Using CPSO for the Engineering Optimization Problems

Ching-Long Su[1], Shutan Hsieh [2],
[1]Department of Information Management, Chang Jung Christian University
396 Chang Jung Rd., Sec.1,Kway Jen, Tainan 71101, Taiwan
clsu@mail.cjcu.edu.tw

[2]Department of Accounting, National Kaohsiung University of Applied Sciences
415 Chien Kung Road, Kaohsiung 807, Taiwan
shutan@cc.kuas.edu.tw

Abstract: - The aim of this study is to solve the optimization problems of different engineering designs by using nonlinear mixed integer programming mode. In the past, this type of engineering design optimization problem has been widely studied and discussed. They are usually solved through mathematical programming method or heuristics. However, there are more constraints and more constraints that cannot be satisfied. In solving this type of problems, we used a penalty guided cooperative particle swarm optimization to avoid the disadvantage of decreased efficiency from the increase of search spatial dimension and to raise the efficiency. In resolving some engineering design problems, the results shows that the solutions found by cooperative particle swarm optimization are equal or better than the best-known solutions from past literature. Thus, the results of this study indicate that cooperative particle swarm optimization is another effective method to find solutions to optimization problems.

Key-Words: - Engineering optimization, Nonlinear mixed integer programming, Cooperative particle swarm optimization.

1 Introduction

Many optimization problems and their solutions are related to satisfying many constraints. Therefore, these constraints are focused on to search optimization and to find better target values, which must also satisfy the constraints. Often, optimization problems with constraints are listed as follow:

$$Min. \quad f(x)$$
$$subject\ to \quad g_i(x) \leq b,\ i = 1, 2, \ldots, n.$$
$$h_j(x) = 0,\ j = 1, 2, \ldots, p.$$

In this minimization problem, a decision vector can be expressed as $x=[x_1, x_2, \ldots x_d]^T$ and n is referred as the number of unequal constraints, while p refers to the number of equal constraints. Traditional mathematical procedures such as Lagrange multiplier method, usually requires some target functions and derivative information of constraints. However, the outcomes are often partial best solution. In recent years, as evolutionary algorithms have the best ability in full search for best solutions, many optimization problems have also begun to utilize the advantages of evolutionary algorithms to conduct problem exploration and development.

Some approaches have been considered to deal with the above problems. For example, mathematic programming relative approaches are applied [Hikita et al. 1993] [Nakagawa and Miyazaki 1981] [Xu et al. 1990]. However, most of those require derivatives for all nonlinear constraint functions. It makes the exact optimal solutions to this problem hard to be derived easily because of the highly computational complexity. For overcome this difficulty, Hsieh et al. (1998) and Yokota et al. (1996) applied genetic algorithms (GAs) and Chen (2005) applied immune algorithms (IAs) to solve these problems more effectively.

Recently, particle swarm optimization (PSO), which was originally proposed by Kennedy and Eberhart (1995), have been widely studied and applied to a variety of optimization problems. The main concept of PSO is based on the food-searching behavior of birds flocking or fish schooling. When PSO is adopted to solve problems, each particle has its own location and velocity, which determine flying direction and distance respectively. The fitness value is evaluated by the optimization function. Compared with other meta-heuristic evolutionary approaches such as GAs and IAs, the PSO has the following advantages (i) less parameters (ii) easy implementation (iii) fast constringency. The advantages are good for solving the mixed-integer programming problems because a population of particles in PSO can operate simultaneously so that the possibility of paralysis in the whole process can be reduced. In this article a penalty guided PSO is applied to solve the engineering design problems.

Particle Swarm Optimization (PSO) is a method inspired from observing the social behavior of bird flocks. Due to the simplicity of the concept, easily executable, and fast convergence, PSO has been successfully applied in many optimization fields. Potter et al. (1994) proposed applying the method of partitioning the vector in search space on genetic algorithm, proving significant improvement on execution efficiency. Furthermore, Van den Bergh et al. (2004) utilized the technique of search space segmentation proposed by Potter et al. (1994) on PSO and proposed cooperative particle swarm optimization (CPSO). In order to solve the constraints in optimization, penalty function method (Chen, 2006) can be used to effectively search for feasible solutions.

This study uses cooperative particle swarm optimization (Van Den Bergh et al., 2004) (He and Wang, 2007)to search the solution for optimization in engineering design (including system reliability design and machine parts design problem) through comparing the solution from cooperative particle swarm optimization and best-known solutions from past literature.

