© 2006 UICEE

An optimum design course supported by the particle swarm optimisation algorithm for undergraduate students

Wen-Jye Shyr

National Changhua University of Education
Changhua, Taiwan

ABSTRACT: An optimum design course is being taught to undergraduate students as a basic discipline in engineering and technology education. Optimum design aims primarily at determining the best possible combination of solutions used as design parameters to maximise or minimise an optimisation problem. The development of conventional optimum approaches has dramatically influenced modern teaching technologies. In this article, the author proposes the particle swarm optimisation algorithm for use in an optimum design course. An intensive course on optimum design is supported by the particle swarm optimisation algorithm for undergraduate students. A new concept, combining the particle swarm optimisation algorithm with conventional material, is introduced. Upon completion of this course, students can explain the basic concepts and terminologies associated with particle swarm optimisation. Students can also utilise relevant software on the particle swarm optimisation algorithm in order to solve optimum design problems. Three words can summarise the main features of the proposed approach: *faster*, *cheaper* and *simpler*.

INTRODUCTION

Almost every engineering design problem can be formulated as an optimisation problem. Solving an optimisation problem requires the computation of the global maxima or minima of the object function. Obviously, reaching this goal makes the search process complicated and the selection of an optimum technique critical. It is a challenge for engineers to design efficient and cost-effective systems without compromising the integrity of the system. The conventional design process depends upon the designer's intuition, experience and skill.

Many optimisation algorithms have been developed and adapted for various problems. Methods to solve the general optimisation problem have been studied for many years and considerable literature exists [1][2]. Engineering optimal design studies can often be cast in terms of optimisation problems. However, for such an approach to be worthwhile, designers must be content that the optimisation techniques employed converge fast. In this article, the author describes recent convergence problem studies found when applying the particle swarm optimisation algorithm to those optimisation problems often found in design. The particle swarm optimisation algorithm has exhibited good function optimisation performance. Particle swarm optimisation is an invented high performance optimiser that is very easy to understand and implement. It is similar, in some ways, to genetic algorithms, but requires less computational bookkeeping and generally only a few lines of code [3].

OPTIMUM DESIGN PROBLEM FORMULATION

The aim of the optimum design course is to identify the best possible combination of solutions for use as design parameters in order to maximise or minimise an optimisation function. In this course, it is generally assumed that various preliminary analyses have been completed and a detailed design concept or sub-problem must be carried out. Students should bear in mind that a considerable number of analyses have to be performed before reaching this design optimisation problem stage.

This must be stressed because the optimum solution will only be as good as the formulation. Once the problem is properly formulated, good software is usually available to solve it. In this article, the author uses optimum design software supported by a genetic algorithm for undergraduate students.

Figure 1 shows the formulation procedure for design optimisation problems involving the translation of a descriptive statement of the problem into a well defined mathematical statement.

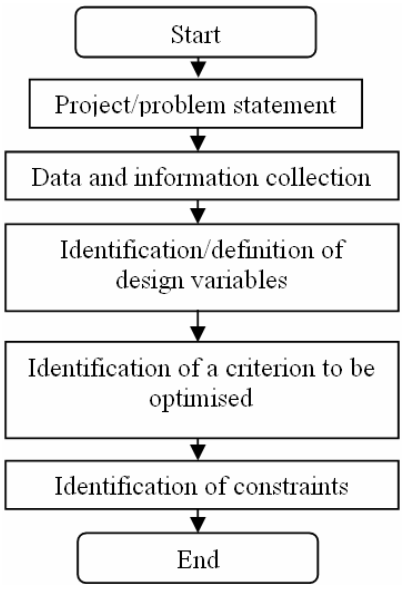

Figure 1: The formulation procedure for design optimisation problems.

