Structural Design Optimization Using Particle Swarm Optimization and Its Variants

Structural Design Optimization Using Particle Swarm Optimization and Its Variants

D. Prayogo[1*], Sebastian[1*], W. Hauwing[1], F.T. Wong[1], Tjandra, D.[1]

[1]Department of Civil Engineering, Petra Christian University, Jalan Siwalankerto 121-131, Surabaya 60236, Indonesia

[*]Corresponding author email: prayogo@petra.ac.id ; m21416037@john.petra.ac.id

Abstract. Structural design optimization has become an extremely challenging and more complex task for most real-world practical applications. A huge number of design variables and complex constraints have contributed to the complexity and nonlinearity of the problems. Mathematical programming and gradient-based search algorithms cannot be used to solve nonlinear problems. Thus, researchers have extensively conducted many experimental studies to address the growing complexity of these problems. Metaheuristic algorithms, which typically use nature as a source inspiration, have been developed over past decades. As one of the widely used algorithms, particle swarm optimization (PSO) has been studied and expanded to deal with many complex problems. Particle swarm optimization and its variants have great accuracy in finding the best solution while maintaining its fast convergence behavior. This study aims to investigate PSO and its variants to solve a set of complex structural optimization problems. Several complex benchmark studies of design problem were provided to study the performance of PSO, linearly decreasing inertia weight PSO and bare bones PSO. The results support the potential use of PSO and its variants as an alternative approach to solving structural design optimization problems.

Keywords: optimization, engineering, algorithm, convergence

1. Introduction

Optimization has been used for minimizing building structure costs in the world of structural engineering. Engineers are challenged to solve many calculations during the process of designing, which is time-consuming. Each calculation problem has its own constraints, variables, and parameters that are usually complicated to be solved manually [1]. Designing aims to obtain the best result to get the minimum weight design while meeting certain code requirements, which can be achieved with optimization [8].

Particle swarm optimization (PSO) is a metaheuristic calculation method that aims to find the most optimum answer or objective from a case that has different parameters and constraints. This algorithm imitates living organisms; it mimics groups such as a flock of birds or a fish of school searching for food. When a bird finds its own best location, the pack eventually agrees with a global best location [2].

Although PSO has been proven to solve many types of problems, the optimum answer depends on the parameter that is set in the beginning [2]. This paper, therefore, discusses two modified PSO algorithms: Linearly Decreasing Inertia Weight PSO (LDW-PSO) [3] and Bare Bones PSO (BB-PSO) [5]. Also, this paper compares three types of PSO in order to find the most reliable and the fastest type

to obtain the most convergent answer. This study includes three examples of engineering design problems with different constraints and parameters.

2. Particle Swarm Optimization

Particle swarm optimization imitates the movement of a living organism as a particle that can find its own best solution following its own best location. Each time an organism finds its own best location, the whole population immediately finds a global best location. This global best location has the most optimum answer from all personal best solutions to get a global solution.

Our calculations start by defining parameters such as inertia weight parameter (w), the cognitive factor parameter (c_1), and the social factor parameter (c_2). Then, the location for each particle is generated randomly with defined upper bound and lower bound. Each particle later starts to search for the answer with a velocity (equation 1). For each iteration, the best location is updated using equation 1, 2.

$$v_i = w \times v_i + rand(0,1) \times c_1 \times (pbest(x_i) - x_i) + rand(0,1) \times c_2 \times (gbest(x_i) - x_i) \quad (1)$$

$$x_i = x_i + v_i \quad (2)$$

Table 1. Particle Swarm Optimization Algorithm.

Algorithm 1	
1.	Initialize PSO parameters
2.	Initialize a population of random particles (solutions)
3.	Evaluate the objective value of each particle
4.	Determine initial pbeat X and gbest X
5.	**while** termination criteria are not satisfied **do**
6.	**for** each particle **do**
7.	Update the velocity for the particle
8.	Update the new location for the particle
9.	Determine the objective value for the particle in its new location
10.	Update pbest X and pbest F if required
11.	**end** for
12.	Update gbest X and pbest F if required
13.	**end while**

3. Linearly Decreasing Inertia Weight Particles Swarm Optimization

Linearly decreasing inertia weight-PSO differs from the standard PSO. This modified PSO can linearly decrease its inertia weight (w) when each iteration finishes [3]. When the value of w is high, the ability to find global search increases. On the other hand, when the value of w decreases the ability to find local search increases [6]. The functions of velocity and location update are the same as in the original PSO, but the inertia weight is updated using Equation 3.

$$w_{\text{i}} = w_1 - (w_1 - w_2)(\text{iter})/(\text{maxiter}) \quad (3)$$

where w_1 and w_2 are the initial and end value of inertia weight, respectively, iter is the number of iterations, and max iter is the maximum number of iterations.

