same orientation were not allowed. Take the numerical example in Liu et al. (2000b), an unstiffened, simply supported, laminated panel with dimension a × b (Fig. 1) is subjected to normal loads per unit length N_x and N_y, and a shear load per unit length N_{xy}. The material of the plies is graphite-epoxy.

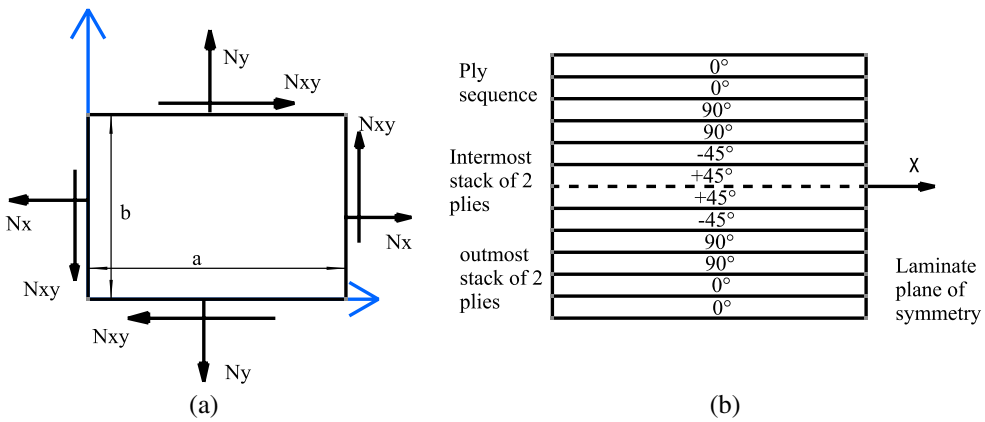

Fig. 1 Laminate plate geometry and load; **a** laminate plate geometry and applied loading; **b** ply stacking sequence

Under biaxial loading, the laminate may buckle into m and n half waves in the x and y directions, respectively. And the buckling factor $\lambda_n^{(m,n)}$ is given by

$$\frac{\lambda_n^{(m,n)}}{\pi^2} = \frac{D_{11}(m/a)^4+2(D_{12}+2D_{66})(m/a)^2(n/b)^2+D_{22}(n/b)^4}{(m/a)^2N_x+(n/b)^2N_y} \tag{1}$$

The smallest value of $\lambda_n^{(m,n)}$, which is called the critical buckling load factor, is resulted from a given combination of a pair of (m, n). Under shear loading, modelling of this buckling mode for a finite plate is computationally expensive. Therefore, the plate is assumed to have an infinite length, and the analytical solutions are used as approximations (Whitney 1985). The critical shear buckling load factor is given by

$$\lambda_s = \begin{cases} \dfrac{4\beta\left(D_{11}D_{22}^3\right)^{1/4}}{b^2N_{xy}} & 1 \le \Gamma \le \infty \\ \dfrac{4\beta_1\sqrt{D_{22}(D_{12}+2D_{66})}}{b^2N_{xy}} & 0 \le \Gamma \le 1 \end{cases} \tag{2}$$

And the variable Γ can be expressed as

$$\Gamma = \frac{\sqrt{D_{11}D_{22}}}{D_{12}+2D_{66}} \tag{3}$$

The values of β_1 are given in Table 1.

When normal and shear loads are applied simultaneously to the panel, the combined critical buckling load factor is approximated by

$$\frac{1}{\lambda_c^{(m,n)}} = \frac{1}{\lambda_n^{(m,n)}} + \frac{1}{\lambda_s^2} \tag{4}$$

Table 1 Coefficient β_1 for shear buckling load factor (Liu et al. 2000b)

Γ	β_1
0.0	11.71
0.2	11.80
0.5	12.20
1.0	13.17
2.0	10.80
3.0	9.95
5.0	9.25
10.0	8.70
20.0	8.40
40.0	8.25
∞	8.13

From the above equation, the combined buckling load factor $\lambda_c^{(m,n)}$ is always more critical than the normal buckling load factor $\lambda_n^{(m,n)}$. Thus, the buckling load factor λ is the smallest load factors as shown in (5).

