INTERNATIONAL JOURNAL OF OPTIMIZATION IN CIVIL ENGINEERING
Int. J. Optim. Civil Eng., 2018; 8(3):401-414

MATLAB CODE FOR AN ENHANCED VIBRATING PARTICLES SYSTEM ALGORITHM

A. Kaveh[1*, †], S. R. Hoseini Vaez[2] and P. Hosseini[2]
[1]*Centre of Excellence for Fundamental Studies in Structural Engineering, School of Civil Engineering, Iran University of Science and Technology, Tehran-16, Iran*
[2]*Department of Civil Engineering, Faculty of Engineering, University of Qom, Qom, Iran*

ABSTRACT

Vibrating particles system (VPS) is a new meta-heuristic algorithm based on the free vibration of freedom system' single degree with viscous damping. In this algorithm, each agent gradually approach to its equilibrium position; new agents are generated according to current agents and a historically best position. Enhanced vibrating particles system (EVPS) employs a new alternative procedure to enhance the performance of the VPS algorithm. Two different truss structures are investigated to demonstrate the performance of the VPS and EVPS weight optimization of structures.

Keywords: vibrating particles system algorithm; enhanced vibrating particles system algorithm; MATLAB code; truss structures.

Received: 8 September 2017; Accepted: 12 November 2017

1. INTRODUCTION

One of the most important applications of the optimization in knowledge engineering is optimal design. This design leads to correct use of what is limited in each engineering problems. Metaheuristic algorithms are usual tools to optimize problems in suitable time but these methods cannot ensure to gain the best answer. Structural optimization results in economical design and is an active topic in the field of civil and in particular structural engineering [1-5]. Meta-heuristic algorithms that are widely used in structural optimization are as follows:

Genetic Algorithms (GA) [6], Particle Swarm Optimization (PSO) [7], Charged System Search algorithm (CSS) [8], Ray Optimization (RO) [9], Colliding Bodies Optimization (CBO)[10], Cyclical parthenogenesis algorithm [11], Ant Colony Optimization (ACO) [12],

[*]Corresponding author: Centre of Excellence for Fundamental Studies in Structural Engineering, School of Civil Engineering, Iran University of Science and Technology, Tehran-16, Iran
alikaveh@iust.ac.ir (A. Kaveh)

Dolphin Echolocation algorithm (DEA)[13], Imperialist Competitive Algorithm (ICA) [14], Harmony Search (HS) algorithm [15], Grey wolf optimizer [16].

In this paper the vibrating particles system (VPS) algorithm [17] and the enhanced vibrating particles system (EVPS) [2] are briefly introduced and employed to weight optimization of some truss structures. The VPS algorithm is inspired by the free vibration of single degree of freedom systems with viscous damping. The EVPS algorithm employs some procedures to enhance the VPS algorithm performance. These procedures are employed to augment the ability of the standard VPS to perform a global search and prevent entrapment in local optima. The VPS and EVPS algorithms are used for weight optimization of two truss structures. Results imply that the performance of EVPS algorithm is enhanced in comparison to the standard VPS algorithm for these structures.

This paper is organized as follow: In the first section, introduction is presented. Vibrating particles system (VPS) and enhanced vibrating particles system (EVPS) algorithms are imposed in section 2. The formulation of the weight optimization of truss structures is performed in section 3. In the fourth section, two benchmark problems are investigated with VPS and EVPS algorithms. Conclusion is presented in last section. Computer code in MATLAB for EVPS algorithm is presented in Appendix 1, and a hypothetical objective function is presented in Appendix 2.

2. OPTIMIZATION ALGORITHMS

In this part, Vibrating Particles System **(**VPS) and Enhanced Vibrating Particles System (EVPS) algorithms are presented.

2.1 Vibrating particles system algorithm

Vibrating particles system (VPS) is a new meta-heuristic algorithm based on the free vibration of freedom system' single degree with viscous damping [18]. In this meta-heuristic algorithm, firstly the initial agents are generated in permissible range by:

$$x_i^j = x_{min} + rand.(x_{max} - x_{min}) \tag{1}$$

where x_i^j is the *j*th variable of the *i*th particle. x_{min} and x_{max} are the starting and ending points of permissible search space for the *j*th variable and *rand* is a random number in the range of [0, 1].

In the VPS algorithm three parameters are defined as:

1. *HB* (the historically best position of the entire population) is the best candidate until that iteration.
2. *GB* (a good particle) is selected randomly between partially best answers in each iteration.
3. BP (a bad particle) is selected randomly between partially worst answers in each iteration.

