Springer

Journal of Mechanical Science and Technology 23 (2009) 426~434

Journal of Mechanical Science and Technology

www.springerlink.com/content/1738-494x
DOI 10.1007/s12206-008-0917-4

An integrated method of particle swarm optimization and differential evolution [†]

Pyungmo Kim[1] and Jongsoo Lee[2,*]

[1]*Department of Mechanical Engineering, Yonsei University, Seoul, 120-749, Korea*
[2]*School of Mechanical Engineering, Yonsei University, Seoul, 120-749 Korea*

(Manuscript Received February 19, 2008; Revised July 21, 2008; Accepted September 17, 2008)

Abstract

Particle swarm optimization (PSO) and differential evolution (DE) have their similarities and compatibility in the design update process, such that a new design vector is determined by using neighborhood designs under algorithm control parameters. The paper deals with an integrated method of a hybrid PSO (HPSO) algorithm combined with DE in order to refine the optimization performance. PSO and DE also possess common characteristics compared with genetic algorithm (GA). The crossover- and mutation-like operators are suggested in the HPSO. A bounce back method is also implemented to prevent the design from locating out of design spaces during the optimization process. For the purpose of further enhancing the search capabilities, such HPSO is combined with the Q-learning that is one of efficient reinforcement learning methods. Using a number of nonlinear multimodal functions and engineering optimization problems, the proposed algorithms of HPSO and HPSO with Q-learning are compared with PSO, DE and GA.

Keywords: Particle swarm optimization; Differential evolution; Hybrid method; Bounce back; Q-learning

1. Introduction

Particle swarm optimization (PSO) [1] has been developed as one of the efficient global optimization tools. It has distinct characteristics such that the design variable is represented by the particle in a swarm and considered as continuous variable, while the design variable that corresponds to a chromosome in genetic algorithm (GA) is discrete. PSO is easy to implement with a small number of algorithm control parameters compared with GA as well. There has recently been considerable attention given to PSO for mechanical and structural design optimization [2-5]. The design update in PSO is available by using a neighborhood and previously obtained best designs. Such an update method is similar to that in differential evolution (DE), which is also one of the easily implemented global optimization techniques [6-9]. A new design in the DE process is determined by using randomly selected two position vectors and the best design found within the current generation.

Based on the compatibility between PSO and DE, the present study proposes a method of hybrid PSO, so-called HPSO that combines the design update formula adopted from both PSO and DE. Some of the algorithm control parameters used in PSO and DE are also similar to operators used in GA as well. The comparison of PSO with DE and GA is summarized in Table 1. One can find the similarities and differences among three optimization algorithms in terms of type of design parameters and method of design update. The present study develops an integrated method of HPSO and compares its optimization performance with traditional PSO, DE and genetic algorithm [10, 11]. There have been a number of studies in the comparison and combinations of PSO and DE,

† This paper was recommended for publication in revised form by Associate Editor Tae Hee Lee
*Corresponding author. Tel.: +82 2 2123 4474, Fax.: +82 2 362 2736
E-mail address: jleej@yonsei.ac.kr

Table 1. Comparison PSO with other algorithms.

	PSO	DE	GA
Design parameters	Particle	Position vector	Chromosome
	Continuous	Continuous	Discrete
	Swarm	Population	Population
Design update	Neighbors & previous design	Crossover & Mutation	Crossover & Mutation
Learning from	Global best & local best	Local best	Reproduction

[12-16] where PSO and DE are separately operated or control parameters in one algorithm are simply added to the other. The present study suggests an integrated approach that includes PSO and DE algorithms in the whole process.

A conventional optimization process would be normally conducted with predetermined values of control parameters such as crossover and mutation values. The search capability can be improved if control parameters are adaptively selected according to the progress of optimization results; a better design solution could be obtained with a variety of control parameter values. The present study employs the Q-learning technique [17] in order to intelligently select control parameters during the HPSO process, thereby enhancing the optimization result. Using a number of nonlinear multi-modal functions and engineering optimization problems, the proposed algorithms of HPSO and HPSO with Q-learning are compared with conventional PSO, DE and GA.

