A Modified Particle Swarm Optimization For Engineering Constrained Optimization Problems

Choosak Pornsing, Kawinthorn Saichareon, and Thanathorn Karot

***Abstract*—**This paper presents a modified particle swarm optimization (PSO) algorithm for solving engineering constrained optimization problem. The proposed PSO presents time-varying inertia weight swarm topology technique. The novelty of this proposed method is that the whole swarm may divide into many sub-swarms in order to find a good source of food or to flee from predators. This behavior engages the particles disperse through the search space and the sub-swarm with the worst performance dies out while that with the best performance grows by producing offspring. Furthermore, the penalty function method was used to handle the constraints of the problem. The penalty factor is a self-adaptive function in which based on the performance of the last iteration. Numerical experiments based on well-known constrained engineering design problem show that the modified method outperforms other competitive algorithms from literature.

***Keywords*—**constrained optimization problem, particle swarm optimization, swarm intelligence.

I. Introduction

Generally, a constrained optimization problem can be stated as follows.

$$\text{Find } \boldsymbol{X} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix} \text{ which minimize } f(\boldsymbol{X}) \tag{1}$$

subject to the constraints

$$g_j(\boldsymbol{X}) \le 0, \quad j = 1, 2, \dots, m \tag{2}$$

$$l_j(\boldsymbol{X}) = 0, \quad j = 1, 2, \dots, p \tag{3}$$

where $\boldsymbol{X}$ is an *n*-dimensional vector called the *design vector*, $f(\boldsymbol{X})$ is termed the *objective function*, and $g_j(\boldsymbol{X})$and $l_j(\boldsymbol{X})$ are known as *inequality* and *equality* constraints, respectively. In a common practice an equality constraint $l_j(\boldsymbol{X})$ can be replaced by a couple of inequality constraints $l_j(\boldsymbol{X}) \le \delta$ and $l_j(\boldsymbol{X}) \ge -\delta$ (δ is a small tolerant amount). Therefore all constraints can be transformed to $M = m + 2p$ inequality constraints. We call $\boldsymbol{X}$ a feasible solution if it satisfies all the constraints.

Usually, a tradition mathematical programming method such as the Lagrange multiplier methods requires the gradient information of the objective function and relevant. Furthermore, the obtained solution often tends to be a local optimum unless the search space is convex. Recently, evolutionary computation (EC) techniques have attracted much attention for a variety of complex optimization method especially nonlinear constrained optimization problem and combinatorial optimization problem due to ECs do not need the objective function to be derivable or even continuous. ECs perform as global optimization techniques in which the cooperation of exploration and exploitation processes are well balanced.

In this study, we made use of a novel EC technique which founded by Eberhart and Kennedy [1] in 1995 to solve engineering constrained optimization problems, named *Particle Swarm Optimization (PSO)*. We proposed a modified PSO algorithm in order to mitigate a premature convergence problem in which yields a local optimum solution. Beside, we proposed a self-adaptive penalty function technique which the penalty factors are adjustable based on the level of constraints violation. Two engineering optimization problems were used to test the proposed method's performance compare to published results in literature.

The rest of this paper is organized as follows. Section 2 briefly describes conventional PSO. This section expresses the concept of PSO and its mathematical formulation. Section 3 describes the modified PSO in which attempting to alleviate the premature convergence problem. Section 4 reports the computational experiments with two well-known engineering optimization problems; minimization of the weight of a tension/compression spring design problem and minimization of the total cost of a pressure vessel design problem. A conclusion summarizing the contributions of this paper is in Section 5.

II. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is an evolutionary computing (EC) technique that belongs to the field of Swarm Intelligence proposed by Eberhart and Kennedy [1] inspired by the social behavior of bird flocking or fish schooling. PSO is an iterative algorithm that engages a number of simple entities−the particles−iteratively over the search space of some functions. The particles evaluate their fitness values with respect to the search function, at their current locations.

