Academic Journal of Nawroz University (AJNU), Vol.10, No.3, 2022

https://doi.org/10.25007/ajnu.v11n3a1499

Vibrating Particles System Algorithm performance in solving Constrained Optimization Problem

Saman M. Almufti[1]

[1] Department of Computer Science, Nawroz University, Duhok, Kurdistan Region, Iraq

ABSTRACT

Metaheuristic algorithms are a collection of sophisticated techniques that mimic natural phenomena and the rational behavior of socially intelligent living organisms like insects and animals. These techniques are employed in the fields of computer science and engineering to address various optimization problems. In this paper, the vibrating particles system(VPS) which is a recently developed metaheuristic algorithm. Generally, an under-damped single degree of freedom (SDOF) free vibration oscillates and slowly comes into a resting or equilibrium position, and this is the inspiration idea of VPS. The tensions and compressions spring design problematic, which is a famous constrained based optimizations problem in engineering fields have been used to evaluate VPS algorithm. The experimental result section shows the result of solving the mentioned problem by VPS with various value for variables.

KEYWORDS: Metaheuristic Algorithm, vibrating particles system (VPS), tension/compression spring design problem, Constrained Optimization.

1. Introduction

Fundamental frequencies of structures are significant, accessible properties that allow the designer to avoid the risky resonance phenomenon. These characteristics cannot be ignored when dynamic excitations are crucial. The search spaces typically contain several local minima and demand for a capable optimization technique to be effectively addressed Because frequency responses are extremely explicit, non-convex, and non-linear with according to the cross-section zone in bar elements [1]. Since the 1980s, structural optimization has been researched [2]and has been approached using meta-heuristic algorithms and mathematical programming.

Constrained optimization is a set of numerical methods used to solve issues in which the goal is to minimize overall cost based on inputs with unfulfilled restrictions or limits.

Nature-inspired meta-heuristic algorithms have grown in popularity since the 1970s. These algorithms consist of a set of algorithmic concepts which can be used to determine heuristic techniques that are applied to a variety of different problems. The possibility of quickly solving difficult combinatorial optimization problems with high quality increases significantly with the introduction of meta-heuristics. Meta-heuristic algorithms can be consider the best techniques for solving constrained based optimization problems [3].

A well-known engineering optimization problem called the tension/compression spring design problem belongs to Single-Objective Constrained Optimization Problems, has been used to evaluate the effectiveness of recently developed metaheuristics.

In this paper, the vibrating particles system (VPS), a recently discovered optimization algorithm, is proposed to converge the a solution to the tension/compression spring design problem (TCSD) to an optimal solution.

In the experimental solution, various value have been tested as an initial value for VPS attributes. After finding the best values for variable of VPS, The minimum, maximum and mean result of the penalized objectives functions ($PFit$) calculated.

2. Literature Review

Design optimization can be defined as the process of finding the optimal parameters, which yield maximum or minimum value of an objective function, subject to certain set of specified requirements called constraints. Such problem of optimization is known as constrained optimization problems or nonlinear programming problems. Most design optimization problems in structural engineering are highly nonlinear, involving mixed (discrete and continuous) design variables under complex constraints, which cannot be solved by traditional calculus - based methods and enumerative strategies. Although these numerical optimization methods provide a useful strategy to obtain the global optimum (or near to it) for simple and ideal models but they have some disadvantages to handle engineering problems (i.e. complex derivatives, sensitivity to initial values, and the large amount of enumeration memory required). Many real-world engineering optimization problems are highly complex in nature and quite difficult to solve using these methods. In order to solve these type of problems, several heuristic, global optimization as well as meta-heuristic methods exist in the literature [4].

The computational drawbacks of existing numerical methods have forced researchers to rely on heuristic algorithms . Heuristic methods are quite suitable and powerful for obtaining the solution of optimization problems. Although these are approximate methods (i.e. their solution are good, but not provably optimal), they do not require the derivatives of the objective function and constraints. Also, they use probabilistic transition rules instead of deterministic rules. The heuristics technique includes genetic algorithms (GA), simulated annealing (SA), tabu search (TS), Harmony search (HS), Artificial Bee Colony (ABC), Cat Swarm Optimization (CSO), Ant Colony Optimization (ACO), Fish Swarm Algorithm (FSA), Lion Algorithm (LA), Elephant Search Algorithm (ESA), Grey Wolf Optimization (GWO), Particle Swarm Optimization (PSO), and other optimization algorithms [5].

Since the 1980s, many researches adapted metaheuristics algorithms for solving constrained-based problems. In 1986 Deb and Goyal presented a combined genetic search technique (GeneAS) which combined binary and real-coded GAs to handle mixed variables [6]. After that in 2002 Coello [7] applied genetic algorithms to solve these mixed-integer engineering design optimization problems. Coelho and Montes [8] proposed a dominance-based selection scheme to incorporate constraints into the fitness function of a genetic algorithm used for global optimization. In 2005 Tsai [9] proposed a novel method to solve nonlinear fractional programming problems occurring in engineering design and management. In 2007 Hsu and Liu [10] developed an optimization engine for engineering design optimization problems with monotonicity and implicit constraints. In this, monotonicity of the design variables and activities of the constraints determined by the theory of monotonicity analysis are modeled in the fuzzy proportional-derivative controller optimization engine using generic fuzzy rules. Again in 2007 Montes et al. [11] presented a modified version of the differential evolution algorithm to solve engineering design problems in which a criteria based on feasibility and a diversity mechanism are used to maintain infeasible solution. In 2008 Zhang et al. [12] proposed an algorithm for constrained optimization problem using differential evolution with dynamic stochastic selection. In 2008 Cagnina et al. [13] introduced a simple constraints particle swarm optimization (SiC-PSO) algorithm to solve constrained engineering optimization problems. Gandomi et al. [14] used a cuckoo search and Firefly algorithm for solving mixed continuous/discrete structural optimization problems.

Tension/compression string design problem. Is an engineering constrained-based problem described by Arora [15] and it consists of minimizing the weight of a tension/compression spring (as shown in Fig. 1)

subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables.