2. Optimization algorithms

2.1 Particle swarm optimization

Particle swarm optimization (PSO) is one of the global optimization algorithms that works with a few of the control parameters used in the design update behavior. Suppose a swarm of particles: the position of a particle is represented by x_i^k ($i = 1,...,n$), where n is the number of particles in the k-th generation of the swarm. In PSO, such a value is updated to a new position, x_{i+1}^k as follows [1]:

$$x_{i+1}^k = x_i^k + v_{i+1}^k \tag{1}$$

$$v_{i+1}^k = \chi \times [v_i^k + rand(0,AC) \times (p_i^k - x_i^k) + rand(0,AC) \times (p_g - x_i^k)] \tag{2}$$

where, v_{i+1}^k is a new velocity vector calculated from the current values (x_i^k and v_i^k) and a number of control parameters such as χ and AC. The design value, p_i^k is the 'local best' obtained during the k-th generation of the swarm, while the design, p_g is the 'global best' found among all the particles in the swarms up to the current generation. Such a global best is replaceable whenever the fittest design is found at the new generation of the PSO process. The design p_i^k or p_g is to PSO as the elitist is to GA. The two above equations imply that a new design is searched toward the global optimum by using the velocity vector that is explored based on both the local and global bests. Thus, the design update in PSO is conducted via the learning from neighbors and previous design.

2.2 Differential evolution

Differential evolution (DE) is also one of global optimization algorithms that progressively searches for the updated design based on the following equations [8, 18]:

$$x_{i+1}^k = x_i^k \quad \text{for } rand(.) < CR \tag{3}$$

$$x_{i+1}^k = v_i^k \quad \text{otherwise}$$

$$v_i^k = x_{best}^k + \phi \times (x_{r1}^k - x_{r2}^k) \tag{4}$$

where, CR is the crossover ratio that decides the threshold of the design update. The velocity vector-like value, v_i^k is determined by using randomly selected two position vectors, x_{r1}^k, x_{r2}^k and the best design, x_{best}^k, found within the current generation. Since two position vectors are selected at random, they function as a mutation operator. A parameter, ϕ, ranging between 0.0 and 1.0 is used for the relaxation of the design. It is noted that x_{best}^k in DE is very similar to p_i^k in PSO. However, DE does not adapt the design from the ancestors.

2.3 Drawback in PSO

The unconstrained Rastrigin's function with 10 design variables in Eq. (5) is tested by using a conventional PSO.

$$f(x) = \sum_{i=1}^{n} -x_i \cdot \sin(\sqrt{|x_i|}), \tag{5}$$

$$-500 \le x_i \le 500, \qquad i = 1,...,10$$

The function minimization is conducted ten times with different random seeds as initial particles of a

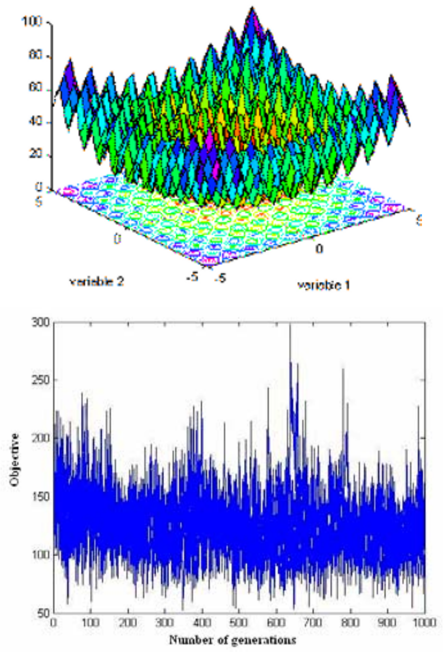

Fig. 1. Rastrigin's function and design histories by PSO.

swarm, and their results are shown in Fig. 1. There detect the considerable numerical oscillations without any convergence [16, 19, 20]. Such phenomena come from the fact that designs are selected out of the design boundaries (i.e., design space) due to the random number term, $rand(0, AC)$ in Eq. (2). That is, by combining Eq. (2) with Eq. (1), the updated design, x_{i+1}^k could be located outside the design space. Now, it is necessary for PSO to search the design within the defined design space, thereby inducing design convergence.

3. Proposed method

3.1 Another PSO

Another version of PSO is expressed with the following equations [1, 5]:

$$x_{i+1}^k = x_i^k + v_{i+1}^k \tag{6}$$

$$v_{i+1}^k = \chi \times v_i^k + c_1 \times rand(.) \times (p_i^k - x_i^k) + c_2 \times rand(.) \times (p_g - x_i^k) \tag{7}$$

$$\text{if } v_i^k > v_{\max}\text{, then } v_i^k = v_{\max} \qquad \text{if } v_i^k < -v_{\max}\text{, then } v_i^k = -v_{\max} \tag{8}$$

where, the algorithm control parameters of χ, c_1, c_2 in Eq. (7) are introduced instead of χ, AC in Eq. (2). The conditions of Eq. (8) are the limits on the over-positioned design that are described in an earlier section.