The detailed formulation procedure steps are as follows:

- *Step 1: Project/problem statement*: The formulation process begins by developing a descriptive statement for the project/problem. This is usually performed by the project sponsor or leader. The statement describes the overall objectives of the project and the requirements to be met;
- *Step 2: Data and information collection*: To develop a mathematical formulation of the problem, students need to gather the material properties, performance requirements, resource limits and other relevant information. Some of the design data and expressions may depend upon design variables that are identified in the next step;
- *Step 3: Identification/definition of design variables*: The next step in the formulation process is to identify a set of variables that describes the system, called the design variables. These design variables should be independent of each other as much as possible;
- *Step 4: Identification of a criterion to be optimised*: There can be many feasible designs for a system and some are better than others. Criteria are necessary to compare different designs for the same problem solution. The criterion must be a scalar function whose numerical value can be obtained once a design is specified. Such a criterion is usually called an objective function for the optimum design problem, which needs to be maximised or minimised depending on the problem's requirements;
- *Step 5: Identification of constraints*: All restrictions placed on a design are collectively called constraints. The final step in the formulation process is to identify all constraints and develop expressions for them. All of these and other constraints must depend on the design variables, since only then do their values change with different trial designs [4].

PARTICLE SWARM OPTIMISATION ALGORITHM

The particle swarm optimisation (PSO) algorithm is a recently invented high performance optimiser that possesses several highly desirable attributes, including the fact that the basic algorithm is very easy to understand and implement. Particle swarm optimisation is a population-based stochastic optimisation technique that was developed by Drs Eberhart and Kennedy in 1995. This concept was inspired by the social behaviour of birds flocking or fish schooling [5][6].

Particle swarm optimisation shares many similarities with evolutionary computation techniques, such as genetic algorithms. The system is initialised with a population of random solutions and searches for optima by updating generations. However, unlike genetic algorithms, particle swarm optimisation has no evolution operators like crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space following the current optimum particles. Each particle keeps track of its coordinates in the problem space, which are associated with the best solution (fitness) it has achieved so far (the fitness value is also stored). This value is called *pbest*. Another *best* value that is tracked by the particle swarm optimiser is the best value obtained so far by any particle among the neighbours of that particle. This location is called *lbest*. When a particle considers the population as its topological neighbours, the best value is a global best and is called *gbest*. The particle swarm optimisation concept consists of, at each time step, changing the velocity of (accelerating) each particle towards its p*best* and *lbest* locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration towards *pbest* and *lbest* locations [7].

IMPLEMENTATION OF THE PARTICLE SWARM OPTIMISATION ALGORITHM

As stated before, the particle swarm optimisation (PSO) algorithm simulates the behaviour of flocking birds. Imagine the following scenario: a group of birds are randomly searching for food in an area. There is only one piece of food in the area being searched. None of the birds knows where the food is; however, they do know how far away the food is in each search iteration. What is the best strategy for finding the food? An effective strategy is to follow the bird nearest the food.

Particle swarm optimisation was learned from this scenario and used to solve optimisation problems. In PSO, each single solution is a *bird* in the search space and is called a *particle*. All of the particles have fitness values that are evaluated by the fitness function to be optimised. Each particle has velocities that direct particle flight. The particles fly through the problem space following the paths of current optimum particles.

In PSO, instead of using genetic operators, each particle (individual) adjusts its *flight* according to its own flight experience and the experience of its companions. Each particle is treated as a point in a D-dimensional space. The *ith* particle is represented as $X_I = (x_{i1},x_{i2},\ldots, x_{iD})$. The best previous position (the position giving the best fitness value) for the *ith* particle is recorded and represented as $P_I = (p_{i1},p_{i2},\ldots, p_{iD})$. The index of the best particle among all the particles in the population is represented by the symbol *g*. The rate of position change (velocity) for particle *i* is represented as $V_I = (v_{i1},v_{i2},\ldots, v_{iD})$. The particles are manipulated according to the following equations:

$$v_{id} = w * v_{id} + c_1 * rand\,() * (pid - x_{id}) + c_2 * Rand\,() * (p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

where c_1 and c_2 are two positive constants, c_1 and c_2 are usually $c_1 = c_2 = 2$, *rand( )* and *Rand( )* are two random functions in the range [0, 1], and *w* is the inertia weight. Equation (1) is used to calculate the particle's new velocity according to its previous velocity and the distances from its current position from its own best experience (position) and the group's best experience. The particle then flies towards a new position according to equation (2).