In 1982 Tension/compression string design problem has been solved by Belegundu [16] using eight different mathematical optimization techniques. In 1989 Arora [15] solved this problem using a numerical optimization technique called a constraint correction at the constant cost. Coello [7] and Coello and Montes [8] solved this problem using GA-based method. Montes and Coello used various evolution strategies to solve this problem. Additionally, in 2003 He and Wang [17] utilized a co-evolutionary particle swarm optimization (CPSO). In 2014 [18] Harish solved this problem by Artificial Bee Colony Algorithm.

3. OPTIMIZATION

Optimization occurs naturally and, obviously, plays a significant role in human existence. In the actual world, every decision is an attempt to handle a situation that is ideal or nearly optimal. The question is if there is a better answer from the one have discovered by this paper. In theory, any optimization task can be viewed as a decision-making problem. Commonly, optimizations refers to finding the best method for enhancing the functionality of the system in question.

According to Beightler [1], optimizations represents a three-step process that entails modelling the problem based on the problem's knowledge, selecting effectiveness measures or the target function, and putting the optimization method or theory into practice. The development of computers, which started in the middle of the 1940s, has undoubtedly benefited the field of optimization in general and the last step in particular.

In recent decades, significant advancements in computer performance have led to the capacity for modeling, analyzing, and designing real-world problems as precisely as seems in a variety of disciplines. Almost all real-world issues can be categorized as complex optimization issues.

The increased interests in optimizing the real-world optimizations issues by inspiring natural phenomenon consume major advantages for various fields, including engineering, and computer science [5].

Since 2000, several practical optimization issues have been stated, modeled, and optimized, yielding optimal solutions. During this time, a large amount of advanced literature was developed in fields such parallel metaheuristics, constrained optimization, of large-scale metaheuristics optimization, synergy metaheuristics, and cloud computing metaheuristics.

Generally, developing new methods for achieving more efficient tradeoff among explorations and exploitations, had an increasing attention during this period, and as a result, many new nature-based metaheuristics were developed such as Artificial Bee Colony (ABC) [19], Cat Swarm Optimization (CSO) [20], teaching learning-based optimization (TLBO) [21], Ant Colony Optimization (ACO) [22], Charged System Search (CSS) [23], Fish Swarm Algorithm (FSA) [24], Big Bang Big Crunch (BB-BC) [25], Lion Algorithm (LA) [26], Krill Herd (KH) [27], Elephant Search Algorithm (ESA) [28], Grey Wolf Optimization (GWO) [29], Colliding Bodies Optimization (CBO) [30], Cuckoo Search (CS) [31], Dolphin Echolocation (DE) [32], Particle Swarm Optimization (PSO) [33], Vibrating Particles System (VPS) [34], and other optimization algorithms [35].

4. CONSTRAINED OPTIMIZATION PROBLEM

Constrained optimization is a set of numerical methods used to solve issues in which the goal is to minimize overall cost based on inputs with unfulfilled restrictions or limits. The constrained optimization strategy used is determined by the type of problem and function to be solved. In a broader sense, such methods are connected to constraint fulfillment problems, in which the user must satisfy a set of given constraints. In contrast, constrained optimization

problems require the user to reduce the entire cost of the unfulfilled restrictions.

Metaheuristic Algorithms are the best technique to discover a solution that produces the best possible performance of the problem under the given conditions.

The performance of the problem is defined by an objective function or a set of objective functions. In this sense, constrained optimization issues are divided into two types:

- Single-Objectives Constrained Optimizations Problems (SOCOP).
- Multi-Objectives Constrained Optimizations Problems (MOCOP).

These functions can either be maximally or minimally optimum.

In this paper, a Vibrating Particles System (VPS) metaheuristic algorithm is used to solve a known Single-Objective Constrained Optimization Problems called tension/compression spring design problem.

4.1 Constraints handling

When applying metaheuristics based approaches to the optimization of engineering design problems, a key issue is how the algorithm handles constraints relating to the problem. The literature proposes several methods for constraint handling in evolutionary algorithms and swarm intelligence approaches [8].These methods can be grouped into categories, such as methods that preserve solution feasibility, penalty-based methods, methods that clearly distinguish between feasible and unfeasible solutions, and hybrid methods [36]. It is usual to handle constraints in optimization methods based on the concept of penalty functions (which penalize unfeasible solutions) [36].That is, one attempts to solve an unconstrained minimization problem in the search space S using a modified fitness function, such as Eq(1):

$$eval(x) = \begin{cases} f(x), & \text{if } x \in F \\ f(x) + penalty(x), & \text{otherwise} \end{cases} \quad (1)$$

where penalty(x) is zero if no constraint is violated, and is positive otherwise. Usually, the penalty function is based on a distance measure to the nearest solution in the feasible region F or on the effort to repair the solution. The methodology proposed for constraint handling is divided into two steps. The first step aims at finding solutions for the decision variables that lie within user-defined upper (ub) and lower (lb) bounds, that is, x ∈[Lb, Ub]. Whenever a lower bound or an upper bound constraints is not satisfied, a repair rule is applied, according to Eqs. (2) and (3), respectively:

$$x_i = x_i + rand[0,1] \cdot \{ub(x_i) - lb(x_i)\} \quad (2)$$

$$x_i = x_i - rand[0,1] \cdot \{ub(x_i) - lb(x_i)\} \quad (3)$$

where rand[0,1] is a uniformly distributed random value between 0 and 1. In the second step decision variables are considered inequalities $g(x) \leq 0$. In this work a metaheuristic algorithm called Vibrating Particles System (VPS) is used for minimizing a constrained based engineering problem called tension/compression spring design problem.

4.2 tension/compression spring design problem

Vibration is a mechanical phenomenon that causes oscillations around an equilibrium state. It is one of well-known Single-Objective Constrained Optimization Problems (SOCOP) in the engineering fields. Vibrations are classified into two types: (1) free vibration and (2) forced vibration [3].

An example of a continuous constrained problem is the tension/compression spring design problem (TCSD) in Fig. 1. Assuming a constant tension/compression load, the challenge is to reduce a coil spring's volume V. There are three design variables that make up the problem. They are:

- $P = x1 \in [2, 15]$
- $D = x2 \in [0.25, 1.3]$
- $d = x3 \in [0.05, 2]$.

Where, P represents the number of spring's active coils, D represents the diameter of the winding, and d represents the diameter of the wire. which are employed to reduce weight while meeting restrictions on sheared stress, flow frequencies, and minimum deflections [3].