3.2 Hybrid PSO

PSO and DE have their compatibility in the design update process such that a new design vector is determined by using neighborhood designs under algorithm control parameters. The present study suggests a new method combining PSO with DE, the so-called hybrid PSO (HPSO). The main part of the proposed HPSO algorithm can be written as follows:

$$\begin{aligned}
&\text{For } rand(.) < CR \\
&x_{i+1}^k = x_i^k + v_{i+1}^k \\
&v_{i+1}^k = \chi \times v_i^k + c_1 \times rand(.) \times (p_i^k - x_i^k) \\
&\qquad\qquad + c_2 \times rand(.) \times (p_g - x_i^k) \\
&\text{For } rand(.) > CR\text{, } v_i^k = p_i^k + \phi \times (x_{r1}^k - x_{r2}^k) \\
&\qquad \text{For } rand(.) < MR\text{, } x_{i+1}^k = x_i^k \\
&\qquad \text{For } rand(.) > MR\text{, } x_{i+1}^k = v_i^k .
\end{aligned} \tag{9}$$

The above algorithm employs the same control parameters that are used in both PSO and DE. The HPSO process is conducted first according to the crossover ratio, *CR,* whether PSO or DE is activated. The sub-process of PSO is formulated by Eq. (7) instead of Eq. (2). In the sub-process of DE, the design update is controlled by mutation ratio, *MR*. The sub-process of PSO does not require *MR* since such operation is replaced by two terms of $c_1 \times rand(.)$ and $c_2 \times rand(.)$.

3.3 Bounce back

When a particle keeps moving to another position, it may reach outside of the design boundary so that the numerical oscillation and consequent divergence of design solutions occur. The present study proposes a method such that the over-positioned particle should be bounced back within a specific region, the so-called resolution region. That is, such 'bounce back' facilitates particles to be relocated into the design boundary. The condition for the bounce back is as follows:

$$\begin{aligned}
&\text{If } (x_i^k > x_{\max}) \text{ then} \\
&\qquad x_i^k = v_{\max} \times [BB + rand(.) \times (1.0 - BB)]
\end{aligned} \tag{10}$$

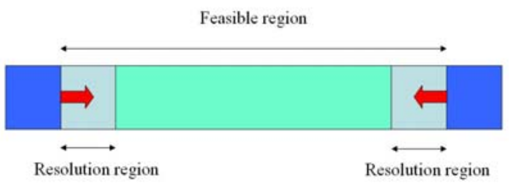

Fig. 2. Bounce back.

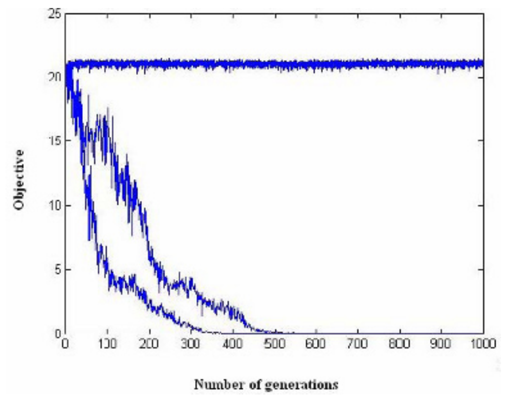

Fig. 3. PSO without bounce back.

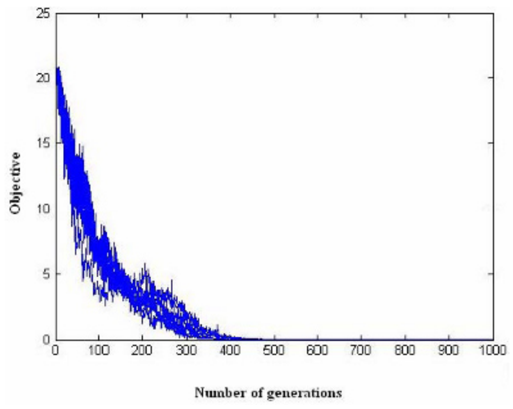

Fig. 4. PSO with 20% bounce back.