Dr. ChoosakPornsing, Department of Industrial Engineering and Management, Faculty of Engineering and Industrial Technology, Silpakorn University, Thailand: choosak@su.ac.th

Kawinthorn Saichareon, Department of Industrial Engineering and Management, Faculty of Engineering and Industrial Technology, Silpakorn University, Thailand: kawintho@gmail.com

Thanathorn Karot, Department of Industrial Engineering and Management Faculty of Engineering and Industrial Technology, Silpakorn University Thailand: ttkarot@gmail.com

Subsequently, each particle determines its movement through the search space by combining information about its current fitness, its best fitness from previous locations with regards to one or more members of the swarm (social perspective), with some random perturbations. The next iteration starts after the positions of all particles have been updated as in (4) and (5).

$$V_{id}^{t} = V_{id}^{t-1} + \phi_1\beta_1(pbest_{id}^{t} - X_{id}^{t}) + \phi_2\beta_2(gbest_{id}^{t} - X_{id}^{t}) \quad (4)$$

$$X_{id}^{t+1} = X_{id}^{t} + V_{id}^{t} \quad (5)$$

V_{id}^{t} is the current velocity of *i*th particle of the *d*-th dimension, V_{id}^{t-1} is the previous velocity of *i*th particle of the *d*th dimension, and X_{id}^{t} is the current position of *i*th particle of the *d*th dimension. ϕ_1 and ϕ_2 are weighted factors which are also called the *cognitive* and *social* parameter, respectively. β_1 and β_2 are random variables uniformly distributed within [0,1]. X_{id}^{t+1} is the position of *i*th particle of the *d*-th dimension in the next iteration. $pbest_{id}$ and $gbest_{id}$ are the *best personal position* ever visited by the particle and the *best global position* ever visited by all particles in the swarm.

$$pbest_{id}^{t} = \arg\min_{s \le t} f_i^s \quad (6)$$

$$gbest_{id}^{t} = \arg\min_{i} f(pbest_i^t) \quad (7)$$

Shi and Eberhart [2] introduced an inertia weight factor, ω, to the original PSO. Then, Shi and Eberhart [3] suggested that in order to obtain the best performance — the initial setting ω is set at some high velue (e.g. 0.9), which corresponds to particles moving in a low viscosity medium and performing extensive exploration — and the value of ω is then gradually reduced to some low value (e.g. 0.4). At this low value of ω the particles move in a high viscosity medium, perform exploitation, and are better at homing towheads local optima. Equation (4) is modified as below.

$$V_{id}^{t} = \omega V_{id}^{t-1} + \phi_1\beta_1(pbest_{id}^{t} - X_{id}^{t}) + \phi_2\beta_2(gbest_{id}^{t} - X_{id}^{t}) \quad (8)$$

Because of a number of advantages of PSO — rapid convergence towards an optimum, its quality of being easy to encode or decode, and being fast and easy to compute — it has been applied in many research areas such as global optimization, artificial neural network training, fuzzy system control, engineering design optimization, and logistics and supply chain management. Nonetheless, a number of researchers have noted that PSO tends to converge prematurely on local optima, especially in complex multimodal functions (Clerc and Kennedy [4], Parsopoulos and Vrahatis [6]). Accordingly, many papers have been proposed to improve PSO in order to avoid the problem of premature convergence.