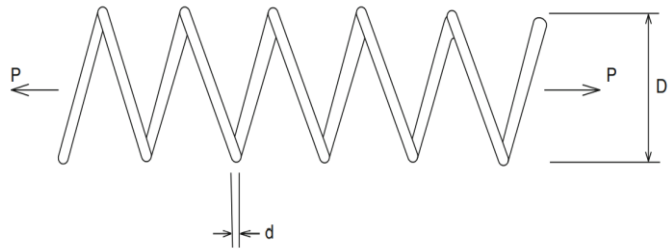

FIGURE 1: SCHEMATIC OF THE TENSION/COMPRESSION SPRING

The TCSD problem is expressed mathematically by the cost functions Eq(4), that is required to be minimized with constraints g1,g2,g3, and g4 in Eq(5):

$$f(x) = (X_3 + 2)X_2X_1^2 \tag{4}$$

The design space is constrained by the values$X_1, X_2, and\ X_3$. The range of designing-variables between Lower-Boundary $(Lb) = [0.05, 0.25, 2]$ and Upper-Boundary $(Ub) = [2, 1.3, 15]$. Using the descriptions for the modelling of a COP and penalty provided in Eq (1). Four constraints are about the shear stress, surge frequency, and the minimum deflection. One objective function, three designing variables, and four constraints are shown in Eq(5) [37].

$$\begin{aligned} g_1(x) &= 1 - \frac{X_2^3 X_3}{72785X_1^2} \leq 0 \\ g_2(x) &= \frac{4X_2^2 - X_1X_2}{12566(X_2X_1^3) - X_1^4} + \frac{1}{5108X_1^2} - 1 \leq 0 \\ g_3(x) &= 1 - \frac{140.45X_1}{X_2^2X_3} \leq 0 \\ g_4(x) &= \frac{X_1 + X_2}{1.5} - 1 \leq 0 \end{aligned} \tag{5}$$

The tension/compression spring design problem has been chosen as a well-known engineering optimization problem to test the effectiveness of recently developed metaheuristics. In this paper the mentioned problem has been used for evaluating the a metaheuristic based algorithm called Vibrating Particles System (VPS).

5. VIBRATING PARTICLES SYSTEM (VPS)

In 2017 Kaveh and Ilchi Ghazaan created the Vibrating Particles System (VPS) method [38], an evolutionary metaheuristic search technique that stimulate the free vibration of single degree of freedom systems with viscous dampening. VPS has been utilized to solve various structural optimization issues, and the results prove its viability in terms of convergence and accuracy [34].

VPS, Starts with a random collection of initial solutions and considers them as free vibrated single degree of freedom systems with vibration, similar to previous population-based metaheuristics. Any free vibrating system oscillates and come back to its equilibriums point with a specific formulation when subjected to dampening conditions. This is easily demonstrated using differential equations. As the optimization process moves forward, VPS improves the quality of the particles periodically by oscillating them forward towards the equilibrium position by using a combination of randomness and exploitation of the values obtained [19].

Consider that each particle's equilibrium position is made up of three positions: the best/highest position(HP), a good particle (GP), and a bad particle (BP). Thus, the essence of VPS is based on three fundamental concepts:

- self-adaptations: particle travels toward HP .
- cooperation's: GP and BP, are chosen among the particle themselves, can affect the new position of particles.
- Competitions: GP influency becomes higher than that of BP.

Generally VPS corrects the position of particles exiting the search space using a memory based on the harmony search technique.

6. Vibrating Particles System Frameworks

This section presents the ideas behind the VPS algorithm, its pseudocode, and the associated flowchart.

6.1 VPS Formulation

The formula employed to characterize the free vibration of an under-damped single degree of freedom (SDOF) system as follows in Eq(6).

$$X(t) = \rho e^{-\varepsilon \omega_n t} \sin(\omega_D t + \theta) \quad (6)$$

where ρ and θ are constants that are typically derived from the vibration's beginning circumstances, ω_n is the vibration's natural circular frequency. ω_D and ε are respectively, the damped natural frequency and the damping ratio that are shown in Fig 2 , and can be calculated by Eq(7) and Eq(8).

$$\omega_D = \omega_n \sqrt{1 - \varepsilon^2} \quad (7)$$

$$\varepsilon = \frac{c}{2m\omega_n} \quad (8)$$

All particles' initial positions in the search space are generated at random by Eq(9):

$$X_j^i = X_{min} + rand * (X_{max} - X_{min}) \quad (9)$$

Where X_{min} and X_{max} are the minimum and the maximum allowable variables vectors. And rand is a random number uniformly distributed in the range of [0, 1].

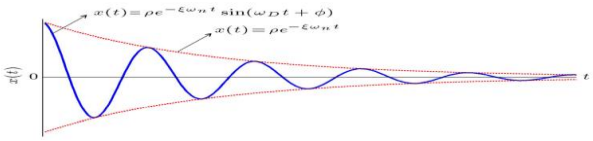

FIGURE 2. VIBRATING MOTION OF UNDER-DAMPED SYSTEM.

As is evident, an under-damping SDOFs free vibrate and slowly comes into a its equilibrium point. And this vibrating characteristics is the idea which the VPS is inspires.

similar to other metaheuristics algorithms, VPS begins with a set of intrant solutions that are randomly produced inside the search space. nVP stands for the number of vibrated-particle. The appropriate objective-functions (Fit) and its penalized-functions ($PFit$) are generated once the items are evaluated [34]. VPS updates HP, GP, and BP with various weights (ω_1, ω_2,and ω_3) for each particle. After the population is sorted in ascending order by the values of the penalized objective function.For each particle, GP and BP are randomly selected from the first and second halves of the population, with the exception of itself.

The damping level has a significant impact on vibration. The amplitude of a free damped vibration decreases as the damping level increases. To mimic this behavior in the VPS, the following descending function (D) relative to number of repetitions is calculated by Eq(10):

$$D = \left(\frac{NITs}{maxNITs}\right)^{\alpha} \quad (10)$$

$$maxNITs = \frac{\text{maxNFEs}}{nVP} \quad (11)$$

where NITs is the algorithm's current iteration number, maxNITs is the maximum number of algorithm iterations chosen as the halting criterion, maxNFEs is the maximum number of function evaluation, and is α is a constant It is recommended to choose a value of 0.05 [1].