Table 2. Linearly Decreasing Inertia Weight Particles Swarm Optimization Algorithm.

Algorithm 2	
1.	Initialize PSO parameters
2.	Initialize a population of random particles (solutions)
3.	Evaluate the objective value of each particle
4.	Determine initial pbeat X and gbest X
5.	**while** termination criteria are not satisfied **do**
6.	Update inertia weight
7.	**for** each particle **do**
8.	Update the velocity for the particle
9.	Update the new location for the particle
10.	Determine the objective value for the particle in its new location
11.	Update pbest X and pbest F if required
12.	**end** for
13.	Update gbest X and pbest F if required
14.	**end while**

4. Bare Bones Particles Swarm Optimization

Different from the abovementioned types of PSO, Bare Bones PSO ignores all parameters and does not need to use velocity to find a new location. Bare Bones PSO mainly uses Gaussian distribution. The new location is updated based on the location, which is the mean between the personal best solution and the global best solution. The formula is shown in Equations 4, 5, and 6.

$$\mu = \frac{pbest+gbest}{2} \tag{4}$$

$$\sigma = |pbest - gbest| \tag{5}$$

$$x(i+1) = \begin{cases} N(\mu,\sigma) & if(\omega > 0.5) \\ pbest & else \end{cases} \tag{6}$$

Table 3. Bare Bones Particles Swarm Optimization Algorithm

Algoritm	
1	Initialize PSO parameters
2	Initialize a population of random particles (solutions)
3	Evaluate the objective value of each particle
4	Determine initial pbeat X and gbest X
5	**while** termination criteria are not satisfied **do**
6	**for** each particle **do**
7	Determine the objective value for the particle in its new location
8	Update pbest X and pbest F if required
9	**end** for
10	Update gbest X and pbest F if required
11	**end while**

5. Test Problem and Results of Particles Swarm Optimization

This section presents three cases that were solved using three types of PSO. Each case addresses an engineering design problem, which has different constraints and parameters.

5.1. Case 1-A three-bar truss design

This case involves a 3-bar planar truss structure, as shown in Figure 1 [7]. The weight of the structure minimizes subject to stress constraint on each bar element. The objective function of this case is to find the optimal value of cross-sectional areas (A_1, A_2). The function of this case is given in Equations 7-10.

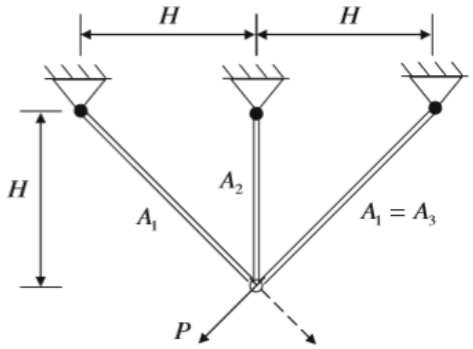

Figure 1. Three-bar truss.

Minimize: $f(A_1, A_2) = \left(2\sqrt{2}\,A_1 + A_2\right) \times l$ (7)

Subject to

$$g_1 = \frac{\sqrt{2}A_1 + A_2}{\sqrt{2}A_1^2 + 2A_1A_2} P - \sigma \leq 0 \quad (8)$$

$$g_2 = \frac{A_2}{\sqrt{2}A_1^2 + 2A_1A_2} P - \sigma \leq 0 \quad (9)$$

$$g_3 = \frac{1}{A_1 + \sqrt{2}A_2} P - \sigma \leq 0 \quad (10)$$

Where

$$0 \leq A_1 \leq 1 \text{ and } 0 \leq A_2 \leq 1;$$

$$l = 100 \text{ cm},$$
$$P = 2\,\text{KN/cm}^2$$
$$\sigma = 2\,\text{KN/cm}^2$$

Table 4 shows the statistical result for the best objective value by the three methods. Table 5 compares the results obtained by the three methods and the algorithm used in a previous study. The previous study used another algorithm called Cuckoo Search (CS) [4]. As it is seen, not only the BB-PSO obtained the best result, the results obtained by the other types of PSO were better than those of the previous study that used a different algorithm. Figure 2 shows the convergence behavior of each method.

Table 4. Statistical result for Case 1.