2 Introduction of Particle Swarm Optimization

Particle swarm optimization (PSO) is a type of evolution algorithm proposed by J. Kennedy and R.C. Eberhart in 1995. Random optimization of PSO can be described as the behavior of a flock of birds or the social behavior of a group of human. Searching for the best position in a specific space means the best solution. PSO and genetic algorithm (GA) both produce the same set of initial position. The best solution is derived through evolution. The difference from GA is that there is no crossover and mutation in PSO. In the swarms of PSO, each particle represents a possible solution. Each particle decides the direction and speed it travels based on its own experiences and the information exchanged between the swarms. The above description of the evolution is expressed as the mathematical model below:

$$V_i(t+1)=wv_i(t)+c_1\times rand()\times(P_{Best}-x_i)+c_2\times rand()\times(G_{Best}-X_i)$$

$$x_i(t+1)=x_i(t)+v_i(t+1)$$

If s represents the swarm size then x_i expresses the position of particle i in a specific search space, $x_i=(x_1,x_2,...,x_s)$ while $x_i(t)$ represents the current position of particle i. $x_i(t+1)$ represents the position of particle i in the next iteration. $v_i(t)$ is the current velocity of particle i. $v_i(t+1)$ is the velocity of particle i in the next iteration. Therefore, the new position of each particle is calculated with current position $x_i(t)$ plus the new velocity $v_i(t+1)$ from equation (4) to determine the position $x_i(t+1)$ of the particle in the next iteration. Equation (4) can be seen in three parts. The first is the previous habits of the particle. The second part is cognition, and the third part is social. The so-called "cognition" represents the thinking and experience of the particle itself, while "social" means the sharing of information among the particles.

Moreover, w in equation (4) represents inertia weight. c_1 and c_2 are positive constant called acceleration coefficients. *rand()* is the uniform distribution of a random number in [0, 1]. P_{Best} and G_{Best} are the best positions of particle and the swarm, respectively. Usually, the velocity v_i of each particle is constrained by V_{max} to fall within $[-V_{max},V_{max}]$. If it is out of bounds, v_i will be revised as $-V_{max}$ or V_{max}.

3 Research Approach

3.1 Cooperative Particle Swarm Optimization

In this study, the cooperative particle swarm optimization (CPSO) is applied to solve the nonlinear engineering optimization problems. The differences in the mathematical model with regard to updating velocity and position in the original PSO are as follow:

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1\times rand()\times(P_{Best_{i,j}} - x_i) + c_2 \times rand()\times(G_{Best_j} - x_{i,j})$$

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1)$$

As CPSO utilizes multi-swarm search method, $x_{i,j}(t)$ represents the current position of particle i in jth swarm. $x_{i,j}(t+1)$ represents the position of particle i in jth swarm in the next iteration. $v_{i,j}(t)$ represents the current velocity of particle i in jth swarm. $v_{i,j}(t+1)$ represents the velocity of particle i in jth swarm in the next iteration. $P_{Best_{i,j}}$ is the best position of particle i in j_{th} swarm. G_{Best_j} is the best position in population in j_{th} swarm. The steps of CPSO are explained as follow:

Step 1 According to n, the dimension of the problem, produce n groups of particle swarms with m particles and random positions and velocity.

Step 2 Use different combinations of n swarms and personal best solutions of n swarms and substitute appropriate functions to find the fitness value of the following combination and assess the result.

Step 3 In particle swarm, if a new personal best solution is better than the previous one, the position of the particle will replace the previous best position (P_{Best}).

Step 4 In particle swarm, if a new best solution is better than the previous one of the population, the position of the particle will replace the previous best position (G_{Best}).

Step 5 Update the velocity and position of particle according to equations (4) and (5).

Step 6 Assess whether stop condition is satisfied. If not, return to step 2.

The following is a description of the differences between PSO and CPSO, expressed in matrices. In CPSO, there are mainly three matrices: particle swarm matrix, personal best matrix, and global best matrix. Particle swarm matrix produces n groups of swarms according to the dimensions of the problem and defines the number of particles m in each group. Thus, particle swarm matrix is a $m \times n$ matrix. Personal best matrix records the best position of each particle in every swarm group up to the moment; therefore, it is also a $m \times n$ matrix. Global best matrix records the best positions found by each swarm group up to the moment. The matrices are illustrated as in Figure 1.

For each iteration, the position found by each particle is assessed through fitness function, and a fitness value is derived. This decides whether to update or save the positions. The following is an introduction of the difference in the update process between CPSO and PSO:

(1) Take the first particle of the first group in particle swarm matrix and substitute it for the first particle of the first group in personal best matrix. Plug in the appropriate function and assess whether it is better than the best fitness value of previous personal best and the fitness value of group best before deciding whether $P_1.x_1$ substitutes $P_1.y_1$ and $P_1.\hat{y}$ (as shown in Figure 2).