The performance of each particle is measured according to a predefined fitness function related to the problem to be solved. The inertia weight *w* is employed to control the impact of the previous history of velocities on the current velocity. This influences the trade-off between global (wide-ranging) and local (nearby) exploration abilities of the *flying points*. A larger inertia weight *w* facilitates global exploration (searching new areas) while a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area. A suitable selection of the inertia weight *w* can provide a balance between global and local exploration abilities and thus require fewer iterations to find the optimum.

In this article, an analysis of the impact of this inertia weight, together with the maximum velocity allowed on particle swarm optimisation performance, is given, followed by experiments that illustrate the analysis and provide some insights into optimal selection of the inertia weight and the maximum velocity allowed.

The pseudo code of the procedure is as follows:

For each particle
Initialise particle
END
Do
For each particle
Calculate fitness value
If the fitness value is better than the best fitness value (pBest) in history set current value as the new pBest
End
Choose the particle with the best fitness value of all the particles as the gBest
For each particle
Calculate particle velocity according equation (1)
Update particle position according equation (2)
End
While maximum iterations or minimum error criteria is not attained

Particles' velocities on each dimension are clamped to a maximum velocity *Vmax*. If the sum of accelerations would cause the velocity on that dimension to exceed *Vmax*, which is a parameter specified by the user, the velocity on that dimension is then limited to *Vmax*.

THE TEACHING METHOD

The lectures are held in a optimum design laboratory. The teacher explains the concepts. Examples of the particle swarm optimisation algorithm are executed and projected demonstrating the behaviour of different optimum design problems. Time is left for students to run some examples with different parameters realising an interactive learning process.

The optimum design laboratory serves active problem solving and is tightly attached to the theoretical material. Each student works on his/her own computer and solves the optimum design problems by himself/herself. The teacher sets up the problem and provides guidance. Students develop particle swarm optimisation algorithm programs or combine programs from given software. The results are then evaluated. Additional programs are also provided. The examination is held in the optimum design laboratory and consists of solving an optimum problem assigned by the teacher. The solution is accepted only if the program works correctly.

DEMONSTRATION PROVIDES A BETTER UNDERSTANDING

To verify particle swarm optimisation algorithm performance, three optimal design objective functions are considered. These are detailed below.

Example 1: Simple Evaluation Function

Equation (3) is a simple evaluation function for the particle swarm optimisation (PSO) algorithm as follows:

$$f(x) = x_1{}^2 + x_2{}^2 \tag{3}$$

where $[x_{1\min,} x_{1\max}] = [-1, \quad 2]$, and $[x_{2\min,} x_{2\max}] = [-1, \quad 2]$.

The genetic algorithm (GA) is used to evaluate the optimisation problem in [8]. The evaluation function is the driving force behind the GA. The evaluation function is called from the GA to determine the fitness of each solution string generated during the search. A simple example evaluation function is described in ref. [8], as follows:

$$function[val, x] = gaDemo1Eval(x, parameters) \tag{4}$$
$$val = x_1{}^2 + x_2{}^2;$$

To run the *ga* using this test function, either of the following function calls from MATLAB should be used:

$$bstX = ga([-1 \quad 2; -1 \quad 2], 'gaDemo1Eval') \tag{5}$$
$$bstX = ga([-1 \quad 2; -1 \quad 2], 'x_1{}^2 + x_2{}^2');$$

where gaDemo1Eval.m is contains the evaluation function as given above. The defining parameters in [8] are as follows: population size=20, probability of crossover=0.6, probability of mutation=0.005.

The parameters created for particle swarm optimisation (PSO) algorithm are list in Table 1. The simulation of search space via genetic algorithm is depicted in Figure 1. Figure 2 presents the fitness function performance via the PSO algorithm.

Table 1: The parameters created for the PSO algorithm.