$$\lambda = \min\left\{|\lambda_s|, \lambda_c^{(m,n)}\right\} \tag{5}$$

In this paper, we will focus on solving the stacking sequence optimization (SSO) problem by using a permutation DPSO algorithm. As explained above, the objective of the optimization is to maximize the critical buckling load of the laminated panel. Additionally, to reduce chances of matrix cracking, we do not allow more than four contiguous plies with the same orientation. This is the so called contiguity constraint. The objective function is equal to maximizing the failure load factor λ penalized by violations of the limitation of contiguity constraint. So the objective function to be maximized is given as (Li et al. 2006)

$$\Phi = \frac{\lambda}{p_{cont}^{N_{cont}}} \tag{6}$$

P_{cont} is penalty parameter(set to 1.05 here) for violation of the contiguity constraint. N_{cont} is total number of contiguous plies in excess of four. To improve convergence and reduce permutation operations, we assume that the laminate is composed of pairs of 0° plies, 90° plies, or ±45° plies. It should be noted that the contiguity constraint just applied to 0 and 90° plies only. The ±45° plies alternate between +45 and −45°, so they do not have any contiguity problem. Meanwhile, the ±45° plies always appear in the laminate together as a ±45° pair, so the balanced constraint is satisfied automatically. The symmetric constraint is solved implicitly because only half laminate stacking sequence is used as design variables.

3 Permutation discrete particle swarm optimization

In this section, the detailed procedures of permutation discrete particle swarm optimization (PDPSO) for solving the SSO problem of laminate are presented. Examples are provided for better understanding. A standard PSO mimics the behaviour of a bird flock in searching for food. Each particle updates velocity and position value via the memory and clue from the flock. The basic steps in the PSO algorithm are as follows (Kathiravan and Ganguli 2007):

1. Initialize the swarm with random position values and random initial velocities.
2. Determine the velocity vector for each particle in the swarm using the knowledge of the best position attained by each particle, the swarm as a whole, and the previous position of each particle in the swarm.

3. Modify the current position of each particle using the velocity vector and the previous position of each particle.
4. Repeat from step 2 until the stop criterion is satisfied.

The velocity vector and position value of each particle can be expressed as:

$$\left.\begin{aligned} \boldsymbol{V}_k^i &= w\boldsymbol{V}_{k-1}^i + c_1 r_1 \left(\boldsymbol{Pbest}^i - \boldsymbol{X}_{k-1}^i\right) \\ &\quad + c_2 r_2 \left(\boldsymbol{Gbest}_{k-1}^g - \boldsymbol{X}_{k-1}^i\right) \\ \boldsymbol{X}_k^i &= \boldsymbol{X}_{k-1}^i + \boldsymbol{V}_k^i \end{aligned}\right\} \tag{7}$$

where the superscript $\boldsymbol{i}$ denotes the particle and the subscript k denotes the iteration number; $\boldsymbol{V}$ denotes the velocity and $\boldsymbol{X}$ denotes the position; r_1 and r_2 are uniformly distributed random numbers in the interval [0,1]; c_1 and c_2 are the acceleration constants; w is the inertia weight; $\boldsymbol{Pbest}^i$ is the best position attained by the particle i in the swarm so far and $\boldsymbol{Gbest}_{k-1}^g$ is the global best position attained by the swarm at iteration $k-1$.

3.1 Calculating rules of permutation discrete particle swarm optimization

As the design variables are discrete, the calculating rules should be revised. A detailed description of rules that should be used in (7) is stated as follows:

(1) Definition of particle's position

The position of particles, which represents the stacking sequence of the laminate (upper to mid-plane), is noted by a vector $\boldsymbol{X}_{k-1}^i$, shown in (8).

$$\begin{aligned} &\boldsymbol{X}_{k-1}^i = \left(x_{k-1,1}^i, x_{k-1,2}^i, \cdots, x_{k-1,j}^i \cdots x_{k-1,N}^i\right), \\ &1 \le j \le N, 1 \le x_j \le N \end{aligned} \tag{8}$$

where N is number of plies in the half laminate; $x_{k-1,j}^i$ is the integer code representing the fibre orientation angle of the j^{th} ply.