A descending function according to Eq. (2) is defined. This parameter is presented to include the effect of the damping level in the vibration.

$$D = (\frac{iter}{iter_{max}})^{-\alpha} \tag{2}$$

where *iter* is the number of current iteration, $iter_{max}$ is the total number of iterations and α is a constant value. Creating the next agents in VPS algorithm is updated using the following equation:

$$\begin{gathered} x_i^j = \omega_1.\left[D.A.rand1 + HB^j\right] + \omega_2.\left[D.A.rand2 + GP^j\right] + \omega_3.[D.A.rand3 + BP^j] \\ A = \left[\omega_1.\left(HB^j - x_i^j\right)\right] + \left[\omega_2.\left(GP^j - x_i^j\right)\right] + \left[\omega_3.\left(BP^j - x_i^j\right)\right] \\ \omega_1 + \omega_2 + \omega_3 = 1 \end{gathered} \tag{3}$$

where $\omega_{1,2}$ and ω_3 are of relative importance to *HB*, *GB* and *BP*, respectively and *ran1*, *rand2* and *rand3* are random numbers uniformly distributed in the range [0, 1].

A parameter like p (within 0 to 1) is defined to accelerate the convergence of the VPS algorithm. This parameter is compared with rand and if $p < rand$, then $\omega_3=0$ and $\omega_2=1-\omega_1$. Pseudo code of the vibrating particles system algorithm is presented in Fig. 1 adopted from Kaveh and Ilchi Ghazaan [19].

```
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
                New location is obtained by Eq. (3)
                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 1. Pseudo code of the standard vibrating particles system algorithm [19]

When an agent violates a boundary, it is changed by the harmony search-based side constraint handling approach. In this technique, *HMCR* (harmony memory considering rate) parameter determines whether the violating component should be changed with the corresponding value in *HB* or it must be selected from the permissible search space. Also, if changed with *HB*, there is a parameter, *PAR* (pitch adjusting rate), that determines whether this value should be changed with the neighboring value or not. This process is repeated for $iter_{max}$ times.

2.2. Enhanced vibrating particles system algorithm

In this section, Enhanced Vibrating Particles System (EVPS) is presented which in main follows [2]. This improvement results in increasing the convergence speed, augmenting the

ability of search, helping the EVPS to escape from local optima and overall resulting in better results. Changes to the standard version of the VPS algorithm are as follows:

In this method, *Memory* parameter acts as *HB* with the difference that it saves *Memorysize* number of the best historically positions from the entire population. It should be noted that if the best answer of each *iteration* is better than the worst value of the *Memory*, it should replace the worst value in the memmory.