III. Proposed PSO

A. The Modified PSO

The idea of the modified PSO manipulate a swarm animal behavior such that when the swarm is large, sub-swarms often form to find a good source of food or to flee from predators. This method supports a higher capability to find food sources or to flee from predation for the whole swarm. The swarm topology has been changed because the agents communicate within its sub-swarm instead of the whole swarm. As a result, the sub-swarms enable the whole swarm to conduct an extensive exploration of the search space. Instead of using the best position ever visited by all particles (*gbest*), the sub-swarms have their own *sbest* — the best position ever visited by particles in the sub-swarms. Equation (5) and (8) are modified as:

$$V_{sid}^{t} = \omega V_{sid}^{t-1} + \phi_1\beta_1(pbest_{sid} - X_{sid}^{t}) + \phi_2\beta_2(gbest_{sid} - X_{sid}^{t}) \quad (9)$$

$$X_{sid}^{t+1} = X_{sid}^{t} + V_{sid}^{t} \quad (10)$$

where the index *s* is the sub-swarm index. Subsequently, the sub-swarm with the capability to find a sufficient food source has a higher probability of survival. This example is more prominent in a school of fish in which the whole swarm fractions to sub-swarms in an effort to escape from a dangerous situation. The unlucky sub-swarm that is still stuck among predators will be exterminated.

Be exterminated, what would happen to the whole swarm if the worst sub-swarm disappeared from the flock? Reasonably, the animal swarm should still be able to produce their offspring and maintain its species in its environment. The particle swarm should also be able to maintain its swarm size in the system. Accordingly, we propose a process called an *extinction and offspring reproduction* process. In this process, the worst performance sub-swarm is periodically killed throughout the evolutionary, named as *extinction process*. Then, a sub-swarm in the system should be selected as an ancestor to produce offspring in what is known as *offspring reproduction process*. We shall borrow a selection method from the genetic algorithm. Although, there are many selection methods — e.g. roulette wheel, sigma scaling, Boltzmann, tournament selection — many researchers have found that the elitism method significantly outperforms the other methods (see Melanie [6]) and is therefore what has been used in this study.

Furthermore, we applied a *nonlinear decreasing inertia weight* approach (NLDIWA) which inspired by the study of Xu [7] in which the inertia weight factor was controlled by the following nonlinear decreasing function.

$$\omega^{t} = \omega_{ini} \times \frac{1+\cos(t\pi/T_{end})}{2} \quad (11)$$

where ω^t is the inertia weight of *t*th iteration, ω_{ini} is the initial inertia weight, and T_{end} is set as $0.95 \times T$, T is the maximum number of iterations.

B. Constraints Handling

Normally, in the evolutionary computation, the constraints are dealt with when evaluating solutions. The violated constraint of each solution is considered separately; the

relationship between infeasible solutions and feasible regions is exploited to guide search. However, the difficulty of this technique is there is no certain metric criterion to measure this relationship [8]. In the study of Michalewicz [9], he pointed that there are at least three choices to cope with this problems: count the number of violations for a given solution; consider the amount of infeasibility in terms of constraints violation; and compute the effort of repairing the individual.

In this study, we made use of the penalty function method in order to cope with the constraints. Constraints are incorporated into the objective function. Consequently, it transforms constrained problem into unconstrained problem. Thus, the constrained optimization problem in (1), (2), and (3) can be transformed as.

$$\emptyset_t = \emptyset(\boldsymbol{X}, r_t) = f(\boldsymbol{X}) + r_t \sum_{j=1}^{m} G_j[g_j(\boldsymbol{X})] \quad (12)$$

where G_j is some function of the constraint g_j, and r_t is a positive constant known as the *penalty parameter*. The solution may be brought to converge to that original problem stated in (1), (2), and (3) if the unconstrained minimization of the $\emptyset$ function is repeated for a sequence of the value of i_k $(t = 1,2, \dots, T)$. In this study, we made use of the interior formulation to form G_j as (see [10]):

$$G_j = -\frac{1}{g_j(\boldsymbol{X})} \quad (13)$$

The penalty factors are the key performance of this technique. We deployed an objective function ratio between the feasible *global best* and the infeasible particle. The penalty factor can be calculated as follows.

$$r_{t+1} = \frac{f(\boldsymbol{X})_t^*}{f(\boldsymbol{X})_t^\Delta} \quad (14)$$

where f, $f(\boldsymbol{X})^*$ is the objective value of non-violation particle, $(\boldsymbol{X})^\Delta$ is the objective value of the violation particle. It is noticeable that the particle of which severely violate a constraint yields very low value of objective function while the feasible *global best* particle yields higher value of objective function; consequently, a big number of r_t is received.