Particle Swarm Matrix

$P_1.x_1$	$P_2.x_1$	$\cdots$	$P_{n-1}.x_1$	$P_n.x_1$	$\rightarrow$	$f(x_1)$
:	:	:	:	:	$\rightarrow$	:
$P_1.x_m$	$P_2.x_m$	$\cdots$	$P_{n-1}.x_m$	$P_n.x_m$	$\rightarrow$	$f(x_m)$

$P_j.x_i$: Position of *i*th particle in *j*th swarm

Particle Swarm Matrix

$P_1.x_1$	$P_2.x_1$	$\cdots$	$P_{n-1}.x_1$	$P_n.x_1$	$\rightarrow$	$f(x_1)$
:	:	:	:	:	$\rightarrow$	:
$P_1.x_m$	$P_2.x_m$	$\cdots$	$P_{n-1}.x_m$	$P_n.x_m$	$\rightarrow$	$f(x_m)$

$P_j.x_i$: Position of ith particle in jth swarm

Particle Swarm Matrix

$P_1.\hat{y}$	$P_2.\hat{y}$	$\cdots$	$P_{n-1}.\hat{y}$	$P_n.\hat{y}$	$\rightarrow$	$f(\hat{y})$

Pj.xi : Position of ith particle in jth swarm

Fig.1 Three matrices of CPSO: particle swarm matrix, personal best matrix, and global best matrix.

$P_1.x_1$	$P_2.y_1$	$\cdots$	$P_{n-1}.y_1$	$P_n.y_1$	$\rightarrow$	$f(temp_1)$

```
if $f(temp_1)$ is better than $f(y_1)$ then
        $P_1.y_1 = P_1.x_1$
        $f(y_1) = f(temp_1)$
end

if $f(temp_1)$ is better than $f(\hat{y})$ then
        $P_1.\hat{y} = P_1.x_1$
        $f(\hat{y}) = f(temp_1)$
end
```

Fig. 2 The individual particle updating process in CPSO

(2) Follow the assessment and update as described in (1) until the condition is no longer satisfied (as shown in Figure 3).

$P_1.y_1$	$P_2.x_1$	$\cdots$	$P_{n-1}.y_1$	$P_n.y_1$	$\rightarrow$	$f(temp_2)$
		:				

Fig. 3 New individual particle

3.2 Constrained Optimization

In order for particles to quickly move to the correct positions in each iteration, by allowing the movement of particles in unfeasible solution positions to feasible solution positions, reducing the occurrences of constraint violations, best solution or the position closest to best solution is more easily obtained. Through penalty function, constraint violations can be dealt with and unfeasible solutions are given penalty value to guide particles to feasible areas. According to equation (2), when constraint function $g(x)_i$ violates constraints, it is dealt with according to the following:

$$Penalty_i = \begin{cases} g(r,n) - b, & if\ g(r,n) > b \\ 0, & Otherwise \end{cases}$$

After defining penalty function, the fitness of each particle are adjusted through fitness function. The fitness function is defined as follow:

$$Fitness = \frac{f(x)}{1 + \sum_{i=1}^{n} Penalty_i}$$

where n represents the number of constraints in the problem. $Penalty_i$ is applicable in problems of maximization. When penalty occurs, the fitness value should be lower. Conversely, in solving minimization problems, to give penalty is to raise the fitness value as shown in Fitness.

$$Fitness = f(x) \times (1 + \sum_{i=1}^{n} Penalty_i)$$

4 Numerical Results

To evaluate the performance of the proposed CPSO approach for the mixed-integer nonlinear engineering design problems, six test problems (P1 ~ P6) from previous literature are solved. They are compared how CPSO and other methods find solutions. Please refer to Table 1 for parameter settings. This study includes six problems described and illustrated in the table.

Table 1 Initial setting of parameters

parameters	value
Acceleration coefficient (c_1, c_2)	1.41
Inertia weight (w)	0.729
Maximum velocity (V_{max})	1
Swarm size	80
Iiterations	500

The determination of the parameters in evolutionary algorithm is a significant problem for the implementation. However, no formal methodology can be used to solve the problem because various value-combinations of the parameters result to different characteristics as well as different performance. Therefore, one should note that the best values for the parameters in the algorithm are case-dependent and based upon the experience from preliminary runs.

4.1 Series-parallel system

The series-parallel system is made up of five sub-systems as shown in Figure 4. It is a problem of non-linear mixed integer programming with three non-linear and inseparable constraints. The system total reliability is Rs. The reliability of sub-systems is expresses as R_j=$R_j(n_j)$=1-$(1-r_j)^{n_j}$. Failure rate is Q_j=1-R_j, where Q_j=1-R_j. R_j represents the reliability of jth sub-system. n_j represents the number devices used by jth sub-system. r_j is the reliability of device used by jth sub-system. The goal of the problem is to maximize the reliability of the system; therefore, the number and reliability of the device of each sub-system are distributed.