	Best	Avg	Worst	SD	Time (sec)
PSO	263.8959	263.9829	264.7531	0.187233	0.166044
LDW-PSO	263.8959	266.4239	282.8427	6.550067	0.159809

	Best	Avg	Worst	SD	Time (sec)
BB-PSO	263.8959	263.8985	263.9198	0.004515	0.085384

Table 5. Comparison of optimization result for Case 1.

	PSO	LDW-PSO	*BB-PSO*	Gandomi *et al.* [4]
A_1	0.79494	0.78789	0.78902	0.78867
A_2	0.39080	0.41046	0.40727	0.40902
g1	-0.22099	-0.20478	-0.09537	-0.00029
g2	-1.26311	-1.25670	-1.16958	-0.26853
g3	-0.95788	-0.94808	-0.92579	-0.73176
f	263.92410	263.89640	263.89593	263.9716

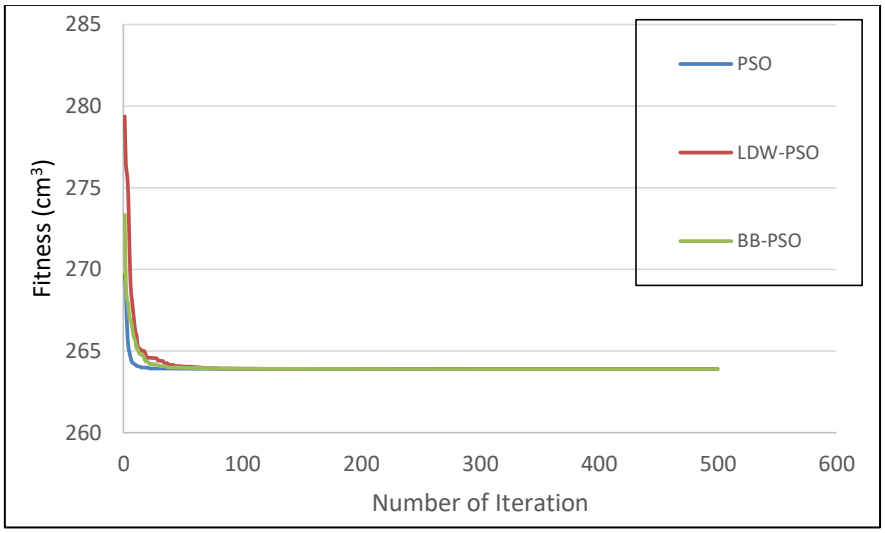

Figure 2. Convergence behavior for Case 1 for each algorithm.

5.2. Case 2-Tubular column design

Figure 3 shows a tubular column that receives an axial load (P) of 2500 kg [8]. The column material has a yield stress (ry) of 500 kg/cm^2, a modulus of elasticity (E) of 0.85E-06 kg/cm^2, and a density (q) of 0.0025 kg/cm3. The length (L) of the column is 250 cm. This case is aimed to find the minimum cost of the column (f) that includes material and construction cost which taken as Equation 11. The constraint and the optimization function are given in Equations 11-17.

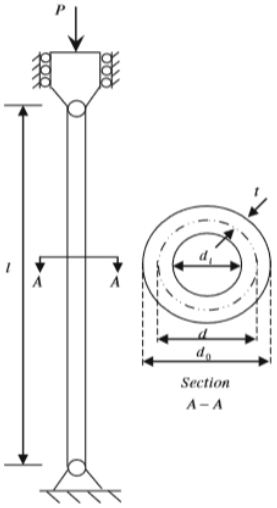

Figure 3. The tubular column

Minimize: $f(\mathrm{d},\mathrm{t}) = 9.8dt + 2d$ (11)

Boundary conditions: $2 \le d \le 14$, $0.2 \le t \le 0.8$

Subject to:

$$g_1 = \frac{P}{\pi dt\sigma_y} - 1 \le 0 \tag{12}$$

$$g_2 = \frac{8PL^2}{\pi^3 Edt(d^2+t^2)} - 1 \le 0 \tag{13}$$

$$g_3 = \frac{2.0}{d} - 1 \le 0 \tag{14}$$

$$g_4 = \frac{d}{14} - 1 \le 0 \tag{15}$$

$$g_5 = \frac{0.2}{t} - 1 \le 0 \tag{16}$$

$$g_6 = \frac{t}{0.8} - 1 \le 0 \tag{17}$$

Table 6 represents the statistical result for the best objective value by the three methods and the algorithm used in the previous study. Table 7 compares the results obtained by the three methods and the algorithm used in the previous study. The results obtained by PSO and its variants were also better than those of the previous study [4] like in Case 1. Figure 4 shows the convergence behavior of each algorithm.