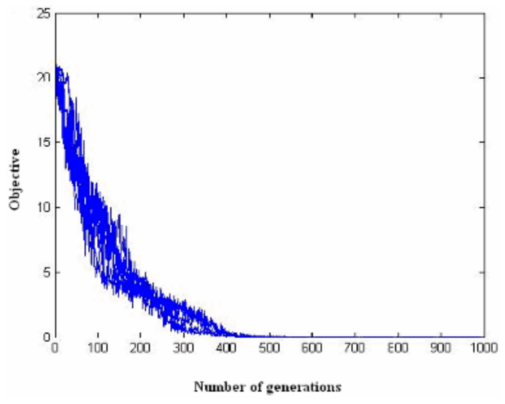

Fig. 5. PSO with 10% bounce back.

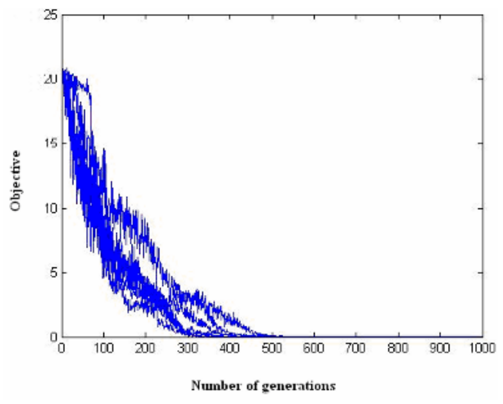

Fig. 6. PSO with 5% bounce back.

where, the bounce back parameter, *BB* ranging between 0 and 1 is used. The maximum limit on the distance movement is also imposed by a factor, $v_{\max}$.

It is noted that the condition of Eq. (8) is slightly modified as Eq. (10). This method may cause a reduction in the search speed and the loss of diversity in global optimization. However, the resolution regions obtained from Eq. (10) are near design boundaries as shown in Fig. 2. In a case where the optimal solution is to be located close to design boundaries, this method is helpful.

A PSO-based numerical experiment for the bounce back is conducted by using Ackley's function as shown in Figs. 3 to 6. Each result is obtained with three to five different random seeds as initial particles. The result with the bounce back ratio of *BB*=20% in Fig. 4 gives the best performance. The solution convergence with *BB*=20% is quite independent of the initial swarm, while other results still depend upon the bounce back ratio.

4. Hybrid PSO with reinforcement learning

4.1 General description

Reinforcement learning (RL) is an intelligent machine learning technique dealing with how an agent takes actions under a certain environment so as to maximize some notion of long-term reward. RL aims to find the optimal policy that maps states to the actions the agent should take in those states [21]. In the context of RL, the Q-learning works with incomplete information Markovian action problems based on the action value function Q that maps state-action pairs to expected returns. Q-learning successively improves its evaluations of particular actions at particular states. An agent conducts an action at a particular state and evaluates its performance in terms of the reward or penalty received from the policy and its corresponding state value. The aim of the agent is not merely to maximize its instantaneous reward in the current state, but to maximize the cumulative reward received during the overall period of time.

The search with an updated velocity vector in HPSO is dependent upon the selection of control parameters such as $\chi, c_1, c_2, \phi, CR, MR$ and *BB*. The ordinary HPSO process would be performed with given values of control parameters. The motivation behind the HSPO with Q-learning is to increase the search capabilities via adaptively selecting such control parameters based on the Q-learning techniques. The subsequent section discusses how the HPSO is implemented with Q-learning process.

4.2 HPSO with Q-Learning

A particle in HPSO moves to a new position based on the local best (p_i^k), global best (p_g) and difference between randomly selected two vectors of $x_{r1}^k - x_{r2}^k$ under control parameters such as $\chi, c_1, c_2, \phi, CR, MR, BB$ as shown in Eqs. (9) and (10). The present study considers such three values as the three-dimensional state since they are the most effective decision parameters that influence the new velocity and position of a particle. Three components of the state are as follows:

$$s_1 = p_i^k - x_i^k \tag{11}$$

$$s_2 = p_g - x_i^k \tag{12}$$

$$s_3 = x_{r1}^k - x_{r2}^k \tag{13}$$

A Q-table is used to match the relation between the three-dimensional state, (s_1, s_2, s_3) and three actions, (*MR*, *CR*, *BB*) out of seven control parameters. In a case where an agent begins to participate into Q-learning, it chooses action parameter values, and works with the policies (i.e., HPSO algorithm). Each of three cells in action has three levels in the present study so that a total of $3 \times 3 \times 3$ cells are used for a particle. After a new design and its corresponding fitness value of the optimization problem are obtained, the Q-table is then evaluated as to whether its values receive the reward according to the new state.