(2) Subtraction of particle's position

$\boldsymbol{X}_{k-1}^q$ subtracts from $\boldsymbol{X}_{k-1}^p$ can be stated as

$$\begin{aligned} \Delta\boldsymbol{X} &= \boldsymbol{X}_{k-1}^p - \boldsymbol{X}_{k-1}^q \\ &= \left(x_{k-1,1}^p, x_{k-1,2}^p, \cdots, x_{k-1,j}^p \cdots x_{k-1,N}^p\right) \\ &\quad - \left(x_{k-1,1}^q, x_{k-1,2}^q, \cdots, x_{k-1,j}^q \cdots x_{k-1,j}^q\right) \end{aligned} \tag{9}$$

The value of ΔX can be calculated element by element as follows

$$\Delta x_j = x_{k-1,j}^p - x_{k-1,j}^q = \begin{cases} 0 & if \quad x_{k-1,j}^p = x_{k-1,j}^q \\ x_{k-1,j}^p & otherwise \end{cases} \tag{10}$$

$x_{k-1,j}^p$ and $x_{k-1,j}^q$ are the j^{th} element of $\boldsymbol{X}_{k-1}^p$ and $\boldsymbol{X}_{k-1}^q$, respectively. So $\left(\boldsymbol{Pbest}^i - \boldsymbol{X}_{k-1}^i\right)$ and $\left(\boldsymbol{Gbest}_{k-1}^g - \boldsymbol{X}_{k-1}^i\right)$ can be evaluated by (10).

(3) Definition of particle's velocity

The velocity of a particle is used to adjust the particle position. The velocity is given by

$$\boldsymbol{V}_{k-1} = \left(\boldsymbol{X}_{k-1}^p - \boldsymbol{X}_{k-1}^q\right)/\Delta t = \Delta\boldsymbol{X}/\Delta t \tag{11}$$

Each element value of V_{k-1} can be calculated by

$$\begin{aligned} v_{k-1,j} &= \left(x_{k-1,j}^p - x_{k-1,j}^q\right)/ \\ \Delta t &= \begin{cases} 0 & if \quad x_{k-1,j}^p = x_{k-1,j}^q \\ x_{k-1,j}^p & otherwise \end{cases} \end{aligned} \tag{12}$$

where Δt denotes the time interval. So velocity = distance (space)/time. Note that here $\Delta t = 1$ and it is used to provide physical meaning. $\boldsymbol{v}_{k-1,j}$ is the value of j^{th} element in vector $\boldsymbol{V}_{k-1}$.

(4) Procedures for calculating the velocity

Multiply velocity by a constant is given as

$$\boldsymbol{V}'_{k-1} = C \cdot \boldsymbol{V}_{k-1}\, C \in [0, 1] \tag{13}$$

where C is user-defined constant (just like the c_1 or c_2 in (7)). $\boldsymbol{V}'_{k-1}$ is the result of velocity $\boldsymbol{V}_{k-1}$ multiply a constant C. An array of random numbers $rand1(j)$, where $j = 1..N$, corresponding to each element in vector $\boldsymbol{V}_{k-1}$ should be randomly generated. Note that the random numbers $rand1(j)$ should be confined within interval [0, 1]. Here $rand1(j)$ just like r_1 and r_2 in (7).

The value of each element in $\boldsymbol{V}'_{k-1}$is given by the following equation:

$$v'_{k-1,j} = \begin{cases} v_{k-1,j} & rand1\,(j) \ge C \\ 0 & otherwise \end{cases} \tag{14}$$

In (7), we can find the sum of velocities was denoted as the next generation velocity of the particle. Therefore, the sum of two velocities can be stated as:

$$\boldsymbol{V}_k = \boldsymbol{V1}_{k-1} + \boldsymbol{V2}_{k-1} \tag{15}$$

The value of each element in V_k is defined as shown in (16)

$$v_{k,j} = \begin{cases} v1_{k-1,j} & rand2\,(j) > 0.5 \\ v2_{k-1,j} & otherwise \end{cases} \tag{16}$$

where $\boldsymbol{V1}_{k-1}$ and $\boldsymbol{V2}_{k-1}$ represent two different velocities. The subscript j denotes the j^{th} element of velocity. $rand2(j)(j = 1..N)$ is another array of random number corresponding to each component of the velocity. 0.5 is the threshold of $rand2(j)$. When $rand2(j)$ is great than 0.5, $\boldsymbol{v}_{k,j}$ is set to equal to $\boldsymbol{v1}_{k-1,j}$, otherwise, $\boldsymbol{v}_{k,j}$ is set to equal to $\boldsymbol{v2}_{k-1,j}$. The $\boldsymbol{V}_k^i$ in (7) can be calculated using above (9–16).