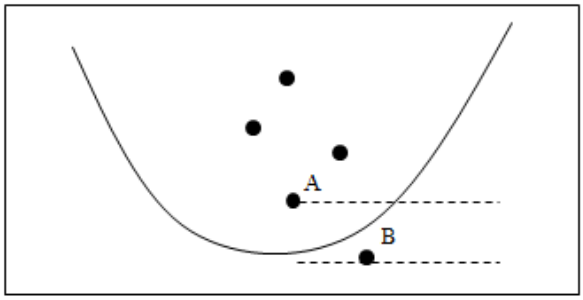

Fig. 1 The ratio between an infeasible solution and the feasible global best solution

The idea of r can be shown in Fig. 1. The particle B is the best solution of the t^{th} iteration; however, it is infeasible because it violates a constraint in constrained optimization problem. Thus, the *global best* among all particles in this iteration is particle A. Next, r_{t+1} is calculated for particle B in order to calculate $\emptyset(\boldsymbol{X}, r_{t+1})$ of the next iteration.

In summary, a PSO algorithm is proposed based on the adaptive parameters (nonlinear decreasing inertia weight) approach in which evaluate the performance of each sub-swarm is evaluated, and the worst performing sub-swarm is periodically eliminated. Offspring are produced form the best performing sub-swarms. Additionally, the constraints are handled by the penalty function method in which the penalty parameter is adjustable belong to the level of constraints violation. The algorithm of the proposed PSO is described in Table 1.

IV. Computational Experiments

In this section, two constrained engineering optimization problems that are commonly used in literature are performed.

A. Minimization of the Weight of a Tension/Compression Spring Design Problem

This problem was introduced by Belegundu [11] and Arora [12]. The objective is to minimize the weight of a tension/compression spring subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside and on design variables. The design variables are the mean coil diameter (*D*), the wire diameter (*d*), and the number of active coils (*N*). The problem can be described as follows:

$$\textbf{Minimize } f_1(\boldsymbol{X}) = (x_3 + 2)x_2x_1^2 \quad (15)$$

subject to

$$g_1(\boldsymbol{X}) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0 \quad (16)$$

$$g_2(\boldsymbol{X}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0 \quad (17)$$

$$g_3(\boldsymbol{X}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \quad (18)$$

$$g_4(\boldsymbol{X}) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \quad (19)$$

where $\boldsymbol{X} = [x_1\ x_2\ x_3]^T$ and $x_1 = D, x_2 = d, x_3 = N$, resepectively. The ranges of each design variables were set as: $0.05 \leq x_1 \leq 2.0, 0.25 \leq x_2 \leq 1.3, 2.0 \leq x_3 \leq 15.0$. Fig. 3 shows the system of this problem.

B. Minimization of the Total Cost of a Pressure Vessel Design Problem

The objective of this problem is to minimize the total cost $f(\boldsymbol{X})$ of the material, forming, and welding processes of a cylindrical vessel. The cylindrical vessel must be capped by hemispherical heads at both ends as shown in Fig. 3.

The thickness of the shell (T_s), the thickness of the head (T_h), the inner radius (R), and the length of the cylindrical section of the vessel (L) are the design variables. The problem can be formulated as follows:

$$\textbf{Minimize } f_2(\boldsymbol{X}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 \quad (20)$$

subject to

$$g_1(\boldsymbol{X}) = -x_1 + 0.0193x_3 \leq 0 \quad (21)$$

$$g_2(\boldsymbol{X}) = -x_2 + 0.00954x_3 \leq 0 \quad (22)$$

$$g_3(\boldsymbol{X}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0 \quad (23)$$

$$g_4(\boldsymbol{X}) = x_4 - 240 \leq 0 \quad (24)$$

where $\boldsymbol{X} = [x_1\ x_2\ x_3\ x_4]^T$ and $x_1 = T_s,\ x_2 = T_h, x_3 = R, x_4 = L$, resepectively. The ranges of each design variables were set as: $1 \leq x_1 \leq 99, 1 \leq x_2 \leq 99, 10.0 \leq x_3 \leq 200.0, 10.0 \leq x_4 \leq 200.0$