Particle swarm optimisation (PSO) algorithm	
Population size	20
Initial inertia weight	0.9
Final inertia weight	0.2
Iterations	50

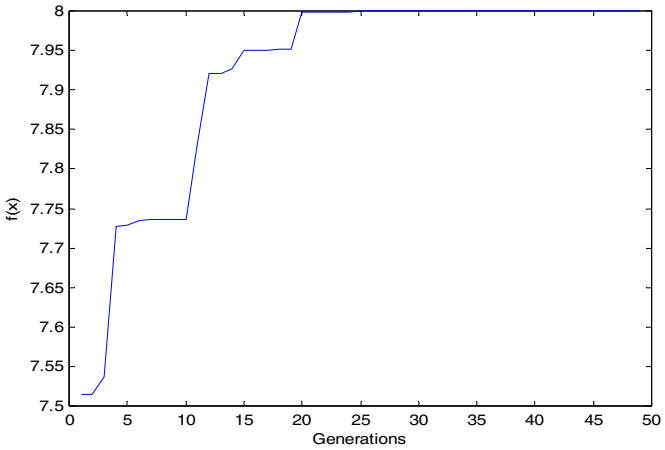

Figure 1: Fitness variation in GA (example 1).

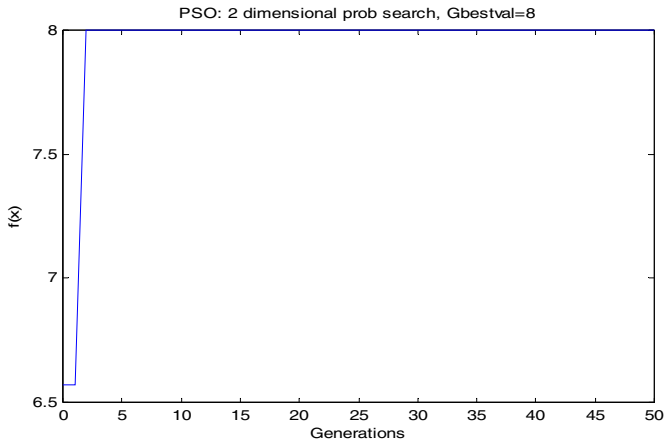

Figure 2: Fitness function performance via PSO (example 1).

Table 2 provides a set of results for the evaluation function being optimised. Each algorithm is executed until the maximum value is found. As can be seen from Table 2, both of

the algorithms perform well at finding the optimal solution. Therefore, in terms of the metric for solution quality, there seems little to distinguish between the three algorithms. However, when the number of generations is taken into account, there are significant differences in the required number of iterations to obtain the solution. The PSO algorithm is shown to converge faster for function optimisation. Also, when comparing Figure 1 with Figure 2, the PSA algorithm achieves faster convergence than the genetic algorithm.

Table 2: Comparison of particle swarm optimisation (PSO) algorithm and genetic algorithm (GA).

Optimal Value / Function	Fitness f(x)		x_1		x_2	
	GA	PSO	GA	PSO	GA	PSO
Simple evaluation function	8	8	2	2	2	2

Example 2: The Problem of Optimal Design

The genetic algorithm was used for the design optimisation problem in Lindfield and Penny [9]. The objective function of engineering design problems is described below.

A manufacturer wishes to produce a wall mounting container that consists of a hemisphere surmounted by a cylinder of fixed height. The height of the cylinder is fixed, but the common radius of the cylinder and hemisphere may vary between two and four. The manufacturer wishes to identify the radius value that maximises the container volume. This optimisation problem can be formulated by taking *r* as the common radius of the cylinder and hemisphere and *h* as the height of the cylinder. Taking *h*=2 units leads to the objective function in equation (6) as follows:

$$\textit{Maximize} \quad v = \left(2*\pi*r^3\right)/3 \; + \; 2*\pi*r^2 \qquad (6)$$

$$\textit{where} \quad 2 \le r \le 4.$$

The defining parameters using the genetic algorithm in Lindfield and Penny are as follows: population size=10, probability of crossover=0.6, probability of mutation=0.005 [9]. The simulation results in the genetic algorithm are depicted as follows: a common radius of the cylinder *r* is equal to 3.8667 and the objective function *v* is equal to 215.0202. The above genetic algorithm shows the algorithm for the design of optimisation problems, but there exists premature convergence and long convergence times. Figure 3 shows the objection function performance via the PSO algorithm. The simulation results in the PSO algorithm are depicted as follows: common radius of the cylinder *r* is equal to four and the objective function *v* is equal to 234.5723.