For the initialization and updating of the Q-table, the following normalization conditions should be satisfied [17]:

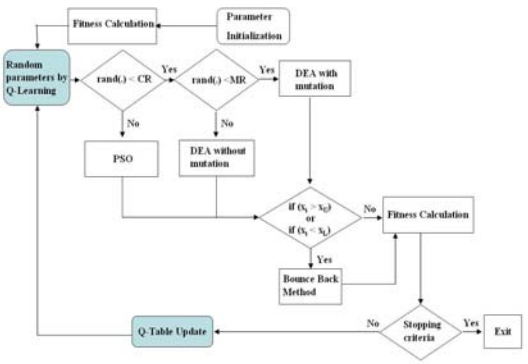

Fig. 7. Process of HPSO with Q-learning.

$$\text{For state } s_1, s_2, s_3 \text{ and action: } CR,\ MR,\ BB \qquad \sum_{action} Q^k(state, action) = 1\ . \tag{14}$$

The Q-table values are initially set to equal one for all agents, and the reward assignment is then conducted in this approach. After the HPSO with Q-learning, the values of a cell according to the appropriate state and action are changed by using Eq. (15).

$$Q^k(state: s_1, s_2, s_3, \quad action: CR, MR, BB) = \frac{1}{2} Q^k(state: s_1, s_2, s_3) + \frac{1}{2}(f(x_i^k) - f(x_i^{k+1})) + R \tag{15}$$

where, *R* is a positive value of 'reward' which is to be added to the current Q-values in a case where when the selected action parameters outperform the best design of the whole swarm. The unconstrained or constrained objective function value at x_i^k is represented by $f(x_i^k)$. All of cells in the Q-table should be normalized every learning process. The next procedure is to select the rule (i.e., HPSO algorithm of Eq. (9)) parameters for the update movement. HPSO parameters for an agent are selected according to its Q-table. In the process of HPSO with Q-learning, three action components are renewed for each of the individual particles.

The overall process of HPSO and HPSO with Q-learning is shown in Fig. 7, wherein HPSO using Eqs. (9) and (10) and the additional Q-learning with Eqs. (11) to (15) are identified.

5. Results of example problems

The proposed optimization strategies are examined

Table 2. Control parameter conditions.

	PSO	DE	GA	HPSO	HPSO with Q-learning
# of generations	1000	1000	1000	1000	1000
Population size	50	50	50	50	50
Crossover (or CR)	-	0.2	0.8	0.2	0.2~0.3
Mutation (or MR)	-	-	0.05	0.5	0.4~0.6
Bounce back ratio	20%	20%	-	20%	10~20%
Others	χ=0.729 AC=2.07	ϕ=0.5 CR=0.2	-	C_1=1.5 C_2=1.5 χ=0.729 ϕ=0.2	C_1=1.5 C_2=1.5 χ=0.729 ϕ=0.2

with three uni-modal/multi-modal function minimization problems and a few engineering optimization problems. Control parameters used in each of optimization algorithms are shown in Table 2.

Sum of different power function (F1)

$$f(x) = \sum_{i=1}^{n} |x_i|^{i+1} \tag{16}$$
$$-1.0 \le x_i \le 1.0, \quad i = 1,...,10$$

Rastrigin's function (F2)

$$f(x) = \sum_{i=1}^{n} -x_i \cdot \sin(\sqrt{|x_i|}) \tag{17}$$
$$-500 \le x_i \le 500, \quad i = 1,...,10$$

Ackley's path function (F3)

$$f(x) = -20\exp(-0.2\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}) - \exp(\frac{\sum_{i=1}^{n}\cos(2\pi x_i)}{n}) + 20 + \exp(1.0) \tag{18}$$
$$-32.768 \le x_i \le 32.768, \quad i = 1,...,10$$

The above three functions with 10 design variables have their global optima at $x_i = 0.0$ in common. Even though Rastrigin's function and Ackley's path function present the nonlinear multimodality, the present study locates the global optimum.

Three-bar planar truss (F4)

A well-known 3-bar planar truss structure is considered. The design objective is to determine the optimal cross sectional areas, X_1 and X_2, by minimizing the total weight (W) of a statically loaded three-bar planar truss subjected to stress (σ) constraints on each of the truss members together with the upper limit of tip deflection (δ). A schematic is shown in Fig. 8 and the mathematical statement of this optimization problem [22] is written as follows:

Table 3. Results of three-bar truss problem.