TABLE I
PROPOSED PSO ALGORITHM

1. **Initialization**
   (a) **Set** t = 0 (iteration counter)
   (b) **Specify** the parameter N (number of particles in a sub-swarm) and S (number of sub-swarm)
   (c) Randomly initialize the position of the N particles $x_{11}, x_{12}, .., x_{SN}$ with $x_{si} \in S \subset R^N$
   (d) Randomly initialize the velocities of the N particles $v_{11}, v_{12}, \dots, v_{SN}$ with $v_{si} \in S \subset R^N$
   (e) **For** $i = 1,2, \dots, N$ do $pbest_{si} = x_{si}$
   (f) **Set** $gbest = \text{argmin}_{s\in1,2,\dots,S;i\in1,2,\dots,N} f(x_{si})$
2. **Terminate Check**. If the termination criteria hold stop. The final outcome of the algorithm will be $gbest$.
3. **Calculate** the inertia weight ω^t using (11)
4. **For** $s = 1,2, \dots, S$ **Do**
4.1. **For** $i = 1,2, \dots, N$ **Do**
4.2. **Updating particle swarm**
   (a) Update the velocity V_{si} using (9)
   (b) Update the position X_{sid}^{t+1}using (10)
   (c) Evaluate the fitness of the particle $i, f(X_{si})$
   (d) If $f(X_{si})$ is feasible and $\leq f(pbest_{si})$ then $pbest_{si} = X_{si}$
   (e) If $f(pbest_{si}) \leq f(sbest_s)$ the $sbest = pbest_{si}$
   (f) If $f(X_{si})$ is infeasible, calculate r_{si}^{t+1} using (14)
   End For
   End For
5. **Set** $gbest = \text{argmin}_{s\in1,2,\dots,N} f(X_{si})$
6. **If** the iteration number can be exactly divided by the predetermined interval **then** perform extinction and offspring reproduction process
   (a) Delete particles in the worst performance sub-swarm
   (b) Copy vector of the best performance particle
   Note that the number of the new particles must be equal to the number of the died-out particles
   Else go to step 7.
7. **Set** $t = t + 1$, **Go to** step 2

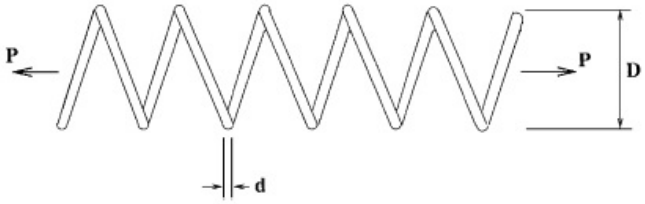

Fig. 2 Tension/Compression Spring Design Problem

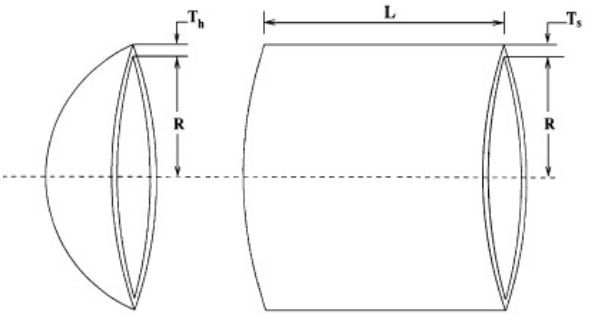

Fig. 3 Pressure Vessel Design Problem

C. Competitive algorithms and Parameter settings

Three other optimization techniques; a classic-PSO [3], an effective co-evolutionary PSO [8], and GA with constraint-handling method [13], have been selected to be compared with our proposed approach. The proposed PSO and a classic-PSO were implemented in Python language on Windows 7 Intel Xeon, 2.4 GHz processor with 64 GB of RAM.