(5) Procedures for updating position

The new position can be determined by

$$\boldsymbol{X}_k^i = \boldsymbol{X}_{k-1}^i + \boldsymbol{V}_k^i \tag{17}$$

If $v_{k,j}^i = 0$, $x_{k,j}^i = x_{k-1,j}^i$; otherwise, search $x_{k-1,l}^i$ in $\boldsymbol{X}_{k-1}^i$, which satisfy $x_{k-1,l}^i = v_{k,j}^i$, then swap $x_{k-1,l}^i$ and $x_{k-1,j}^i$ to form the $\boldsymbol{X}_k^i$.. The $\boldsymbol{X}_k^i$ in (7) can be calculated by using (17). And the detailed process of evaluating new position $\boldsymbol{X}_k^i$ is illustrated in the Fig. 2. Therefore, the discrete velocity and position vector in (8) are all obtained by following abovementioned procedures.

To improve convergence and reduce the number of permutations, we assume that the laminate is composed of pairs of 0° plies, 90° plies, or ±45° plies. If the 0_2, ±45, and 90_2° plies group are used directly as code in the DPSO, it will tend to produce chaos when we interchange the position of elements of particles in (16). To overcome this problem, the permutation coding is represented by a list of distinct integers, such as 1, 2, 3…, coding the orderings of 0_2, ±45, and 90_2 stacks referenced to baseline laminate. As explained in Liu et al. (2000b), we selected the baseline laminate to have all the specified 90° stacks on the outside, followed by the ±45° plies and then the 0° stacks. For example, the stacks $[(90_2)_5/(\pm 45)_3/(0_2)_4]_s$ is described as

Encoding :	1	2	3	4	5	6	7
	8	9	10	11	12		
Decoding :	90_2	90_2	90_2	90_2	90_2	±45	±45
	±45	0_2	0_2	0_2	02		

To explain the evaluation process more clearly, an example is given below

$$\boldsymbol{X}_{k-1}^i = [1/2/4/9/8/3/5/7/6];$$
$$\boldsymbol{Gbest}_{k-1}^g = [4/3/6/9/8/5/2/7/1];$$
$$\boldsymbol{Pbest}^i = [4/2/3/9/8/6/5/1/7];$$
$$C1 = 0.6, \quad c2 = 0.4;$$

The main procedure of the (7) can be given as follows:

$$\boldsymbol{Pbest}^i - \boldsymbol{X}_{k-1}^i = [4/0/3/0/0/6/0/1/7]$$
$$\boldsymbol{Gbest}_{k-1}^g - \boldsymbol{X}_{k-1}^i = [4/3/6/0/0/5/2/0/1]$$
$$\begin{aligned} \boldsymbol{V}_k^i &= c_1\left(\boldsymbol{Pbest}^i - \boldsymbol{X}_{k-1}^i\right) + c_2\left(\boldsymbol{Gbest}_{k-1}^g - \boldsymbol{X}_{k-1}^i\right) \\ &= [4/0/0/0/0/0/0/1/7] + [0/3/6/0/0/0/2/0/1] \\ &= [4/0/6/0/0/0/2/1/1] \end{aligned}$$
$$\begin{aligned} \boldsymbol{X}_k^i &= \boldsymbol{X}_{k-1}^i + \boldsymbol{V}_k^i = [1/2/4/9/8/3/5/7/6] \\ &\quad + [4/0/6/0/0/0/2/1/1] \end{aligned}$$

The process of calculating new position can be described by Fig. 2.

From the above figure, we can see the first integer code of velocity $\boldsymbol{V}_{k,1}^i = 4$ in step 1. We find the third integer code $\boldsymbol{X}_{k-1,3}^i = \boldsymbol{V}_{k,1}^i = 4$, therefore, we swap them and the changes are highlighted by using bold text. Other steps are same as in the step 1.