	PSO	DE	GA	HPSO	HPSO with Q-learning
X_1	0.83	0.84	0.83	0.83	0.81
X_2	0.33	0.35	0.33	0.34	0.32
W(X)	15.3	15.3	15.3	15.2	15.1

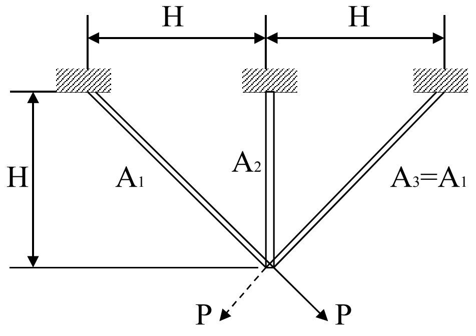

Fig. 8. Three-bar planar truss.

$$\begin{aligned} &\text{Minimize} && W(X_1, X_2) \\ &\text{subject to} && \sigma_1(X_1,X_2) \le \sigma^{upper} = 20 \\ &&& \sigma_2(X_1,X_2) \le \sigma^{upper} = 20 \\ &&& \sigma_3(X_1,X_2) \ge \sigma^{lower} = -15 \\ &&& \delta \le \delta_{\max} = 0.2 \\ &&& 0.1 \le X_1, X_2 \le 5.0\ . \end{aligned} \tag{19}$$

Optimization results for the three-bar truss problem are compared in Table 3 and Fig. 9. Most of optimization methods successfully locate the near optimal solution as shown in Table 3. Especially, the PSO with Q-learning provides the smallest level of numerical oscillation during convergence as shown in Fig. 9.

Piston lever (F5)

The design objective of this problem is to locate the

Table 4. Results of piston oil problem.

	PSO	DE	GA	HPSO	HPSO with Q-learning
x_1	133.3	129.4	250.0	135.5	134.7
x_2	2.44	2.43	3.96	2.48	2.51
x_3	117.14	119.80	60.03	116.62	118.47
D	4.75	4.75	5.91	4.75	4.61
f(x)	970.16	968.80	1059.20	974.79	923.97

piston components, x_1, x_2, x_3, and D by minimizing the oil volume when the lever of the piston is lifted up from 0deg to 45deg as shown in Fig. 10 [23]. The formal optimization statement is given as follows:

$$\text{Minimize} \quad f(x_1,x_2,x_3) = \frac{1}{4}\pi D^2(b-a) \tag{20}$$

$$\begin{aligned}\text{Subject to} \quad & QL\cos\theta - RF \le 0 \quad at \quad \theta = 45\deg \\ & Q(L-x_3) - M_{\max} \le 0 \\ & 1.2(b-a) - a \le 0 \\ & D/2 - x_2 \le 0 \\ & R = \frac{|-x_3(x_3\sin\theta + x_1) + x_1(x_2 - x_3\cos\theta)|}{\sqrt{(x_3-x_2)^2 + x_1^2}} \\ & F = \pi P D^2/4 \\ & a = \sqrt{(x_3-x_2)^2 + x_1^2} \\ & b = \sqrt{(x_3\sin 45^0 + x_1)^2 + (x_2 - x_3\cos 45^0)^2} \\ & 0.05 \le x_1, x_2, D \le 500, \quad 0.05 \le x_3 \le 120\end{aligned}$$

where, the payload is Q=10,000lbs, the lever is L=240in, the maximum allowable bending moment of the lever is $M_{\max} = 1.8\times10^6 lbs\cdot in$, and the oil pressure is given as 1,500psi. A number of inequality constraints are imposed--force equilibrium, maximum bending moment of the lever, minimum piston stroke--and geometrical conditions are considered.

Similar trends in the results of the piston oil problem can be detected as shown in Fig. 11 and Table 4. The PSO with Q-learning approaches to the optimized solution at the earliest generations with the smallest numerical oscillation. HPSO is moderate, and a conventional PSO is the worst among them.

The performance of each of the algorithms is also summarized in terms of the number of function evaluations as shown in Table 5. Each of the example problems are run 20 times to obtain the statistical results. In most of the example problems, HPSO with Q-learning and HPSO demonstrate remarkable performance in terms of mean and standard deviation (SD) of the number of function evaluations. It is noted that the function, f2, turns out to be prematurely converged (PC), that is, fails to locate the optimal solution when PSO, DE and GA are used.