Comparing the result in Figure 3 with the genetic algorithm, the PSO algorithm achieves faster convergence than that proposed in Lindfield and Penny [9]. The good performance obtained by the PSO algorithm shows improved behaviour over that employed in the genetic algorithm. The PSO algorithm is also shown to converge faster for function optimisation.

CONCLUSION

In this article, the particle swarm optimisation (PSO) algorithm was applied to three numerical test functions for optimal design. The PSO algorithm obtained good performance. The PSO algorithm also converged faster for function optimisation. In comparing the proposed particle swarm optimisation algorithm with the genetic algorithm, the particle swarm optimisation algorithm presented the following advantages:

- *Faster*: the PSO algorithm can obtain the same quality results in significantly fewer fitness evaluations;
- *Cheaper*: the PSO algorithm is intuitive and does not need specific domain knowledge to solve these numerical functions for optimisation;
- *Simpler*: while possessing similar capabilities as the genetic algorithm, the much simpler implementation and reduced bookkeeping of the PSO algorithm is appealing.

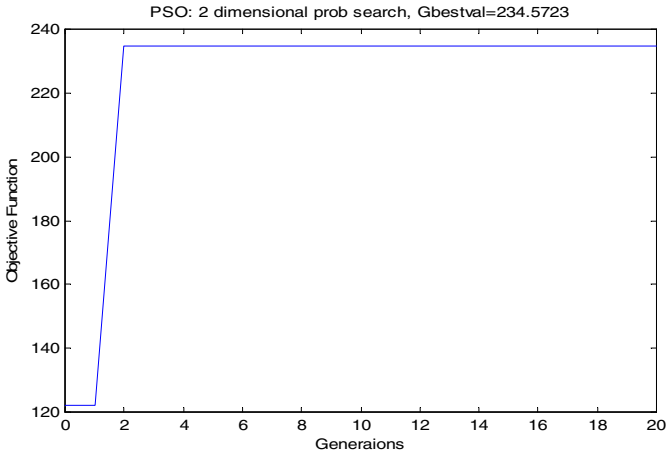

Figure 3: Fitness function performance via PSO.

Another reason that the particle swarm optimisation algorithm is attractive is that there are few parameters to adjust. The simpler the algorithm, the more people can take advantage of it! Eventually, it is hoped that the particle swarm optimisation approach will be helpful in optimising functions faster, cheaper, simpler and more effectively.

REFERENCES

1. Siddall, J.N., *Optimal Engineering Design: Principles and Applications*. New York: Marcel Dekker (1982).
2. Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading: Addison-Wesley Publishing (1989).
3. Shyr, W.J. and Liao, C.W., An optimum design course supported by a genetic algorithm for undergraduate students. *Proc. 4th Asia-Pacific Forum on Engng. and Technology Educ*., Bangkok, Thailand, 165-168 (2005).
4. Arora, J.S., *Introduction to Optimum Design* (2nd edn). San Diego: Elsevier Academic Press (2004).
5. Eberhart, R. and Kennedy, J., A new optimizer using particle swarm theory. *Proc. 6th Inter. Symp. Micro Machine and Human Science*, 39-43 (1995).
6. Kennedy, J. and Eberhart, R., Particle swarm optimization. *Proc. IEEE Inter. Conf. on Neural Networks*, 4, Perth, Australia, 1942-1948 (1995).
7. Kennedy, J. and Eberhart, R., *Swarm Intelligence*. San Francisco: Morgan Kaufmann (2001).
8. Christopher, R.H., Jeffery, A.J. and Michael, G.K., A Genetic Algorithm for Function Optimization: a MATLAB Implementation. North Carolina State University, IE TR 95-09, USA (1995).
9. Lindfield, G. and Penny, J., *Numerical Methods Using MATLAB*. Upper Saddle River: Prentice-Hall (1995).