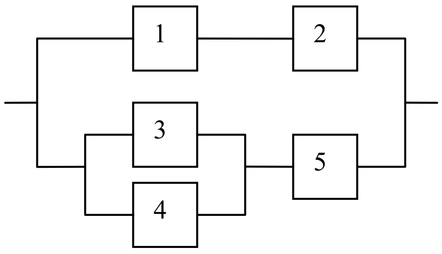

Fig. 4 Series-parallel system

The mathematical model is as followed:

Max. $$f(r,n) = \prod_{j=1}^{5} [1 - (1 - r_j)^{n_j}]$$

Subject to $$g_1(r,n) = \sum_{j=1}^{5} C(r_j)(n_j + \exp(\frac{n_j}{4})) \le c_Q$$

$$g_2(r,n) = \sum_{j=1}^{5} v_j n_j^2 \le v_Q$$

$$g_3(r,n) = \sum_{j=1}^{5} w_j n_j \exp\left(\frac{n_j}{4}\right) \le w_Q$$

$$1 \le n_j \le 10 \ : \ \text{integer}, j = 1,2,...,5$$

$$0.5 \le x_j \le 1 - 10^{-6} \ : \ \text{real}, j = 1,2,...,5$$

$C(r_j)$ in $g_1(r, n)$ in the model above represents the cost for each sub-system for using device. Related parameter and resource constraints for $C(r_i) = \alpha_i[-T/\ln(r_i)]^{\beta_i}$ are shown in Table 2.

Gen et al. (2006) used hybrid genetic algorithm and calculated system total reliability to be 0.931676. After the evolution of CPSO, the result is 0.931679, slightly better than the result from the literature (as shown in Table 3).

Table 2 Series-parallel system parameters and constraints

Sub system	$10^5\alpha_j$	β_j	v_j	w_j	c_Q	v_Q	w_Q
1	2.33	1.5	1	7	175.0	110.0	200.0
2	1.45	1.5	2	8			
3	0.541	1.5	3	8			
4	8.050	1.5	4	6			
5	1.950	1.5	2	9	Operation time T 1000h		

Table 3 Comparison of CPSO, Series-parallel system reliability, and other method

	Prasad et al. (2000)	Gen et al. (2006)	CPSO
n^*	(3,2,2,3,3)	(3,2,2,3,3)	(3,2,2,3,3)
r_1	0.779780	0.780874	0.780532
r_2	0.872320	0.871292	0.871654
r_3	0.902450	0.902316	0.902962
r_4	0.710810	0.711945	0.711181
r_5	0.788160	0.786995	0.787213
$f(x)$	0.931678	0.931676	0.931679

4.2 Bridge Circuit System

Bridge circuit system is also a problem that considers the optimization of system reliability distribution. The aim is to maximize system reliability. The constraints of the bridge system are the same as series-parallel system in 4.1. The target is expresses as followed:

$$\begin{aligned}\text{Max.}\quad f(r,n) = {} & R_1(r,n)R_2(r,n) + \\ & (1-R_1(r,n)R_2(r,n))R_3(r,n)R_4(r,n) + \\ & (1-R_2(r,n))(1-R_3(r,n))R_1(r,n)R_4(r,n)R_5(r,n) + \\ & (1-R_1(r,n))(1-R_4(r,n))R_2(r,n)R_3(r,n)R_5(r,n)\end{aligned}$$

Gen et al. (2006) used hybrid genetic algorithm and calculated system total reliability to be 0.999889. After the evolution of CPSO, the result is 0.999890, slightly better than the result from the literature (as shown in Table 4)

Table 4 Comparison of CPSO and other methods in bridge system reliability distribution

	Gen et al. (2006)	CPSO
n^*	(3,3,3,3,1)	(3,3,2,4,1)
r_1	0.808258	0.829535
r_2	0.866742	0.859574
r_3	0.861513	0.913151
r_4	0.716608	0.645054
r_5	0.766894	0.703808
$f(x)$	0.999889	0.999890

Although, the improvement seems not be very large, it is not easy to get the better feasible improved solution in a reasonable cpu time.

4.3 HS System

The hierarchical series-parallel system is made up of a group of series and parallel systems (see Figure 5). It is a non-linear and inseparable structure. Therefore, the HSP system has 10 sub-systems. The target function is expressed with pivotal decomposition method (Kuo et al., 2002).

Refer to Table 5 for the reliability and parameters of each device. b_1 and b_2 represent maximum usable resources, 120 and 300, respectively. System total reliability is R_s and the reliability of sub-system is $R_j = R_j(x_j) = 1-(1-r_j)^{x_j}$. The failure rate is Q_j=1-R_j，j=1,…,10, where R_j is the reliability of jth sub-system. x_j is the number of device used by jth sub-system. r_j is the reliability of device used by jth sub-system.