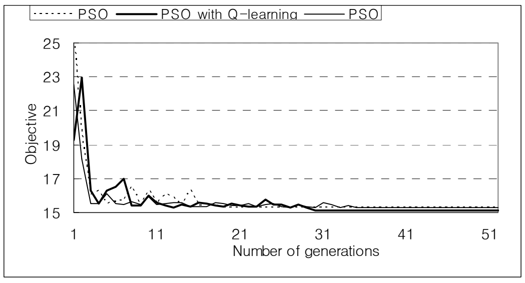

Fig. 9. Convergence history of three-bar truss problem.

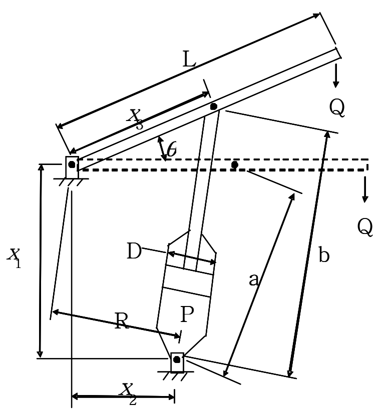

Fig. 10. Piston oil problem.

6. Closing remarks

The paper explores an integrated hybrid PSO (HPSO) algorithm combined with the DE process in order to refine the optimization performance. To further increase the search capabilities, the proposed HPSO is combined with the Q-learning so as to adaptively select the algorithm control parameters. Optimization results are examined by using PSO, DE, GA, HPSO and HPSO with Q-learning. The proposed optimization strategies are examined with a number of uni-modal and nonlinear multi-modal function minimization problems and engineering optimization problems. The PSO with Q-learning provides the best optimum, and its result gives the smallest level of numerical oscillation during convergence. The PSO with Q-learning also generates the optimized solution at the earliest generations with the smallest numerical

Table 5. Number of function evaluations.

Problem		PSO	DE	GA	HPSO	HPSO with Q-learning
F1	Max	158	86	102	53	36
	Min	98	60	74	40	20
	Mean	131	60	74	40	20
	SD	21.4	9.3	14.8	4.2	5.9
F2	Max	PC	PC	PC	615	518
	Min				478	215
	Mean				537	372
	SD				49.9	111.5
F3	Max	462	223	PC	102	113
	Min	356	181		83	137
	Mean	417	201		93	122
	SD	33.5	12.5		6.2	8.3
F4	Max	42	53	52	41	60
	Min	37	36	37	17	31
	Mean	38	48	40	29	39
	SD	1.9	6.4	4.6	6.7	11.1
F5	Max	294	199	216	197	168
	Min	122	159	161	162	129
	Mean	166	187	185	187	151
	SD	51.7	14.2	18.2	13.4	13.4

SD: standard deviation
PC: Prematurely converged

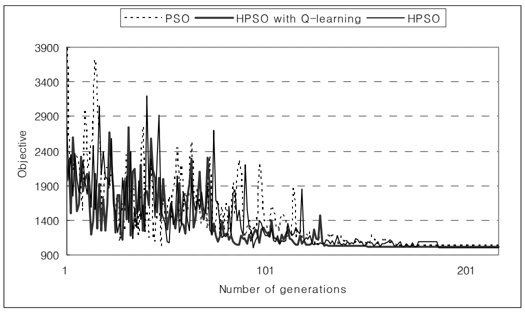

Fig. 11. Convergence history of piston oil problem.

oscillation as well. Such advanced methods present more reliable results in terms of the mean and standard deviation on the number of function evaluations. As further research in this context, it would be more valuable to adopt the various reproduction and crossover characteristics of GA into the proposed approach.

References

[1] J. Kennedy and R. C. Eberhart, Particle Swarm Optimization, Proceedings of the IEEE International Conference on Neural Networks, 4 (1995) 1942-194.

[2] P. C. Fourie and A. A. Groenwold, The Particle Swarm Algorithm in Topology Optimization, Proceedings of the 4th World Congress of Structural and Multidisciplinary Optimization, pp. 52-53, Dalian, China, 2001.

[3] A. A. Groenwold and P. C. Fourie, The Particle Swarm Optimization in Size and Shape Optimization, Structural and Multidisciplinary Optimization, 23 (2002) 259-267.

[4] J. F. Schuttle and A. A. Groenwold, A Study of Global Optimization Using Particle Swarms, Journal of Global Optimization, 31 (1) (2005) 93-108.

[5] J. F. Schuttle, B.-I. Koh, J. A. Reinbolt, R. T. Haftka, A. D. George and B. J. Fregly, Evaluation of a Particle Swarm Algorithm for Biomechanical Optimization, ASME Transactions, Journal of Biomechanical Engineering, 27 (2005) 465-474.