Table 6. Statistical result for Case 2.

	Best	Avg	Worst	SD	Time (sec)
PSO	26.4995	26.4995	26.4995	4.28E-11	0.168175
LDW-PSO	26.4995	26.8337	31.5127	1.271885	0.164555
BB-PSO	26.4995	26.4995	26.4995	3.79E-08	0.082097

Table 7. Comparison of optimization result for Case 2.

	PSO	LDW-PSO	*BB-PSO*	Gandomi *et al.* [4]
d	5.45116	5.45116	5.45116	5.45139
t	0.29196	0.29196	0.29196	0.29196
g1	-3.33066e-16	-2.22045e-16	-1.80071e-09	-0.0241
g2	0	-2.22045e-16	-2.15574e-10	-0.1095
g3	-0.63310	-0.63310	-0.63310	-0.6331
g4	-0.61063	-0.61063	-0.61063	-0.6106
g5	-0.31499	-0.31499	-0.31499	-0.3150
g6	-0.63504	-0.63504	-0.63504	-0.6351
f	26.49950	26.49950	26.49950	26.53217

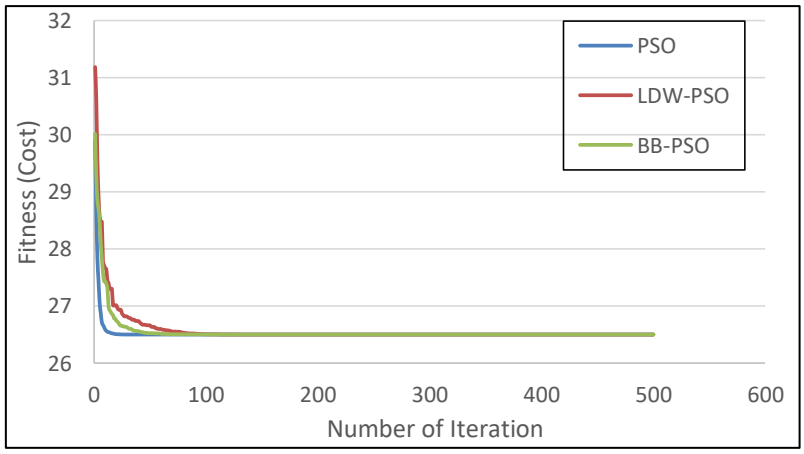

Figure 4. Convergence behavior for Case 2 for each algorithm.

5.3. Case 3-Tension/Compression Spring

Figure 5 shows a spring design with three variables, which are wire diameter (x_1), mean coil diameter (x_2), and the number of active coils (x_3). The objective of this case is to find the minimum tension/compression spring weight. The function and constraint are defined in Equations 18-22.

Minimize: $f(X) = (x_3 + 2)x_2 x_1^2$ (18)

Subject to:

$$g_1(X) = 1 - \frac{x_2^2 x_3}{71785 x_1^4} \leq 0 \quad (19)$$

$$g_2(X) = \frac{4x_2^2 - x_1 x_2}{12566 x_2 x_1^3 - x_1^4} + \frac{1}{5108 x_1^2} - 1 \leq 0 \quad (20)$$

$$g_3(X) = 1 - \frac{140.45x_3}{x_2^2 x_3} \leq 0 \tag{21}$$

$$g_4(X) = \frac{x_2 + x_1}{1.5} - 1 \leq 0 \tag{22}$$

With boundary conditions: $0.05 \leq x_1 \leq 2$, $0.25 \leq x_2 \leq 1.3$, $2 \leq x_3 \leq 15$.

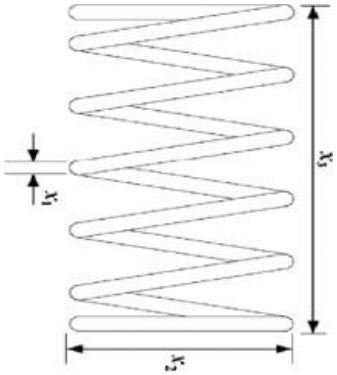

Figure 5. Tension/compression spring design problem.

Table 8 represents the statistical result for the best objective value by the three methods. Table 9 compares the results obtained by the three methods. Figure 6 shows the convergence behavior of each algorithm.

Table 8. Statistical result for Case 3.