OHB (one of the best historically positions in entire population) is one row of *Memory* that is selected randomly. *HB* is replaced with *Memory* in the EVPS algorithm. Another change made to the VPS algorithm is that Eq. (4) is replaced with Eq. (3). In Eq. (4), one of the *(a)*, *(b)* and *(c)* equations are used with the possibility of $\omega_{1,2}$ and ω_3, respectively.

$$x_i^j = \begin{cases} \left[D.A.rand1 + OHB^j\right] & (a) \\ \left[D.A.rand2 + GP^j\right] & (b) \\ \left[D.A.rand3 + BP^j\right] & (c) \end{cases}$$
$$A = \begin{cases} (\pm 1)\left(OHB^j - x_i^j\right) & (a) \\ (\pm 1)\left(GP^j - x_i^j\right) & (b) \\ (\pm 1)\left(BP^j - x_i^j\right) & (c) \end{cases} \qquad (4)$$
$$\omega_1 + \omega_2 + \omega_3 = 1$$

where (±1) are used randomly. It should be noted that *OHB*, *GP* and *BP* are determined for every agent independently. Other parts of the EVPS are exactly the same as in the standard VPS algorithm.

3. FORMULATION

In this formulation, the aim is to minimize the weight of structures besides satisfying certain design constraints. These constraints include strength and displacements constraints according to LRFD-AISC specification [20]. The mathematical formulation can be expressed as:

$$\text{Find } \{x\} = [x_1, x_2, ..., x_{ng}] \quad x_i \in S_i$$
$$\text{To minimize } W(\{x\}) = \sum_{i=1}^{nm} \rho_i A_i L_i \qquad (5)$$
$$\text{Subjected to } \begin{cases} g_j(\{x\}) \le 0, \quad j = 1, 2, ..., nc \\ x_{i_{\min}} \le x_i \le x_{i_{\max}} \end{cases}$$

where *{x}* is a set of design variables containing the cross sectional area; *ng* is the numbers of member groups (number of design variables); W(*{x}*) is the weight of the structure; *nm* is the number of elements of the structure; *nc* is the number of constraints; ρ_i presents the

material density of *i*th member; A_i and L_i denote the cross-sectional area and the length of member *i*, respectively.

4. NUMERICAL PROBLEMS

Two benchmark problems are presented to investigate the performance of the VPS and EVPS algorithms. The values of population size, the total number of iteration, p, w1, w2, HMCR, PAR and *Memorysize* are 50, 1000, 0.2, 0.3, 0.3, 0.95, 0.1 and 4 for all problems, respectively. Thirty independent optimization runs are carried out for all the problems.

4.1 A 72-bar spatial truss

For the 72-bar spatial truss structure shown in Fig. 2, the elements are categorized in 16 design groups:

(1) A1_A4, (2) A5_A12, (3) A13_A16, (4) A17_A18, (5) A19_A22, (6) A23_A30, (7) A31_A34, (8) A35_A36, (9) A37_A40, (10) A41_A48, (11) A49_A52, (12) A53_A54, (13) A55_A58, (14) A59_A66 (15), A67_A70, and (16) A71_A72.