According to the above notions, the particles are updated by the formula Eq(12,13,14) , which will be referred to as the free vibration formula hereafter:

$$\begin{aligned} newVP_i = {} & \omega_1(D * A * rand_1 + HP) \\ & + \omega_2(D * A * rand_2 + GP_i) \\ & + \omega_3(D * A * rand_3 + BP_i) \end{aligned} \quad (12)$$

$$\begin{aligned} A = {} & \omega_1(HP - VP_i) + \omega_2(GP_i - VP_i) \\ & + \omega_3(BP_i - VP_i) \end{aligned} \quad (13)$$

$$\omega_1 + \omega_2 + \omega_3 = 1 \quad (14)$$

where VPi and newVPi are the ith particle's current and updated locations, respectively; ω_1, ω_2, and ω_3 are different constant weight used for comparing the qualified relevance of the ith particle's good particle, bad particle, and algorithm's best-so-far particle; and

rand represents a random value uniformly generated in $[0-1]$ range.

In Eq(12 and 13), A and D effects on algorithm process similar to the effect of ρ and $e^{-\varepsilon\omega_n t}$ in Eq(6). The value of $\sin(\omega_D t+\theta)$ is measured as unity. A parameter p between $(0,1)$ is initialized, and it specifies the influence of BP will/willn't considers when position are updating. p and random are compared as shown in Eq(15):

$$\text{If P<rand} \rightarrow \omega_3 = 0, \omega_2 = 1-\omega_1 \qquad (15)$$

VPS takes into account three key concepts: self-adaptation, cooperation, and competition. As a particle travels toward HP, self-adaptation occurs. In VPS Any particle has the potential to impact the new position of the other, cooperation between particles is provided. Because of the p parameter, the influence of the GP (good particle) is greater than that of the BP (bad particle); hence, competition is created [1].

To handle a particle that has violated the variables' bounds, VPS employs a harmony search-based handling strategy. In tis paper, the nVPs number of the highest vibrating particle, as well as their corresponding objective function ($FitM$) and its penalized ($PFit-M$) values, are saved in a vibrating-particle memory ($VP-M$). To achieve this goal, vibrating particle memory used to store numbers as equal to nVP.

Considering memory and utilizing it in various techniques can improve metaheuristics efficiency without raising computing cost. It can be emphasized that VPS only used it to regenerate particles that have exited the search space. This method allows any element of the solution set that violates the boundaries to be renewed from the $VP-M$ and can be determined by Eq(16).

$$VP(i,j)=\begin{cases} w.p.\ vpmcr \Rightarrow & \text{select a new value for a variable from VP} - \text{M,} \\ & \text{w. p. } (1-\text{par}) \text{do nothing,} \\ & \text{w. p. par choose a neighboring value.} \\ w.p.\ (1-vpmcr) \Rightarrow & \text{Select a new value randomly} \end{cases} \qquad (16)$$

where "w.p." stands for "with the probability," and $VP(i,j)$ is the jth element of the ith vibrated particle, $vpmcr$ is the vibrating-particles memory with a rate within [0,1] and determines the possibility of selecting a value from the historic values saved in $VP-M$, and $(1-vpmcr)$ determines the probability randomly selecting a values in the possible range. After selecting a value from $VP-M$, the pitch-adjusting process begins. The value $(1-par)$ specifies the neglecting rate, and par specifies the rate of selecting a value from the particles bordering the best vibrating particle or those preserved in memory. Random generating step $size$ $(\pm bw * rand)$ can be used for continuous search space to choose a value from particles stored in memory or those closest to the best vibrating particle [20].

6.2 VPS pseudocode

Figure 4, illustrate the VPS pseudocode.

```
procedure Vibrating Particles System (VPS)
Initialize algorithm parameters
Initial positions are created randomly
The values of objective function are evaluated and HB is stored
While maximum iterations is not fulfilled
        for each particle
                The GP and BP are chosen
                if P<rand
                w3=0 and w2=1-w1
                end if
                for each component
                        Update locations
                end for
                Violated components are regenerated by
                        harmony search-based handling approach
        end for
        The values of objective function are evaluated and HB is updated
        end while
end procedure
```

Figure 3: VPS Pseudocode

6.3 VPS flowchart

Figure4, shows the flowchart how the VPS work.

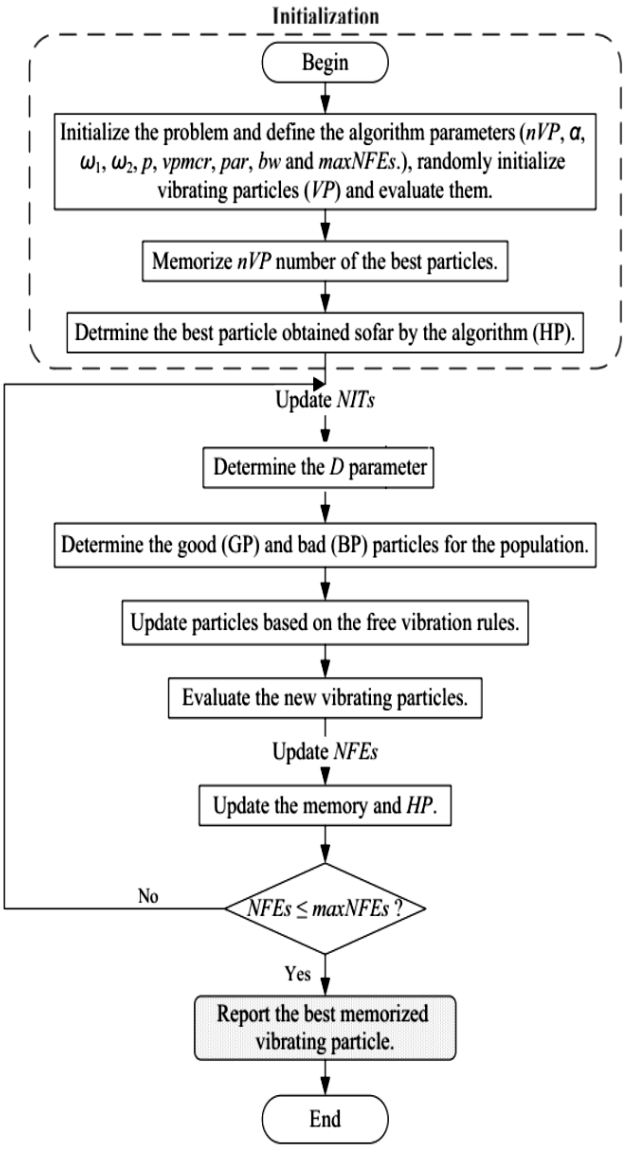

FIGURE 4: VPS FLOWCHART

7. EXPERIMENTAL RESULTS

In this section the results of applying Vibrating Particles System (VPS) algorithm for solving tension/compression spring design problem (TCSD) are illustrated.