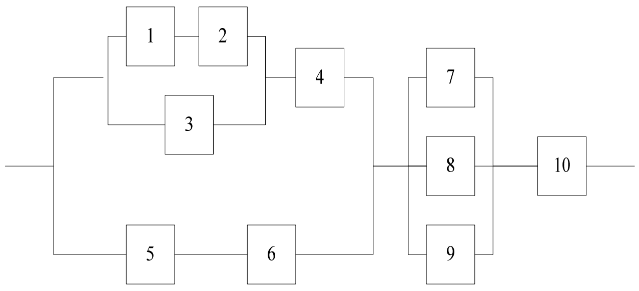

Fig. 5 HSP system

The mathematical model is as followed:

Max. $R_s = \left\{1 - \left\langle 1 - \left[1 - Q_3(1 - R_1R_2)\right]R_4 \right\rangle (1 - R_5R_6)\right\}(1 - Q_7Q_8Q_9)R_{10}$

Subject to

$$c_1 \exp\left(\frac{x_1}{2}\right)x_2 + c_2 \exp\left(\frac{x_3}{2}\right) + c_3c_4 + c_4\left[x_5 + \exp\left(\frac{x_5}{4}\right)\right] + c_5x_6^2x_7 + c_6x_8 + c_7x_9^3 \exp\left(\frac{x_{10}}{2}\right) \le b_1$$

$$w_1x_1^2x_2 + w_2\left(\frac{x_3x_4}{6}\right) + w_3w_5 \exp\left(\frac{x_6}{4}\right) + w_4x_7x_8^3 + w_5\left[x_9 + \exp\left(\frac{x_9}{2}\right)\right] + w_6x_2 \exp\left(\frac{x_{10}}{4}\right) \le b_2$$

$$l_j \preceq x_j \preceq u_j$$

Table 5 HSP system parameter distribution

j	1	2	3	4	5	6	7	8	9	10
r_j	0.83	0.89	0.92	0.85	0.89	0.93	0.83	0.94	0.82	0.91
c_j	8	4	2	2	1	6	2	8	-	-
w_j	16	6	7	12	7	1	9	-	-	-
l_j	1	1	1	1	1	1	1	1	1	1
u_j	4	5	6	7	5	5	3	3	4	4

He et al. (2006) found that when u_{10} is set to 2, the outcome of x_{10} is 4. Therefore, u_{10} is revised as 4. The system total reliability obtained by He et al. (2006) using multi-branch procedure is 0.999876, while the result obtained with CPSO is 0.999881. Compared with the results from past literature, the result obtained by CPSO is superior to that from previous method (as shown in Table 6). Again, it can show that CPSO is another good approach to solve the engineering optimization problems.

Table 6 Comparison of HSP system with CPSO and other method

	Gopal et al. (1978)	Ha et al. (2006)	CPSO
x^*	(1,1,1,2,2,2,1,2,1,4)	(1,1,3,4,2,1,1,3,1,4)	(1,2,3,6,4,3,3,1,1,4)
R_s	0.999097	0.999876	0.999881

4.4 Spring design

Spring design problem (Singiresu et al., 2005) is described as in Figure 6.The aim of this problem is to solve the problem of minimizing the volume of spring material. There are three design variables to consider:

$$X=[x_1, x_2, x_3]^T=[N, d, D]^T$$

of which, N represents the number of spring coils and d is the diameter of metal wire. D is the diameter of the spring.

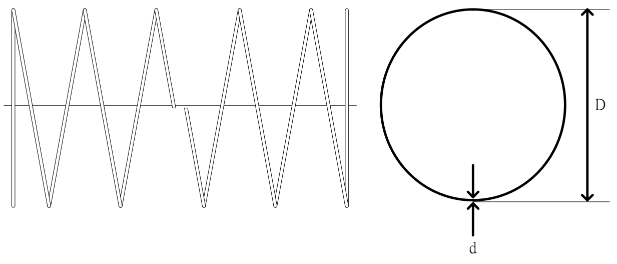

Fig. 6 Spring design (Singiresu et al., 2005)

The mathematical model is as follow:

Min. $f(X) = \pi^2 x_3 x_2^2 (x_1 + 2)/4$

Subject to

$$g_1(X) = S - 8K_sP_{\max}x_3/\left(\pi x_2^3\right) \ge 0$$

$$g_2(X) = l_{\max} - \left(\delta + 1.05(x_1 + 2)x_2\right) \ge 0$$

$$g_3(X) = x_2 - d_{\min} \ge 0$$

$$g_4(X) = D_{\max} - x_3 \ge 0$$

$$g_5(X) = C - 3 \ge 0$$

$$g_6(X) = \delta_{pm} - \delta \ge 0$$

$$g_7(X) = l_f - \delta - (P_{\max} - P_{\text{load}})/K - 1.05(x_1 + 2)x_2 \ge 0$$

$$g_8(X) = (P_{\max} - P_{\text{load}})/K - \delta_w \ge 0$$

where, C=x$_3$/x$_2$, K_s = (4C-1)/(4C-4) + 0.615/C, K=($G x_2^4$)/(8 x_3^3 x_1), 5≤ x_1 ≤20, x_1 =5+k, k =0,1,…,15, 0.207≤ x_2 ≤0.500, x_2∈ {0.207, 0.225, 0.244, 0.263, 0.283, 0.307, 0.331, 0.362, 0.394, 0.4375, 0.5000}, 1.0≤ x_3 ≤3.0