The parameters of the proposed PSO were set as $\omega_{ini} = 0.7, \phi_1 = \phi_2 = 2.0, T = 500, s = 5$, and the swarm size was 30 (that means there were 6 sub-swarm in this study). The best result is taken from 40 independent runs.

D. Results and Discussions

The best result from 40 runs of our proposed PSO and a classic-PSO are shown in Table 2 and 3 for the spring design problem and the pressure vessel design problem, respectively. The results of an effective co-evolutionary PSO and GA with constraint-handling approach are excerpt from [8] and [13], respectively. The best result of each design problem is highlighted in boldface.

TABLE II
COMPARISON OF THE BEST RESULTS FOR THE TENSION/COMPRESSION SPRING DESIGN PROBLEM

Design Variable	Proposed PSO	Classic-PSO [3]	He [8]	Coello [13]
x_1	0.05021	0.051711	0.051706	0.051989
x_2	0.35324	0.357126	0.357126	0.363965
x_3	11.24145	11.265026	11.265083	10.890522
g_1	-0.08597	0.00039	0.0126652	-0.000013
g_2	-0.92235	-0.92679	0.00000	-0.000021
g_3	-634.32122	-640.49598	-0.92677	-4.061338
g_4	-0.73103	-0.72744	-640.43720	-0.722698
$f_1(X)$	**0.01179**	0.01266	0.01266	0.012681
RPD	-	7.38%	7.38%	7.56%

TABLE III
COMPARISON OF THE BEST RESULTS FOR THE PRESSURE VESSEL DESIGN PROBLEM

Design Variable	Proposed PSO	Classic-PSO [3]	He [8]	Coello [13]
x_1	0.81250	0.8125	0.81250	0.81250
x_2	0.43750	0.4375	0.43750	0.43750
x_3	42.09815	42.1077	42.09840	42.09739
x_4	176.60001	176.6616	176.63660	176.65404
g_1	-0.00001	0.00018	0.00000	-0.00002
g_2	-0.03588	-0.03579	-0.03588	-0.03589
g_3	-297.45589	-1299.81938	-518.51534	-27.88607
g_4	-63.39999	-63.33840	-63.36340	-63.34595
$f_2(X)$	**6058.80935**	6061.85311	6059.7143	6059.94634
RPD	-	0.05%	0.015%	0.019%

From Table 2 and Table 3, our proposed PSO outperforms other competitive methods. We made use of the relative percentage deviation (*RPD*) to measure the deviation from the best result of each design problem. The equation of *RPD* is shown as.

$$RPD = \frac{X - O^*}{O*} \times 100 \quad (25)$$

where X is the result which we need to compare and O^* is the best result. As shown in Table 2, other algorithms yield the results higher than the proposed PSO with *RDP* from 7.38% to7.56%. As shown in Table 3, other algorithms yield the results higher that the proposed PSO with *RDP* from 0.015% to 0.05%.

V. CONCLUSIONS

This paper has introduced a modified PSO with self-adaptive penalty factor in order to handle the constraints in constrained optimization problem. In our proposed PSO, the swarm behavior was utilized by that the whole swarm may divide into many sub-swarms in order to find a good source of food. The diversification of the particles is our favorable. Furthermore, we modified the penalty function method as self-adaptive penalty factor which dynamically changed belong to the performance of the last iteration. The experiment results showed that our proposed method outperformed other competitive algorithms. Our future work is to investigate the trajectory of particles in the swarm in which affected from the technique in order to explain the algorithm's performance concisely. Furthermore, the parameter settings also need to be studied. A statistical technique, such as an analysis of variance, is needed in order to define the appropriate parameter settings.