Generally, VPS depend on various constant variable that directly affect the convergence of the results to the optimal value. This paper testes various values for the VPS variables (ε, α, nVP, p) to achieve VPS best efficiency.

Fig. 5., shows of how damping ratio (ε) affects vibratory motion of Free vibrating systems based on Eq (3, 4, and 5).

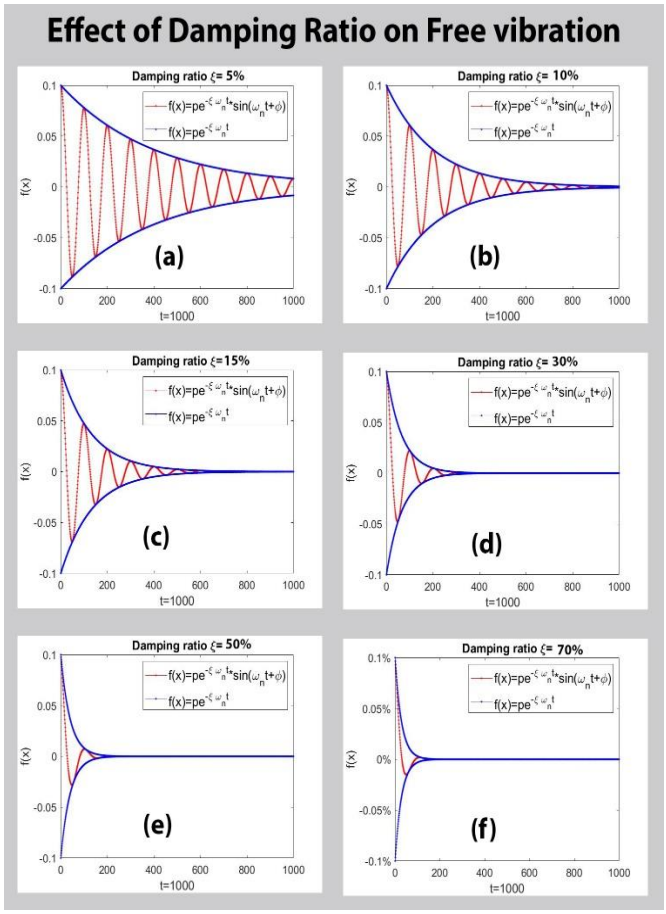

FIGURE 3: FREE VIBRATING SYSTEMS WITH SIX LEVELS OF E (A)5%, (B)10%, (C)15%, (D)30%, (E)50%,AND (F)70%

In Fig 5, six different value for Damping ratio ε={5%, 10%, 15%, 30%, 50%, and 70%} are respectively tested on a free vibrating system. The results shows that as ε increase the damping of system decreases.

In VPS, the attribute α that effects on the value of descending function (D) in Eq(7). Figure 6, shows the minimum, maximum, and mean result of *PFit* based on six different values of α ={0.01, 0.05, 0.1, 0.15, 0.2, and 0.25} respectively. It can be seen that as the value of α increase the PFit minimum, maximum, and the mean results increases, that mean the results are in divergence, and vice versa the decrease in α value outcomes a convergence in PFit results.

Table 1, shows the minimum values of (minPFit, maxPFit, and meanPFit) that are resulted by VPS effected by different value of α.

TABLE 1:EFFECT OF (A)ON MINIMUM OF VPS PFIT RESULTS

α	MIN(minPFit)	MIN(maxPFit)	MIN(meanPFit)
0.01	0.012739	**169012.4**	32668.91
0.05	0.012688	994955.5	248757.6

0.1	0.012675	4161466	**315621**
0.15	**0.012671**	6794985	1616036
0.2	0.01269	5111698	680912.7
0.25	0.012682	7003061	1209539
Opt. result	α =0.15	α =0.01	α =0.1

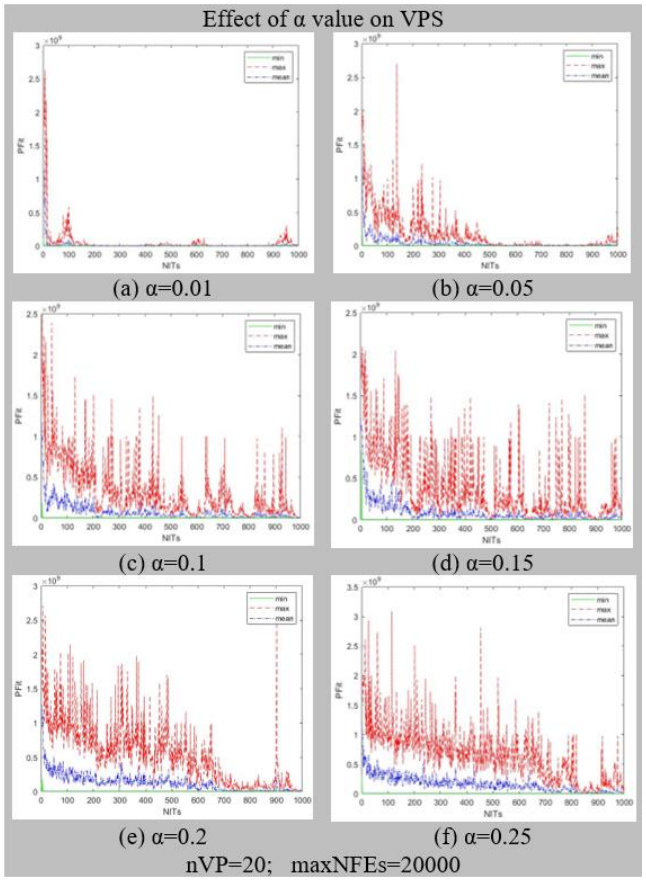

FIGURE 4:EFFECT OF A ON CONVERGENT HISTORIES OF THE MIN, MAX, AND MEAN FOR VPS

In Fig 8, six different number of Vibrating Particles nVP={10, 20, 30, 40, 50, and 60} are respectively tested on a free vibrating system. The results shows that number of VPS algorithm iterations decrease by increasing the nVP number depending on Eq(8).