The settings of other parameters are as follows:

$P_{\max} = 1000$, $S = 1.89e5$, $G = 1.15e7$, $l_{\max} = 14$, $d_{\min} = 0.2$, $D_{\max} = 3$, $\delta_{pm} = 6$, $P_{load} = 300$, $l_f = 6.6$, and $\delta_w = 1.25$

Also, x_1 must be an integer and x_2 is a discrete value. x_3 is a continuous variable. The following are possibilities for x2: 0.207, 0.225, 0.244, 0.263, 0.283, 0.307, 0.331, 0.362, 0.394, 0.4375, 0.500.

The smallest volume of the material obtained by hybrid genetic algorithm proposed by Singiresu et al. (2005) is 2.66342, while the result obtained with CPSO is 2.65856. Compared with the results from past literature, the result obtained by CPSO is superior to those of two earlier methods (as shown in Table 7).

Table 7 Comparison of best spring design and other methods

	Salajegheh et al. (1993)	Singiresu et al. (2005)	CPSO	Remarks
x_1	10	9	9	integer
x_2	0.283	0.283	0.283	discrete
x_3	1.180701	1.22528	1.22304	continuous
$f(x)$	2.7995	2.66342	**2.65856**	

4.5 Pressure vessel design

Pressure vessel is a cylindrical container with one end as a semi-sphere as shown in Figure 7. It is a storage tank for pressurized gas. The aim of problem is to minimize total material cost. Below are four design variables to consider:

$$X=[x_1, x_2, x_3, x_4]^T=[T_s, T_h, R, L]^T$$

of which, T_s represents the thickness of the outer shell and T_h is the thickness of the semi-spherical outer shell. R is the radius of the outer shell, while L is the length.

The mathematical model is as followed:

Minimize

$$f(X)=0.6224x_1x_3x_4+1.7781x_2x_3^2+3.1611x_1^2x_4+19.8621x_1^2x_3$$

Subject to

$$g_1(X)=x_1-0.0193x_3\geq 0$$

$$g_2(X)=x_2-0.00954x_3\geq 0$$

$$g_3(X)=\pi x_3^2x_4+\frac{4}{3}\pi x_3^3-750\times 1728\geq 0$$

$$g_4(X)=240-x_4\geq 0$$

$$g_5(X)=x_1-1.1\geq 0$$

$$g_6(X)=x_2-0.6\geq 0$$

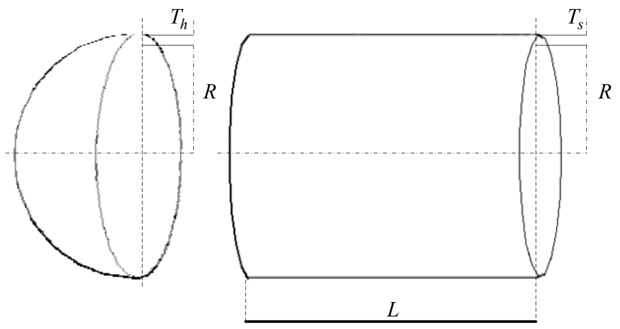

Fig. 7 Pressure vessel

Moreover, x_1 and x_2 must be integral multiples of 0.0625. x_3 and x_4 are continuous variables. As the upper and lower limits of x_1, x_2, x_3, and x_4 were not mentioned in earlier literature, here the upper and lower limits of x_1, x_2, x_3, and x_4 are set as below:

$$0\leq x_1\leq 10\ ;\ 0\leq x_2\leq 10\ ;\ 0\leq x_3\leq 100\ ;\ 0\leq x_4\leq 100$$

Table 8 Comparison of best pressure vessel design

	Sandgren (1990)	Shih et al. (1995)	Fu et al. (1991)	Singiresu et al. (2005)	CPSO
x_1	1.125	1.125	1.125	1.1875	1.125
x_2	0.625	0.625	0.625	0.625	0.625
x_3	48.97	47.448	48.3807	61.4483	58.269
x_4	106.72	119.98	111.7449	27.4037	43.81
$f(x)$	7982.5	8160.80	8048.619	7284.02	**7200.67**

The smallest material cost obtained by hybrid genetic algorithm proposed by Singiresu et al. (2005) is 7284.02, while the result obtained with CPSO is 7200.667. Compared with the results from past literature, the result obtained by CPSO is superior to those in past literature (as in Table 8).