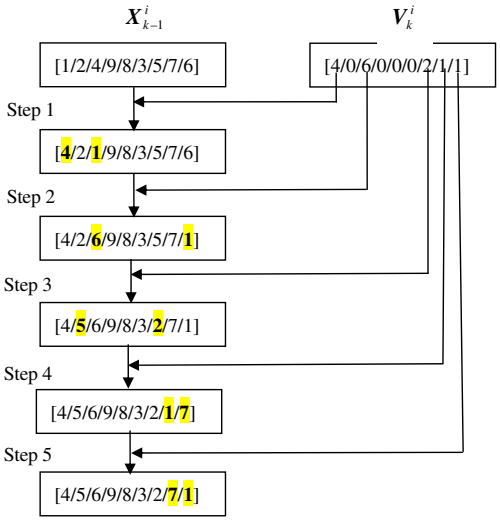

Fig. 2 Process of calculating new position

3.2 Ratio of valid permutation

In this section, the ratio of valid permutations is defined as the number of valid permutations in the laminate divided the total number of permutations of its integer code. We will take a 24-layer laminate as an example to calculate the ratio of valid permutation in the SSO .Only 12 plies will be considered for a symmetrical problem.

```
Begin
 while
  Generate a new design string;
  Search for the given design in the memory;
  if found
    Renew a design string;
    else
    Add the design string into memory;
    Break;
  end if
 end while
end
```

Fig. 3 Pseudocode of checking memory

Theoretically, the number of valid permutations of the stack [0/0/0/45/45/45/−45/−45/−45/90/90/90], in which the plies of 45 and −45° are treated separately, can be given as

$$C_{12}^{3} \times C_{4}^{1} \times C_{9}^{3} \times C_{3}^{1} \times C_{6}^{3} \times C_{2}^{1} \times C_{3}^{3} \times C_{1}^{1} = 8870400 \quad (18)$$

Similarly, the number of total permutation of the encoding [1/2/3/4/5/6/7/8/9/10/11/12] can be given as

$$P_{12}^{12} = 479001600 \quad (19)$$

The ratio of valid permutations is stated as

$$8870400/479001600 = 1.85\% \quad (20)$$

From the above example, it is clear that there is a great amount of invalid exchanges in the stacking sequence optimization problem. Therefore, the evaluation of exchanges is of rather importance, which could avoid unnecessary analyses and thus improve the computational efficiency significantly.

3.3 Concepts of checking memory and self-escape idea

The validity of exchanges is finally evaluated by memory checking. The generated stacking permutations in the optimization process are stored in the memory. The memory should be checked whenever a new design string is created. If the design is not found in the memory, this design string is taken to be an eligible particle. And it is then stored into the memory. Otherwise, the programming should regenerate a design string. The procedure can be stated as in (Fig. 3).

It is well known that the particles will generally move to a local area in searching space in standard PSO. The proposed DPSO has the same characteristic. As a result, the optimization algorithm will prematurely converge to a local optimum. To solve this problem, the most direct and effective approach is to define a variable *CI* which denotes the dispersed grade of particles. If the $CI < \varepsilon$, we should expand the searching space and regenerate the particles. In this paper, the *CI* is defined as the arithmetic average of differences between best position attained by particle and global best position in the whole swarm. For a given limit ε, the conditions $CI \leq \varepsilon$ and the PSO iterations $iter = n$

Table 2 Material properties

E_{11}/psi	E_{22}/psi	G_{12}/psi	υ_{12}
18.5×10^6	1.89×10^6	0.96×10^6	0.3

Table 3 Loading and parameters of laminated panels

Case	Loading (lb/in)			Given number of stacks			
	N_x	N_y	N_{xy}	n_0	n_{45}	n_{90}	n_{total}
1	−20,000	−2,000	1,000	9	18	9	36
2	−15,000	−2,000	1,000	8	17	8	33
3	−10,000	−2,000	1,000	7	15	7	29
4	−5,000	−2,000	1,000	6	12	6	24
5	0	−2,000	1,000	4	8	4	16
6	0	−16,000	8,000	8	16	8	32
7	15,980	−14,764	10,160	9	8	13	30
8	−16,657	1,963	828	13	7	15	35

Table 4 Comparison of computational efficiency of the four algorithms

Case	Number of analyses required for 80% reliability				
	SGA	GR	PMX	SDPSO	PDPSO
1	10,432	1,184	1,328	8,764	2,336
2	8,600	856	1,224	7,780	1,564
3	5,216	776	1,024	4,532	1,140
4	3,304	608	824	2,564	880
5	1,672	408	560	1,100	580
6	7,984	1,328	1,480	5,380	1,524
7	23,944	11,840	5,784	14,800	4,640
8	26,320	2,216	2,504	22,440	2,716