Fig. 7. shows the minimum, maximum, and mean result of *PFit* based on six different values of of nPVs ={10, 20, 30, 40, 50, and 60} respectively. It can be seen that when the PFit result are slightly different when nVP equal to 10 or 20 and obtains its best convergency results, but in case of nVP=10 VPS have a high number of algorithm iteration and high number of Function Evaluation, thus it consume much time to get the result. That's why, this paper uses nVP=20, that resulting PFit almost equal to the results of VPS when nVP=10 in less time.

Table 2, shows the minimum values of (minPFit, maxPFit, and meanPFit) that are resulted by VPS effected by different value of nVP.

TABLE 2: EFFECT OF (NVP)ON MINIMUM OF VPS PFIT RESULTS

nVPs	MIN(minPFit)	MIN(maxPFit)	MIN(meanPFit)
10	**0.012669**	**0.012669**	**0.012669**
20	0.012688	994955.5	248757.6
30	0.012674	29090088	3991628
40	0.012692	19171707	1886142
50	0.012711	32968057	4582201
60	0.012694	1.03E+08	12480661
Opt. result	nVP=10	nVP=10	nVP=10

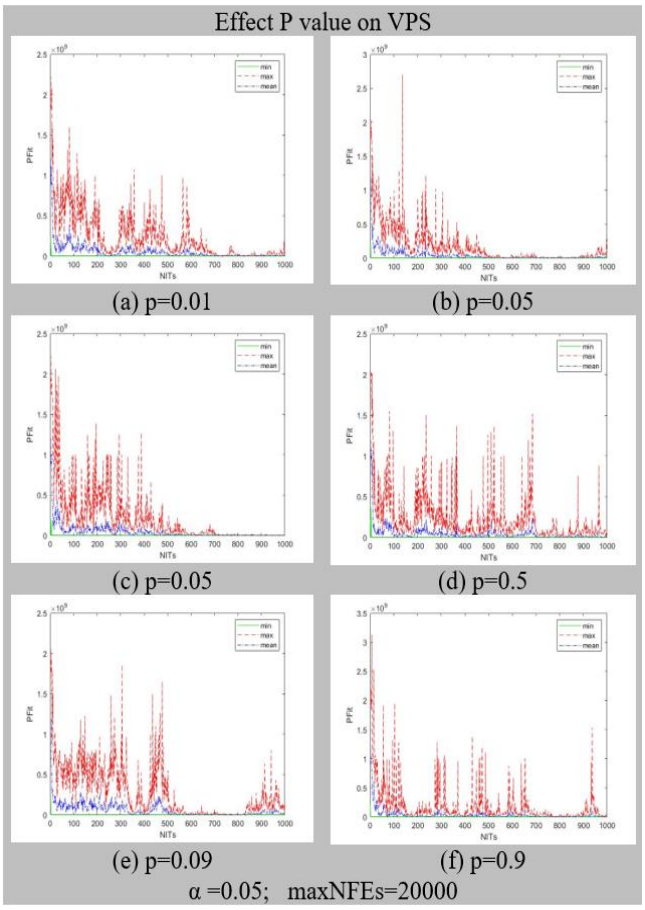

FIGURE 5: EFFECT OF NVPS ON CONVERGENCE HISTORIES OF THE MIN, MAX, AND MEAN VALUE OF VPS

In Fig. 8. six different values have been tested for the Probability Variable P={0.01, 0.05, 0.09, 0.1, 0.5, and 0.9} that belongs to range [0-1] are respectively tested on a free vibrating system. The results shows that VPS algorithm results PFit are in best convergency when

P=0.05.

Table 3, shows the minimum values of (minPFit, maxPFit, and meanPFit) that are resulted by VPS effected by different value of p.

TABLE 3:EFFECT OF (P)ON MINIMUM OF VPS PFIT RESULTS

p	MIN(minPFit)	MIN(maxPFit)	MIN(meanPFit)
0.01	0.012684	1537569	294571.7
0.05	**0.01267**	309300.2	63109.82
0.09	0.012687	**1331415**	231487.7
0.1	0.012688	994955.5	248757.6
0.5	0.012681	6589368	1106855
0.9	0.012671	1800824	309956.8
Opt. result	p =0.05	α =0.09	α =0.05

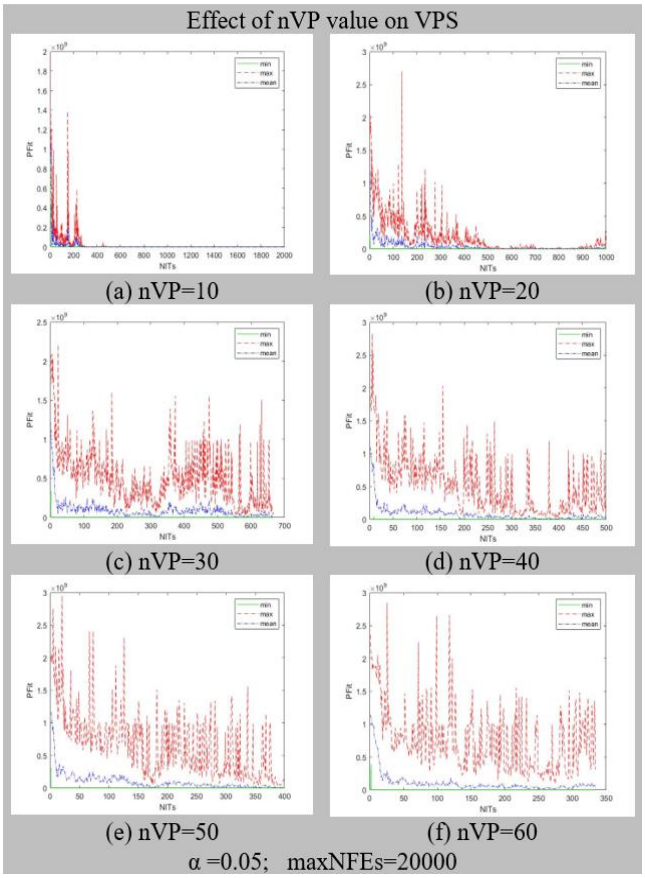

FIGURE 6: EFFECT OF P ON CONVERGENCE HISTORIES OF THE MIN, MAX, AND MEAN VALUE OF VPS

Finally, depending on the previous results, the paper initialize variable to:

- maxNEFs=20000
- p=0.05
- nVP=20
- α=0.15

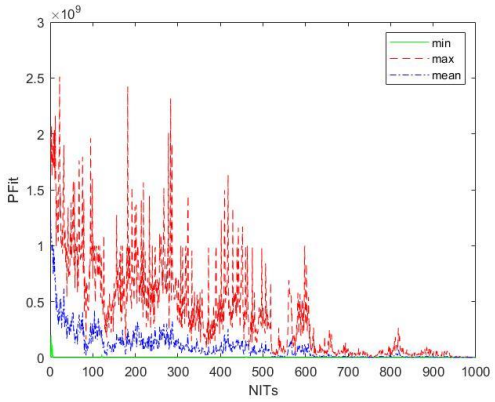

FIGURE 7: VPS RESULTS WITH P=0.05, NVP=20, A=0.15 IN 1000 ITERATION

TABLE 4: VPS MIN(MINPFIT) MIN(MAXPFIT) MIN(MEANPFIT)

VPS	MIN(minPFit)	MIN(maxPFit)	MIN(meanPFit)
maxNEFs=20000 p=0.05 nVP=20 α=0.15	0.012673	861911.2	163049.8

Table 5, compares the experimental results with the results of previous studies. It clearly shows that VPS result in better solution than 5 of the previous studies

TABLE 5: EXPERIMENTAL RESULT COMPARISON

#	Researcher(s)	Algorithm	X1	X2	X3	F(x)	Ref
1.	Coelho	QPSO	0.051515	0.352529	11.538862	0.012665	[39]
2.	Garg	ABC	0.05 I 689 I 6	0.35672003	11.288831	0.012665233	[18]
3.	Omran et al.	CODEQ	0.05168375	0.35658984	11.296471	0.012665238	[40]
4.	Gandomi et al.	BAT	0.05169	0.35673	11.2885	0.01267	[41]
5.	**This paper results**	**VPS**	**0.0522822219758084**	**0.371100339365763**	**10.5165209838107**	**0.012673**	—
6.	He and Wang	CPSO	0.051728	0.357644	11.244543	0.0126747	[42]
7.	Modified FA	MFA	0.05 I 73	0.3577	11.2595	0.01269	[43]
8.	Montes et al.	ES	0.051643	0.35536	11.397926	0.012698	[44]
9.	Coello	GA	0.05 148	0.351661	11.632201	0.01270478	[45]
10.	Raj et al.	ADE	0.053862	0.41 128365	8.6843798	0.0127484	[46]

8. CONCLUSION

Metaheuristic Algorithms are a set of algorithms that inspired from a natural phenomenon, it widely used to solve single objective functions. This paper uses Vibrating Particles System (VPS) metaheuristic algorithm to solve an engineering single objective Constrained Optimization Problems called tension/compression spring design problem.

VPS depends on various variable to converge the result to its optimal solution. In this paper different values are tested to conclude the best value for each attribute. Then finally all variable are reset to the obtained best value and the final result found that result in best (min, max, mean) PFit convergent value.