	Best	Avg	Worst	SD	Time (sec)
PSO	0.004895	0.00467	0.00574	0.000418	0.1689
LDW-PSO	0.004869	0.00510	0.00574	0.000391	0.1773
BB-PSO	0.004869	0.00487	0.00488	2.57E-06	0.0881

Table 9. Comparison of optimization result for Case 3.

	PSO	LDW-PSO	*BB-PSO*
$\mathbf{x_1}$	0.05000	0.05000	0.05000
$\mathbf{x_2}$	0.37389	0.37443	0.37401
$\mathbf{x_3}$	3.20934	3.20012	3.20744
g1	0.99972	0.99972	0.99972
g2	-0.82619	-0.82619	-0.82619
g3	-56179	-56179	-56179
g4	-0.93333	-0.93333	-0.93333
f	0.004895	0.004869	0.004869

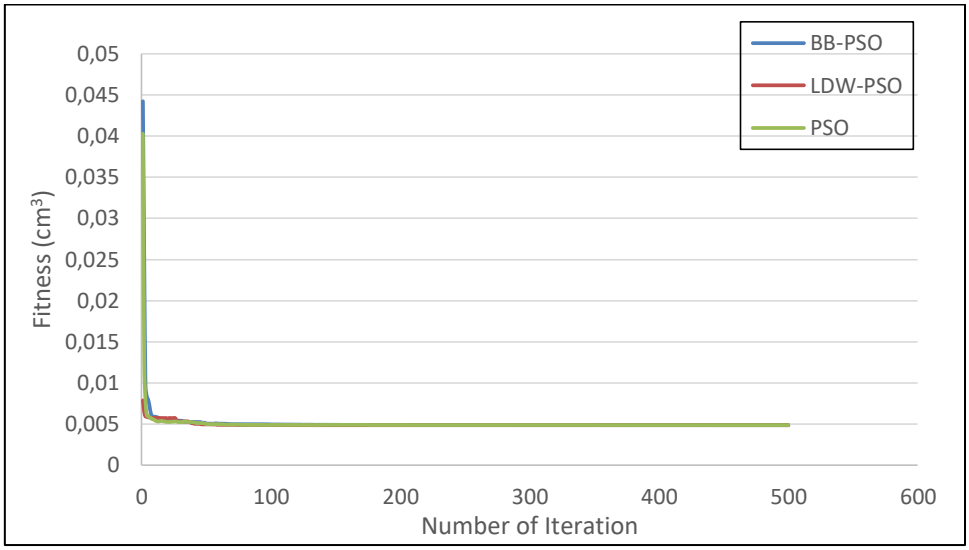

Figure 6. Convergence behavior for Case 3 for each algorithm.

6. Conclusion

This paper compared the results of three problem cases that were optimized with three different types of PSO. The results showed that, with the same number of iterations for each case and each PSO, BB-PSO had the fastest calculation time and always gave the best result. Meanwhile, the standard PSO did not give the best results. The BB-PSO algorithm also had the smallest standard deviation, which means that each iteration had a more stable result and faster to obtain convergence in the results. Since the normal PSO had the smallest standard deviation, the performance of the algorithm would still depend on the cases and the parameter of each case.

References

[1] Cheng M-Y, Prayogo D, Wu Y-W, and Lukito M M 2016 *Autom. Constr.* **69** pp 21-33
[2] Kennedy J and Eberhart R 1995 *Proc. of IEEE Int. Conf. on Neural Networks (Perth)* (New York: IEEE) pp 1942-8
[3] Xin J, Chen G and Hai Y 2009 *Joint Conf. on Computational Sciences and Optimization, CSO (Sanya)* (New York: IEEE) pp 505-8
[4] Gandomi A H, Yang X-S, and Alavi A H 2013 *Engineering With Computers* **29** 245-245
[5] Guo J and Sato Y 2017 *Int. J. Networked Distrib. Comput.* **5** 143
[6] Shi Y and Eberhart R 1998 *IEEE World Cong. On Computational Intelligence Evolutionary Computation Proc. (Anchorage)* (New York: IEEE) pp 69-73
[7] Nowcki H 1974 *Computer Applications in the Automation of Shipyard Operation and Ship Design ed* Y Fujita *et al*(New York: Elsevier) pp 327-38
[8] Rao S S 1996 *Engineering Optimization: Theory and Practice* 3rd edn (Chichester: John Wiley & Sons)
[9] Hare W, Nutini J and Tesfamariam S 2013 *Adv. Eng. Softw.* **59** 19-28