SGA standard GA, *PMX* partially mapped crossover, *GR* gene rank crossover, refer to Liu et al. (2000b)

Table 5 Optimum lay-up for laminates

Case	λ	Optimum stacking sequence
1	0.948	$[(\pm45)_{17}/90_2/\pm45/(90_2)_2/0_2/(90_2)_2/((0_2)_2/90_2)_4]_s$
2	0.948	$[(\pm45)_{17}/90_2/0_2/90_2/(0_2)_2/(90_2)_2/0_2/90_2/(90_2/(0_2)_2)_2]_s$
3	0.909	$[(\pm45)_{15}/((90_2)_2/0_2)_2/(0_2/90_2)_3/(0_2)_2]_s$
4	0.870 0.8707	$[(\pm45)_{12}/((90_2)_2/0_2)_2/(0_2/90_2/0_2)_2]_s$
5	0.778	$[(\pm45)_8/((90_2)_2/(0_2/90_2)_2/(0_2)_2]_s$
6	0.773	$[(\pm45)_{16}/((90_2)_2/0_2)_2/(90_2/(0_2)_2)_2/((90_2)_2/0_2)_2]_s$
7	1.112	$[(90_2)_2/0_2/(90_2)_2/\pm45/90_2/\pm45/90_2/0_2/(90_2)_2/(0_2)_2/(90_2/\pm45)_2/\pm45/0_2/90_2/0_2/(90_2)_2/0_2/(\pm45)_2/(0_2)_2/\pm45]_s$
8	1.093	$[(\pm45)_6/90_2/0_2/(90_2)_2/\pm45/0_2/90_2/(90_2/0_2)_3/(90_2)_2/(0_2/90_2/0_2)_2/90_2/(90_2/0_2)_3/0_2]_s$

are trigger of particles self-escape. The procedure can be stated as

$$if$$

$$CI = \frac{1}{Swarm \times Dim} \sum_{i=1}^{Swarm} \left| \boldsymbol{Gbest}_{k-1}^{g} - \boldsymbol{Pbest}^{i} \right| \leq \varepsilon \;\&\; iter = n$$

$$then$$

$$rand\,(Swarm, Dim) \tag{21}$$

where *Swarm* is the number of particles; *Dim* is the dimension of each particle. In this work, a new approach is adopted in the calculation of arithmetic average due to the discrete characteristic of $\boldsymbol{Pbest}^{i}$ and $\boldsymbol{Gbest}_{k-1}^{g}$, which is illustrated by the following example.

$$\begin{aligned} \boldsymbol{Pbest}^{i} &= [0/0/0/45/45/-45/90], \\ \boldsymbol{Gbest}_{k-1}^{g} &= [0/90/0/45/-45/45/0] \\ \left| \boldsymbol{Gbest}_{k-1}^{g} - \boldsymbol{Pbest}^{i} \right| &= |(0,1,0,0,1,1,1)| = 4 \end{aligned} \tag{22}$$

$Gbest_{k-1,j}^{g}$ denotes j^{th} element of $\boldsymbol{Gbest}_{k-1}^{g}$, $\boldsymbol{Pbest}_{j}^{i}$ denotes j^{th} element of $\boldsymbol{Pbest}^{i}$, if $Gbest_{k-1,j}^{g} = Pbest_{j}^{i}$, then the j^{th} element of the result is zero; otherwise, it is one. And the sum of element generates the final result of $\left| \boldsymbol{Gbest}_{k-1}^{g} - \boldsymbol{Pbest}^{i} \right|$.

3.4 Programming procedure

The simulation procedures of the improved DPSO can be described as following:

1. Obtain the loads and number of plies used in the panel optimization based on the overall wing design.
2. Set the number of particles *Swarm*, dimension of searching space *Dim*, dispersed grade index ε and n. Randomly generate the initial particles in terms of the encoding rule given in Section 3.1. Set the current PSO iteration $t = 1$.
3. Decoding; checking memory; calculate the fitness value. Then evaluate all of the particles; renew the best positions of each individual $\boldsymbol{Pbest}^{i}$ and best positions of the whole swarm $\boldsymbol{Gbest}_{k-1}^{g}$.
4. Calculate the new velocity vector and position of the particles. Evaluate the validity of exchanges. If it is valid exchange, the optimizer would calculate fitness value of the particle. Then, update $\boldsymbol{Pbest}^{i}$ and $\boldsymbol{Gbest}_{k-1}^{g}$.
5. Calculate *CI* and *iter*; if *CI* and *iter* satisfy (17), the optimizer would generate new particle population and go to step 3; if not, go to step 6.
6. If the stop criterion is achieved, finish the PSO loop; if not, set $t = t + 1$ and go to step 3.