ACKNOWLEDGMENT

This research was supported partly by Department of Industrial Engineering and Management, Faculty of Engineering and Industrial Technology, Silpakorn University, Thailand.

REFERENCES

[1] RC. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th International Symposium on Micro Machine and Human Science*, 1995, pp. 39-43.
[2] Y. Shi and RC. Eberhart, "A modified particle swarm optimization," in *Proc. The Evolutionary Computation*, 1998, pp. 69-73.
[3] Y. Shi and RC. Eberhart, "Empirical study of particle swarm optimization," in *Proc. IEEE congress on Evolutionary Computation*, 1999, pp. 1945-1950.
[4] M. Clerc and J. Kennedy, "The particle swarm – explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6(1), pp. 58-72, 2002.
[5] KE. Parsopoulos and MN. Vrahatis, *Particle Swarm Optimization and Intelligence: Advances and Applications*. Hershey, PA: ISTE Ltd., 2010.
[6] M. Melanie, *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press, 1999.
[7] G. Xu, "An adaptive parameter tuning of particle swarm optimization algorithm," *Applied Mathematics and Compuation*, vol. 219, pp. 320-324, 2013
[8] Q. He and L. Wang, "An effective co-evolutionary particle swarm optimization for constrained engineering design problems," *Engineering Application of Artificial Intelligence*, vol. 20, pp. 89-99, 2007.
[9] Z. Michalewicz, "A survey of constraint handling techniques in evolutionary computation methods," in *Proc. 4th Annual Conference on Evolutionary Programming*, 1995, pp. 135-155.
[10] S. S. Rao, *Engineering Optimization: Theory and Practice*. Hoboken, NJ, 2009, pp. 428-442.
[11] A.D. Belegundu, *A Study of Mathematical Programming Methods for Structural Optimization*. Iowa City, IA: Department of Civil and Environmental Engineering, 1982.
[12] J.S. Arora, *Introduction to Optimum Design*. New York, NY: McGraw-Hill, 1989.
[13] C. A. Coello Coello and E. M. Montes, "Constraint-handling in genetic algorithms through the use of dominance-based tournament selection," Advanced Engineering Informatics, vol. 16, pp. 193-203, 2002.

About Author (s):

Choosak Pornsing is currently a lecturer of Department of Industrial Engineering and Management at Silpakorn University, Thailand. He has a Ph.D. in industrial and systems engineering from the University of Rhode Island, an M.S. in industrial and systems engineering from Lehigh University, PA, USA. He also has M.Eng. in industrial engineering from Thammasat University, Thailand and B.Eng. in industrial engineering from Naresuan University Thailand. His research interests are in the area of Computational Intelligence, Swarm Intelligence and its applications. He also conducts the research in area of Logistics and Supply Chain Management, Manufacturing Systems Design, Applied Optimization, and Production and Operation Management.

Kawinthorn Saicharoen is a lecturer of Department of Industrial Engineering and Management at Silapakorn University, Thailand. He has a B.Eng. in Industrial Engineering, M.Eng. in Industrial Management Engineering from King Mongkut's Institute of Technology North Bangkok, Thailand. His research interests are in the area of Logistics and Supply Chain Management. His current researches focus on two echelon inventory management with fuzzy demand.

Thanathorn Karot is a lecturer in logistics management and industrial engineering at Silpakorn University. Currently, he is a teacher of information technology for logistics and simulation for Arena. He has a bachelor's degree in Civil Engineering at Chulalongkorn University, Thailand, and a master's degree in Logistics at Wollongong University, Australia. He started work at Perfect Companion Group, then joined Silpakorn University in 2014. His research interests are Industrial Production Management, Warehouse Management, Transportation Management and Maintenance Management.