[6] R. M. Storn and K. V. Price, Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Space, Journal of Global Optimization, 11 (4) (1997) 341-359.

[7] J. A. Lampinen and I. Zelinka, Mechanical Engineering Design Optimization by Differential Evolution, MaGraw-Hill's Advanced Topics in Computer Science Series: New Ideas in Optimization, pp. 127-146, McGraw-Hill, United Kingdom, 1999.

[8] K. V. Price, R. M. Storn and J. A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer, 2005.

[9] S. Das, A. Konar and U. K. Chakraborty, Improved Differential Evolution Algorithms for Handling Noisy Optimization Problems, Proceedings of the 2005 IEEE Congress on Evolutionary Computation, 2 (2005) 1691-1698.

[10] J. Lee, S. Kim and S. Kang, Evolutionary Fuzzy Modeling in Global Approximate Optimization, International Journal of Vehicle Design, 28 (3) (2002) 81-97.

[11] J. Lee and S. Kim, Optimal Design of Engine Mount Rubber Considering Stiffness and Fatigue Strength, Proceedings of the Institution of Mechanical Engineers, Part D – Journal of Automobile Engineering, 221 (7) (2007) 823-835.

[12] T. Hendtlass, A Combined Swarm Differential Evolution Algorithm for Optimization Problems, Lecture Notes in Computer Science, 20 (2001) 11-18.

[13] W. J. Zhang and X. F. Xie, DEPSO: Hybrid Particle Swarm with Differential Evolution Operator, Proceeding of IEEE International Conference on Systems, Man and Cybernetics, 4 (2003) 3816-3821.
[14] A. Stacey, M. Jancic and I. Grundy, Particle Swarm Optimization with Mutation, Proceedings of Congress on Evolutionary Computation (CEC ‘03), 1425-1430, 2003.
[15] H. J. Meng, P. Zheng, R. Y. Wu, X. J. Hao and Z. Xie, A Hybrid Particle Swarm Algorithm with Embedded Chaotic Search, Proceedings of Conference on Cybernetics and Intelligent Systems, 1 (2004) 367-371.
[16] T. O. Ting, M. V. C. Rao and C. K. Loo, A Novel Approach for Unit Commitment Problem Via an Effective Hybrid Particle Swarm Optimization, IEEE Transactions on Power Systems, 21 (1) (2006) 411-418.
[17] M. Khajenejad, F. Afshinmanesh, A. Marandi and B. N. Araabi, Intelligent Particle Swarm Optimization Using Q-Learning, Proceedings of IEEE Swarm Intelligence Symposium, Indianapolis, IN, 2006.
[18] J. Ilonen, J.-K. Kamarainen and J. A. Lampinen, Differential Evolution Training Algorithm for Feed-Forward Neural Networks, Neural Processing Letters, 17 (1) (2003) 93-105.
[19] A. Ratnaweera, S. K. Halgamuge and H. C. Watson, Self-Organizing Hierarchical Particle Swarm Optimizer with Time-Varying Acceleration Coefficients, IEEE Transactions on Evolutionary Computation, 8 (3) (2004) 240-255.
[20] T. Takahama and S. Sakai, Solving Constrained Optimization Problems by the ε-Constrained Particle Swarm Optimizer with Adaptive Velocity Limit Control, Proceedings of IEEE Conference on Cybernetics and Intelligent Systems, 1425-1430, 2006.
[21] J.-S. Jang, C.-T. Sun and E. Mizutani, Neuro-Fuzzy and Soft Computing, Prentice Hall, Upper Saddle River, NJ, 1997.
[22] R. T. Haftka and Z. Gürdal, Elements of Structural Optimization, Kluwer Academic Publishers, The Netherlands, 1993.
[23] G. N. Vanderplaats, DOT (Design Optimization Tools) User’s Manual, Version 4.20, VR&D, 1995.

Jongsoo Lee received a B.S. degree in Mechanical Engineering from Yonsei University in 1988. He then went on to receive his M.S. degree from University of Minnesota in 1992 and Ph.D. degree from Rensselaer Polytechnic Institute in 1996. Dr. Lee is currently a Professor at the School of Mechanical Engineering at Yonsei University in Seoul, Korea. He is currently serving as a committee member of the division of CAE and Applied Mechanics in the Korean Society of Mechanical Engineers. Dr. Lee’s research interests are in the area of engineering design optimization, fluid-structure interactions, and reliability based robust product design.