4 Numerical example and discussion

To show the efficiency of the proposed algorithm, a 24-inch square graphite-epoxy plate from Liu et al. (2000b) is taken as an example. The material properties and loading conditions of the plate are shown in Tables 2 and 3, respectively.

"A typical number of particle is 20–40. For most of the problems, 10 particles are enough to get good results. However, 100–200 particles were also used for some difficult problems (Kathiravan and Ganguli 2007)". In this work, for

Table 6 Three new load cases

Case	Loading (lb/in)			Given number of stacks			
	N_x	N_y	N_{xy}	n_0	n_{45}	n_{90}	n_{total}
9	10,657	−14,850	10,000	14	10	15	39
10	10,657	−14,850	10,000	14	8	17	39
11	11,274	−18,960	8,000	10	8	12	30

Table 7 Computational efficiency and optimum lay-up of three new load cases

Case	λ	Number of analyses required for 80% reliability	Optimum stacking sequence
9	2.708	5,480	$[(90_2)_2/0_2/90_2/\pm45/(90_2)_2/\pm45/((90_2)_2/(0_2)_2)_2/(\pm45/(90_2)_2)_2/0_2/(\pm45)_2/0_2/\pm45/90_2/(\pm45/(0_2)_2)_2/90_2/0_2/\pm45/(0_2)_2]_s$
10	2.734	5,964	$[90_2/\pm45/((90_2)_2/0_2)_2/90_2/0_2/90_2/\pm45/90_2/(\pm45)_2/((90_2)_2/0_2)_2/0_2/(90_2)_2/0_2/90_2/0_2/(\pm45/0_2/90_2/0_2)_2/0_2/(\pm45)_2/0_2]_s$
11	0.919	4,380	$[(90_2)_2/\pm45/((90_2)_2/0_2)_2/(\pm45/90_2/0_2/90_2)_2/90_2/0_2/\pm45/0_2/(0_2/(\pm45)_2)_2/(0_2)_2/90_2]_s$

comparing with Liu et al. (2000b) a particle size of 8 is chosen and it gives reasonable good results, as illustrated below. The inertia weight w is set linearly decreasing from 0.9 to 0.45. $c_1 = 0.4$ and $c_2 = 0.6$ are selected which give higher efficiency in this work. The other PSO parameters are: dispersed grade index $\varepsilon = 0.2$, iteration limit $n = 20$ and PSO evolution limit for the total optimization process is 500. The reliability is discussed here at some specified computational cost. The reliability is measured by performing 100 optimization runs and checking how many of the 100 runs reached the optimum at any given point. For instance, if 80 runs reached the global optimum after 500 analyses per run, then the reliability of this algorithm is estimated to be 0.80 after 500 analyses. Similarly with Liu et al. (2000b), a design is considered to be a practical optimum if the critical load factor was within 0.5% of the global optimum.

Table 4 shows a comparison of the computational efficiency by using permutation DPSO, standard DPSO and other methods in Liu et al. (2000b). The results of SGA, PMX and GR are quoted from Liu et al. (2000b). The optimum lay-up for laminates is shown in Table 5.

From Table 4, it is clear that the computational efficiency of PDPSO, which is comparable with that of GR and PMX, has been greatly improved compared with SDPSO. Furthermore, the two permutation GAs perform a little better than the PDPSO in the case 1–6 and 8. However, in case 7, the efficiency of GR and PMX is lower than the permutation DPSO. From Table 5, for case 7 when the outermost plies in the optimum design are not ±45, the computational efficiency of PDPSO seems not greatly affected by the contiguity constraint. To explore the performance for similar laminates, three new cases, defined in Table 6, were selected.

The results summarized in Table 7 show that, as expected, thicker laminates are computationally more expensive to optimize. The proposed PDPSO demonstrates higher stability and efficiency to optimize laminate of which outermost plies in the optimum design are not ±45.