4.6 Decelerator design

The aim of this problem as described in Figure 8 is to minimize the total weight of the decelerator. Below are the seven design variables:

$$X=[x_1,x_2,x_3,x_4,x_5,x_6,x_7]^T=[b,m,n,l_1,l_2,d_1,d_2]^T$$

of which, b is the width of the front side, while m is parts of the saw tooth. n is the number of small gears on the saw tooth. l_1 is the distance between axial 1 and bearings. l_2 is the distance between axial 2 and bearings. d_1 is the diameter of axial 1, while d_2 is the diameter of axial 2.

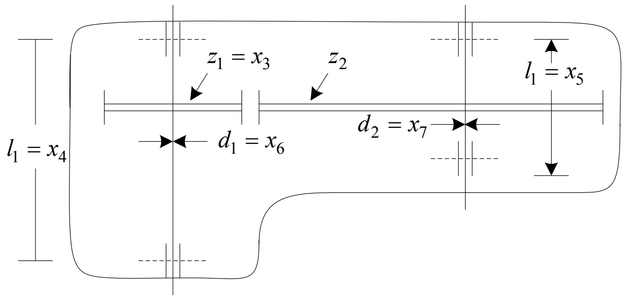

Fig. 8 Decelerator design

Minimize $f(x)=0.7854x_1 x_2^2 (3.33333 x_3^2 +14.9334x_3 - 43.0934) - 15.08x_1(x_6^2+x_7^2) + 7.477 (x_6^3+x_7^3) + 0.7854 (x_4 x_6^2 +x_5+ x_7^2)$

Subject to

$$g_1(X)=27x_1^{-1}x_2^{-2}x_3^{-1} \le 1$$

$$g_2(X)=397.5x_1^{-1}x_2^{-2}x_3^{-2} \le 1$$

$$g_3(X)=1.93x_2^{-1}x_3^{-1}x_4^{3}x_6^{-4} \le 1$$

$$g_4(X)=1.93x_2^{-1}x_3^{-1}x_5^{3}x_7^{-4} \le 1$$

$$g_5(X)=\left[\left(\frac{745x_4}{x_2x_3}\right)^2+(16.9)10^6\right]^{0.5} \Big/ \left(0.1x_6^3\right) \le 1100$$

$$g_6(X)=\left[\left(\frac{745x}{x_2x_3}\right)^2+(157.5)10^6\right] \Big/ \left(0.1x_7^3\right) \le 850$$

$$g_7(X)=x_2x_3 \le 40$$

$$g_8(X)=x_1/x_2 \ge 5$$

$$g_9(X)=x_1/x_2 \le 12$$

$$g_{10}(X)=(1.5x_6+1.9)x_4^{-1} \le 1$$

$$g_{11}(X)=(1.1x_7+1.9)x_5^{-1} \le 1$$

$$2.6 \le x_1 \le 3.6$$

$$0.7 \le x_2 \le 0.8$$

$$17 \le x_3 \le 28$$

$$7.3 \le x_4 \le 8.3$$

$$7.3 \le x_5 \le 8.3$$

$$2.9 \le x_6 \le 3.9$$

Furthermore, x_1, x_2, x_4, and x_5 must be integral multiples of 0.1. x_6 and x_7 must be integral multiples of 0.01, and x_3 must be an integer. The smallest weight obtained by hybrid genetic algorithm proposed by Singiresu et al. (2005) is 3000.83, while the result obtained with CPSO is 2998.27. Compared with past literature, the result obtained by CPSO is superior to those in past literature (as shown in Table 9).

Table 9 Comparison of best decelerator design and other methods

	Singiresu et al. (2005)	CPSO	Remarks
x_1	3.5	3.5	discrete
x_2	0.7	0.7	discrete
x_3	17	17	integer
x_4	7.3	7.3	discrete
x_5	7.8	7.8	discrete
x_6	3.36	3.35	discrete
x_7	5.29	5.29	discrete
$f(x)$	3000.83	**2998.27**	

5 Conclusions

As cooperative particle swarm optimization raises the effectiveness of the execution dealing with multiple dimension problems, this study uses it to solve the problems of non-linear mixed integer programming. These problems include series-parallel reliability system, bridge circuit system, hierarchical series-parallel reliability system, spring design problem, pressure vessel problem, and Decelerator design problem. Totally, these six engineering problems are with highly complexity and difficult to be solved. This study has proven that through CPSO search, the solutions found are better than or tie the well-know best solutions by other heuristic methods in previous literature. While the improvement may be too small to be insignificant, our limited experience suggests that the penalty guided CPSO find solutions which are of a quality and are comparable to that of other heuristic algorithms. The proposed method achieves the optimal/near optimal solution for all problems in this work more effectively and efficiently. Furthermore, guided by penalty function, it allows particles to move to more appropriate search area, thus improving some drawbacks in some studies by using PSO related approach.