The material density is 0.1 lb/in^3 and the modulus of elasticity is taken as 10,000 ksi. The members are subjected to the stress limits of ±25 ksi. The nodes are subjected to the displacement limits of ±0.25 in. The minimum permitted cross-sectional area of each member is taken as 0.10 in^2, and the maximum cross-sectional area of each member is 4.00 in^2. The loading conditions are considered as:

1. Loads 5, 5 and -5 kips in the x, y and z directions at node 17, respectively.
2. A load -5 kips in the z direction at nodes 17, 18, 19 and 20.

Table 1 shows the results obtained by VPS and EVPS algorithms. Fig. 3 presents the convergence diagrams for VPS and EVPS algorithms. According to Table 1, the lightest weight is achieved by EVPS and this algorithm has reached best average weight among the other algorithms. Fig. 3 shows the best of one hundred iterations convergence history for the VPS and EVPS algorithms for the best answer.

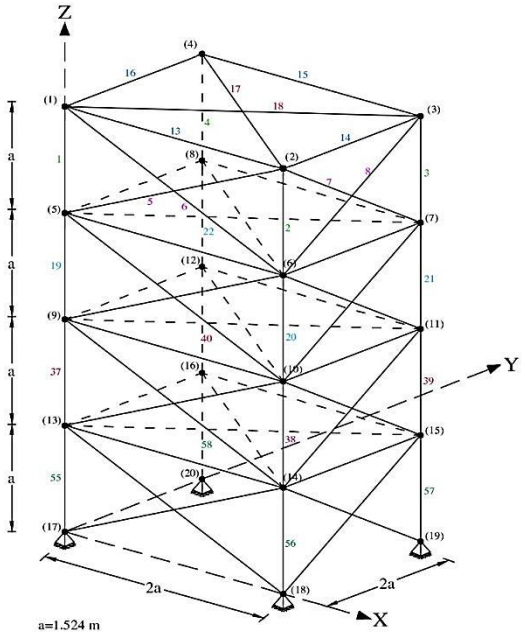


Figure 2. Schematic of the 72-bar spatial truss

Table 1: Comparison of the VPS and EVPS results with other algorithms for the 72-bar spatial truss

Element Group		Optimal cross-sectional areas (in^2)								
		BB-BC [21]	ACO [22]	RO [9]	CBO*	ECBO*	PSO	LCA-Tie-II [25]	VPS	EVPS
1	A1-A4	0.1565	0.156	0.1576	0.156524	0.155807	0.156624	0.1561	0.164272	0.156026
2	A5-A12	0.5507	0.55	0.5222	0.54469	0.561095	0.542412	0.5449	0.585029	0.550565
3	A13-A16	0.3922	0.39	0.4356	0.409962	0.391304	0.414956	0.4131	0.341609	0.41266
4	A17-A18	0.5922	0.592	0.5971	0.568453	0.571421	0.579202	0.5779	0.581378	0.568612
5	A19-A22	0.5209	0.561	0.573	0.522421	0.541031	0.51698	0.5337	0.442988	0.536759
6	A23-A30	0.5172	0.492	0.5499	0.517209	0.530236	0.522745	0.5175	0.493931	0.519986
7	A31-A34	0.1004	0.1	0.1004	0.100002	0.1	0.1	0.1003	0.105423	0.1
8	A35-A36	0.1005	0.107	0.1001	0.100004	0.1	0.100005	0.1014	0.1492	0.100734
9	A37-A40	1.2476	1.303	1.2522	1.268652	1.202484	1.286803	1.2592	1.365438	1.26445
10	A41-A48	0.5269	0.511	0.5033	0.511534	0.505656	0.508058	0.5061	0.492803	0.508508
11	A49-A52	0.1	0.101	0.1002	0.100001	0.100002	0.1	0.1001	0.108361	0.100002
12	A53-A54	0.1012	0.1	0.1001	0.100002	0.1	0.1	0.1	0.100898	0.1
13	A55-A58	1.8577	1.948	1.8365	1.886173	1.86922	1.884583	1.8747	1.936977	1.858562
14	A59-A66	0.5059	0.508	0.5021	0.514037	0.515421	0.506424	0.4164	0.54127	0.511243
15	A67-A70	0.1	0.101	0.1	0.1	0.10001	0.100038	0.1	0.107713	0.1
16	A71-A72	0.1	0.102	0.1004	0.100001	0.100001	0.100015	0.1	0.10143	0.100001
Best weight (lb)		379.85	380.24	380.458	379.61	379.93	379.65	379.93	384.1197	379.6482
Worst weight (lb)		N/A	N/A	N/A	379.74	380.53	379.76	380.2037	392.9363	380.0435
Average weight (lb)		382.08	383.16	382.553	379.66	380.20	380.30	380.5306	387.7259	379.7143