5 Conclusion

In this paper, the maximum buckling load factor of laminates with 0, 45 and 90° layers were investigated under different loading conditions. A novel optimization method PDPSO, which originated from SDPSO method, was proposed to solve the current stacking sequence optimization problem more efficiently. The contiguity constraint was implemented through a penalty function. To improve the computational efficiency, the techniques of self-escape, checking memory and valid/invalid exchanges were incorporated into present PDPSO.

From the numerical examples, it has been demonstrated that the computational efficiency of PDPSO is comparable with that of GR and PMX, and has been greatly improved while compared to SDPSO. Moreover, the PDPSO is more efficient than all other methods when dealing with laminates whose outer plies are affected by the contiguity constraint limit.

PDPSO is quite efficient in dealing with SSO problems and it has great potential to be applied to aero-elastic tailoring optimization problem and more combinatorial optimization problems in future.

References

Bloomfield M, Herencia JE, Weaver PM (2008) Optimization of anisotropic laminated composite plates incorporating non-conventional ply orientations. In: Proceeding of the 49th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics, and materials conference. Schaumburg, IL

Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Sixth international symposium on micro machine and human science (MHS'95). Nagoya, Japan, pp 39–43

Faggiani A, Falzon BG (2007) Optimization strategy for minimizing damage in postbuckling stiffened panels. AIAA J 45:2520–2528

Fourie PC, Groenwold AA (2001) The particle swarm algorithm in topology optimization. In: Proceedings of the fourth world

congress of structural and multidisciplinary optimization. Dalian, pp 52–63

Herencia JE, Weaver PM, Friswell MI (2007) Optimization of long anisotropic laminated fiber composite panels with T-shaped stiffeners. AIAA J 45:2497–2509

Kameyama M, Fukunaga H (2007) Optimum design of composite plate wings for aeroelastic characteristics using lamination parameters. Comput Struct 85:213–224

Kathiravan R, Ganguli R (2007) Strength design of composite beam using gradient and particle swarm optimization. Compos Struct 81:471–479

Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: Proceedings of the world multiconference on systemics, cybernetics and informatics. Piscataway, NJ, pp 4104–4109

Li X, Tian P, Hua J, Zhong N (2006) a Hybrid discrete particle swarm optimization for travelling salesman problem. Lect Notes Comput Sci 4247:181–188

Liu B, Haftka RT, Akgün MA (2000a) Two-level composite wing structural optimization using response surfaces. Struct Multidisc Optim 20:87–96

Liu B, Haftka RT, Akgün MA, Todoroki A (2000b) Permutation genetic algorithm for stacking sequence design of composite laminates. Comput Methods Appl Mech Eng 186: 357–372

Muc A, Gurba W (2001) Genetic algorithms and finite element analysis in optimization of composite structures. Compos Struct 54:275–281

Park JH, Hwang JH, Lee CS, Hwang W (2001) Stacking sequence design of composite laminates for maximum strength using genetic algorithms. Compos Struct 52:217–231

Rameshkumar K, Suresh RK, Mohanasundaram KM (2005) Discrete particle swarm optimization algorithm for permutation flowshop scheduling to minimize makespan. Lect Notes Comput Sci 3612:572–581

Sciuva MD, Gherlone M, Lomario D (2003) Multiconstrained optimization of laminated and sandwich plates using evolutionary algorithms and higher-order plate theories. Compos Struct 59(1):149–154

Seresta O, Abdalla M, Gurdal Z (2006) Minimum weight design of composite structures with local postbuckling and blending constraints. AIAA paper 2006–1818

Shi YH, Eberhart RC (1998) A modified particle swarm optimizer. In: Proceedings of the IEEE international conference on evolutionary computation. Anchorage, AK, pp 69–73

Suresh S, Sujit PB, Rao AK (2007) Particle swarm optimization approach for multi-objective composite box-beam design. Compos Struct 81:598–605

Venter G, Sobieszczanski-Sobieski J (2004) Multidisciplinary optimization of a transport aircraft wing using particle swarm optimization. Struct Multidisc Optim 26:121–131

Walker M, Smith RE (2003) A technique for the multiobjective optimization of laminated composite structures using genetic algorithms and finite element analysis. Compos Struct 62:123–128

Whitney JM (1985) Structural analysis of laminated anisotropic plates. Technomic Publishing Company, Lancaste, pp 119–122