References:

[1] Chen, T.-C. (2006). Penalty Guided PSO for Reliability Design Problems. In PRICAI 2006: Trends in Artificial Intelligence, pp. 777-786.

[2] Chen, T.-C. (2005): An immune algorithm based approach for reliability design problems. in *Proceeding of International Conference on Operations and Supply Management*, Bali, Indonesia .

[3] Coello, Carlos A. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. Computers in Industry, 41(2), pp. 113-127.

[4] Fu, J. F., Fenton, R. G., & Cleghorn, W. L. (1991). A mixed integer-discrete-continuous programming method and its application to engineering design optimization. Engineering Optimization, 17, pp. 263-280.

[5] Gen, Mitsuo, & Yun, YoungSu. (2006). Soft computing approach for reliability optimization: State-of-the-art survey. Reliability Engineering & System Safety, 91(9), pp. 1008-1026.

[6] Gopal, Krishna, Aggarwal, K. K., & Gupta, J. S. (1978). An Improved Algorithm for Reliability Optimization. IEEE Transactions on reliability, R-27(5), pp. 325-328.

[7] Ha, Chunghun, & Kuo, Way. (2006). Reliability redundancy allocation: An improved realization for nonconvex nonlinear programming problems. European Journal of Operational Research, 171(1), pp. 24-38.

[8] He, Qie, & Wang, Ling. (2007). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Engineering Applications of Artificial Intelligence, 20(1), pp. 89-99.

[9] Hikita, M., Nakagawa, Y., Harihisa, H. (1992), “Reliability optimization of systems by a surrogate constraints algorithm,” *IEEE Transactions on Reliability*, R-41(3), pp. 473-480.

[10]Hsieh, Y.-C., Chen, T.-C., Bricker, D.L. (1998), “Genetic algorithms for reliability design problems,” *Microelectronic Reliability*, 38, pp. 1599-1605.

[11]Joines, J. A., & Houck, C. R. (1994). On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. Paper presented at the Proceedings of the First IEEE Conference on Evolutionary Computation.

[12]Kuo, W., & Zuo, M.J. (2002). Optimal Reliability Modeling: Principles and Applications: John Wiley and Sons.

[13] Li, H. L., & Papalambros, P. (1985). A production system for use of global optimization knowledge. ASME Journal of Mechanisms, Transmission, and Automation, 107, pp. 277-284.

[14] Li, W. C., & Azarm, S. (1990). Optimality and constrained derivatives in two-level design optimization. ASME Journal of Mechanical Design, 112, pp. 563-568.

[15]Nakagawa, Y., Miyazaki, S. (1981), “Surrogate constraints algorithm for reliability optimization problems with two constraints,” *IEEE Transaction on Reliability*, R-30, 175-180.

[16]Potter, M. A., & A., K. (1994). A Cooperative Coevolutionary Approach to Function Optimization. Lecture Notes in Computer Science, 866.

[17] Prasad, V. R., & Kuo, W. (2000). Reliability optimization of coherent systems. Reliability, IEEE Transactions on, 49(3), pp. 323-330.

[18] Salajegheh, E., & Vanderplaats, G. N. (1993). Optimum design of trusses with sizing and shape variables. Structural and Multidisciplinary Optimization, 6(2), pp. 79-85.

[19] Sandgren, E. (1990). Nonlinear integer and discrete programming in mechanical design optimization. ASME Journal of Mechanical Design, 112, pp. 223-229.

[20] Shih, C. J., & Lai, T. K. (1995). Mixed-discrete fuzzy programming for nonlinear engineering optimization. Engineering Optimization, 23, pp. 187-199.

[21] Singiresu, S. Rao, & Ying, Xiong. (2005). A Hybrid Genetic Algorithm for Mixed-Discrete Design Optimization. Journal of Mechanical Design, 127(6), pp. 1100-1112.

[22] Van Den Bergh, F., & Engelbrecht, A. P. (2004). A Cooperative approach to particle swarm optimization. Evolutionary Computation, IEEE Transactions on, 8(3), pp. 225-239.

[23]Xu, Z., Kuo, W., & Lin, H.H. (1990),

"Optimization limits in improving system reliability," *IEEE Transactions on Reliability*, R-39, pp. 51-60.

[24]Yokota, T., Gen, M., & Li, Y.-X. (1996), "Genetic algorithm for non-linear mixed integer programming problems and its application," *Computers and Industrial Engineering*, 30(4), pp. 905-917.