9. REFERENCES

1. Kaveh and M. I. Ghazaan, "Vibrating particles system algorithm for truss optimization with multiple natural frequency constraints," *Acta Mech,* 2016.
2. L. Bellagamba and T. Y. Yang, "Minimum-mass truss structures with constraints on fundamental natural frequency," *AIAA Journal,* vol. 19, no. 11, 1980.
3. Y. Celik and H. Kutucu, Solving the Tension/Compression Spring Design Problem by an Improved Firefly Algorithm, 2018.
4. H. Garg, "SOLVING STRUCTURAL ENGINEERING DESIGN OPTIMIZATION PROBLEMS USING AN ARTIFICIAL BEE COLONY ALGORITHM," *JOURNAL OF INDUSTRIAL AND MANAGEMENT OPTIMIZATION,* vol. 10, p. 777–794, 2014.
5. S. M. Almufti, "Lion algorithm: Overview, modifications and applications," *International Research Journal of Science, Technology, Education, and Management,* vol. 2, no. 2, pp. 176-186, 2022.
6. K. D. a. M. Goyal, " A combined genetic adaptive search (GeneAS) for engineering design," *Computer Science and Informatics,* vol. 26, p. 30–45, 1986.
7. C. A. C. C. a. E. M. Montes, "Constraint- handling in genetic algorithms through the use of dominance-based tournament selection," *Advanced Engineering Informatics,* vol. 16, p. 193–203, 2002.
8. C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," *Comput. Methods Appl. Mech. Engrg,* p. 1245–1287, 2002.
9. J. Tsai, "Global optimization of nonlinear fractional programming problems in engineering design," *Engineering Optimization,* vol. 37, p. 399–409, 2005.
10. Y. L. H. a. T. C. Liu, "Developing a fuzzy proportional derivative controller optimization engine for engineering design optimization problems," *Engineering Optimization,* vol. 39, pp. 679-700, 2007.
11. C. A. C. C. J. V. R. a. L. M. D. E. M. Montes, "Multiple trial vectors in differential evolution for engineering design," *Engineering Optimization,* vol. 39, p. 567–589, 2007.
12. W. L. a. X. W. M. Zhang, "Differential evolution with dynamic stochastic selection for constrained optimization," *Information Sciences,* vol. 178, p. 3043–3074, 2008.
13. S. C. E. a. C. A. C. C. L. C. Cagnina, "Solving engineering optimization problems with the simple constrained particle swarm optimizer," *Informatica,* vol. 32, p. 319–326, 2008.
14. X. S. Y. a. A. H. A. A. H. Gandomi, "Mixed variable structural optimization using firefly algorithm," *Computers & Structures,* vol. 89, p. 2325–2336, 2011a.
15. J. S. Arora, "Introduction to Optimum Design," *McGraw-Hill,* 1989.
16. D. Belegundu, "A Study of Mathematical Programming Methods for Structural Optimization," *PhD thesis, Department of Civil and Environmental Engineering, University of Iowa,USA,* 1982.
17. Q. H. a. L. Wang, "An effective co - evolutionary particle swarm optimization for constrained engineering design problems," *Engineering Applications of Artificial Intelligence,* vol. 20, p. 89–99., 2007.
18. H. Garg, "Solving Structural Engineering Design Optimization Problems Using an Artificial Bee Colony Algorithm," *Journal of Industrial And Management Optimization* , vol. 10, pp. 777-794 , 2014.
19. I. B. o. H. B. S. F. N. Optimization, Karaboğa, Derviş, 2005.
20. R. R. Ihsan, S. M. Almufti, B. M. Ormani, R. R. Asaad and R. B. Marqas, "A Survey on Cat Swarm Optimization Algorithm," *Asian Journal of Research in Computer Science,* vol. 10, no. 2, pp. 22-32, 2021.
21. R. V. Rao, Teaching Learning Based Optimization Algorithm, Springer, 2016.
22. S. M. Almufti, U-Turning Ant Colony Algorithm powered by Great Deluge Algorithm for the solution of TSP Problem, 2015.
23. Z. Tabrizian, G. G. Amiri and M. H. A. Beigy, "Charged System Search Algorithm Utilized for Structural Damage Detection," *Shock and Vibration,* 2014.
24. F. S. Lobato and V. S. Jr., "Fish swarm optimization algorithm applied to engineering system design," *Latin American Journal of Solids and Structures,* vol. 11, no. 1, 2014.
25. H. M. GENC, I. EKS˙IN and O. K. EROL, "Big bang-big crunch optimization algorithm with local directional moves," *Turkish Journal of Electrical Engineering & Computer Sciences,* vol. 21, p. 1359 – 1375, 2013.
26. S. M. Almufti, "lion optimization algorithm: Overview, modifications and applications," *International Research Journal of Science, Technology, Education, and Management,* 2022.
27. L. Abualigah, "Feature Selection and Enhanced Krill Herd Algorithm for Text Document Clustering," *Studies in Computational Intelligence,* 2019.
28. S. Deb, S. Fong and Z. Tian, "Elephant Search Algorithm for optimization problems," in *Tenth International Conference on Digital Information Management (ICDIM)*, 2015.
29. R. B. Marqas, S. M. Almufti, H. B. Ahmed and R. R. Asaad, "Grey wolf optimizer: Overview, modifications and applications," *International Research Journal of Science, Technology, Education, and Management,* vol. 1, no. 1, pp. 44-56, 2021.
30. V. M. A. Kaveh, Colliding Bodies Optimization, springer, 2015.

31. S. A. Uymaz and G. Tezel, "Cuckoo Search (CS) Optimization Algorithm for Solving Constrained Optimization Problems," in *International Conference on Computer Science, Engineering and Technology*, 2014.
32. Kaveh and N. Farhoudi, "A new optimization method: Dolphin echolocation," *Advances in Engineering Software,* vol. 59, p. 53–70, 2013.
33. Y. Zebari, H. K. Omer and S. M. Almufti, "A comparative study of particle swarm optimization and genetic algorithm," *Journal of Advanced Computer Science & Technology,* vol. 8, no. 2, pp. 40-45, 2019.
34. Kaveh and T. Bakhshpoor, Metaheuristics: Outlines, MATLAB Codes and Examples, Springer, 2019.
35. S. M. Almufti, "Historical survey on metaheuristics algorithms," *International Journal of Scientific World,* vol. 7, no. 1, pp. 1-12, 2019.
36. Z. &. S. M. Michalewicz, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation,* vol. 4, no. 1, pp. 1-32, 1996.
37. K. Parsopoulos and M. N. Vrahatis, "Unified Particle Swarm Optimization for Solving Constrained Engineering Optimization Problems," in *Lecture Notes in Computer Science*, 2005.
38. Kaveh and M. I. Ghazaan, "A new meta-heuristic algorithm: Vibrating particles system," *Scientia Iranica,* vol. 24, no. 2, pp. 551-566, 2017.
39. L. S. Coelho, "Gaussian Quantum-Behaved Particle Swarm Optimization Approaches for Constrained Engineering Design Problems," *Expert Systems with Applications,* vol. 37, pp. 1676-1683, 2010.
40. M. G. H. S. A. Omran, "Constrained optimization using CODEQ, Chaos," *Solitons & Fractals,* vol. 42, pp. 662-668, 2009.
41. H. Y. X. S. A. A. H. T. S. Gandomi, "Bat Algorithm for Constrained Optimization Tasks," *Neural Computing and Applications,* vol. 22, no. 6, pp. 1239-1255, 2013.
42. Q. W. L. He, "An Effective Co-Evolutionary Particle Swarm Optimization for Constrained Engineering Design Problems," *Engineering Applications of Artificial Intelligence,* vol. 20, pp. 89-99, 2007.
43. M. J. Kazemzadeh-Parsi, "A Modified Firefly Algorithm for Engineering Design Optimization Problems," *IJST, Transactions of Mechanical Engineering,* vol. 38, pp. 403-421, 2014.
44. E. M. C. C. A. C. Montes, "An Empirical Study About The Usefulness of Evolution Strategies to Solve Constrained Optimization Problems," *International Journal of General Systems,* vol. 37, pp. 443-473, 2008.
45. C. C. Carlos, "Use of a self-adaptive penalty approach for engineering optimization problems," *Computers in Industry,* vol. 41, p. 113–127, 2000.
46. K. H. S. R. S. M. G. S. D. A. P. C. Raj, "An Evolutionary Computational Technique for Constrained Optimisation in Engineering Design," *Journal of the Institution of Engineers India Part Me Mechanical Engineering Division,* vol. 86, pp. ,121-128, 2005.