*CBO and ECBO algorithms were used for optimization of this problem by Kaveh and Mahdavi [23] and Kaveh and Ilchi Ghazaan [24], respectively. These results obtained by the authors in this article.

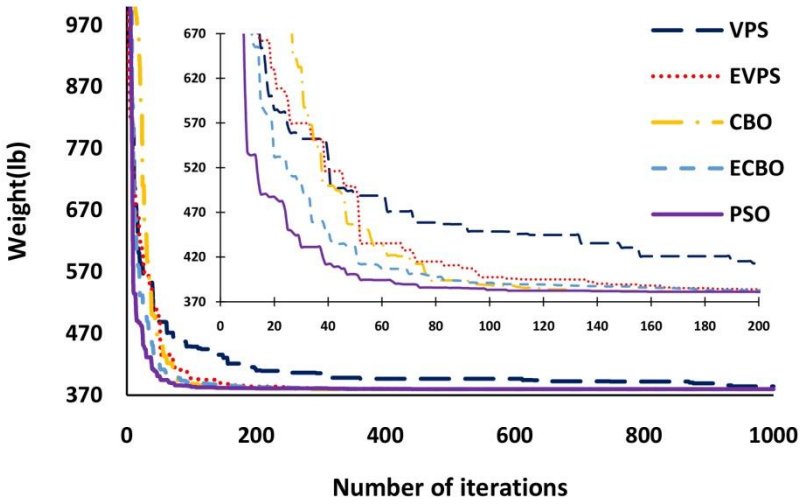


Figure 3. The convergence curves for the spatial 72-bar spatial truss

4.1 A 582-bar tower truss

The schematic of the 582-bar tower truss with the height of 80 m is presented in Fig. 4. The symmetry of the tower around x-axis and y-axis is considered to group the 582 members into 32 independent size variables. A single load case is considered such that it consists of lateral loads of 5.0 kN applied in both x and y directions and a vertical load of -30 kN applied in the z-direction at all nodes of the tower. A discrete set of 137 economical standard steel sections selected from W-shape profile list based on area and radii of gyration properties is used to size the variables. The lower and upper bounds on size variables are taken as 39.74 cm^2 and 1387.09 cm^2, respectively. The stress limitations of the members are imposed according to the provisions of ASD-AISC [20]. The other constraint is the limitation of nodal displacements (these should not be more than 8.0 cm or 3.15 in. in any direction). Also, the maximum slenderness ratio is limited to 300 for tension members, and it is recommended to be 200 for compression members according to ASD-AISC design code provisions [20]. Table 2 shows the results obtained by VPS and EVPS algorithms. According to Table 2, the EVPS algorithm has reached better answer in comparison to the standard VPS algorithm in the best, worst and average answers. Fig. 5 presents the allowable and existing stress ratio and displacement values in 3 directions for the VPS and EVPS algorithms. Fig. 6 shows the best of one hundred iterations convergence histories for the VPS and EVPS algorithms for the best answer.

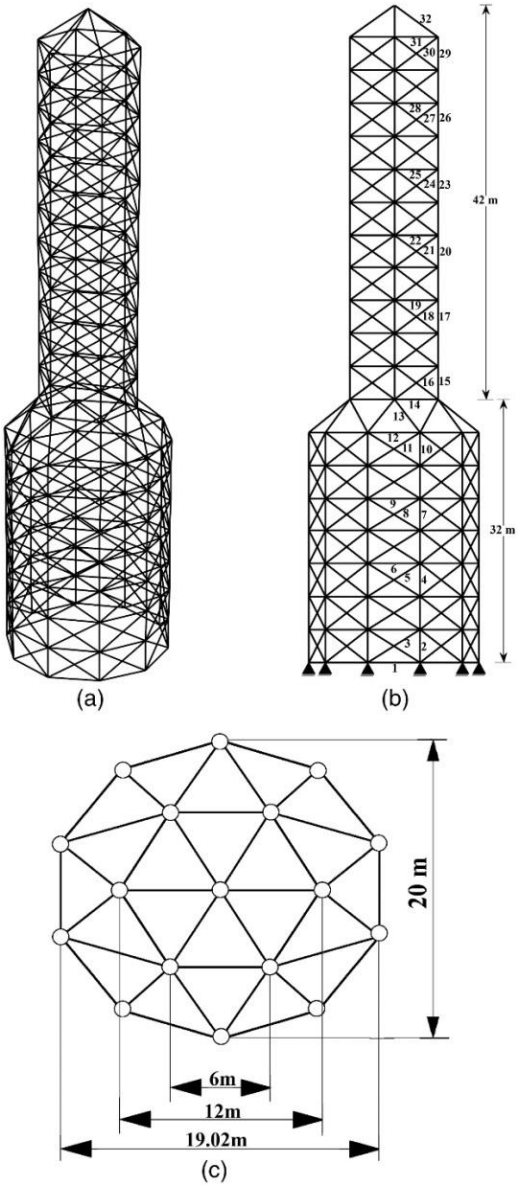


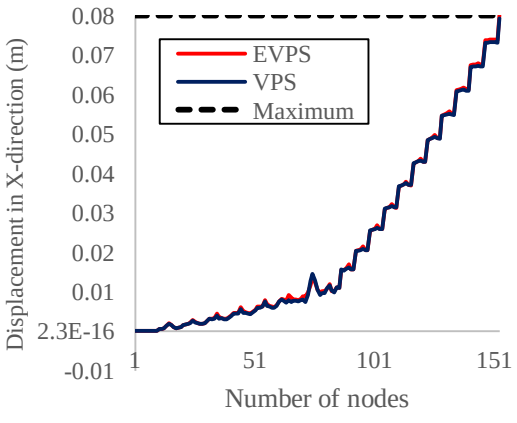
Figure 4. Schematic of the 582-bar tower truss

Table 2: Comparison of the VPS and EVPS results with other algorithms for the 582-bar tower truss

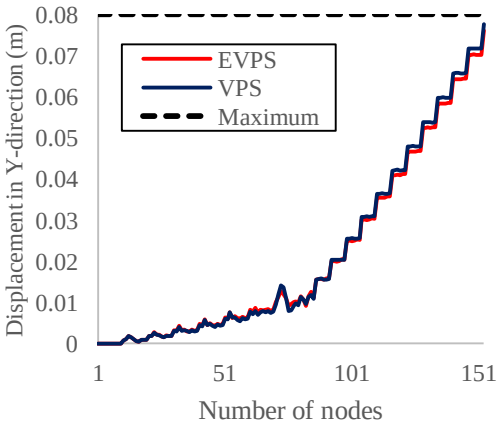
Element Group	Optimal cross-sectional areas (cm^2)						
	PSO*	CBO	ECBO	DHPSACO [27]	SDE [1]	VPS	EVPS
1	39.74186	39.74186	39.74186	45.68	39.74186	41.87088	41.87088
2	136.1288	136.1288	163.2255	136.13	136.1288	140.6449	136.1288
3	53.2257	53.2257	45.67733	53.16	53.2257	53.2257	53.2257
4	114.1933	115.4836	115.4836	109.68	115.4836	170.9674	115.4836
5	45.67733	45.67733	45.67733	45.68	45.67733	45.67733	45.67733
6	39.74186	39.74186	39.74186	45.68	39.74186	41.87088	39.74186
7	85.80628	81.29016	94.19336	92.9	92.90304	74.1934	90.96756

8	45.67733	45.67733	45.67733	45.68	45.67733	45.67733	45.67733
9	39.74186	39.74186	39.74186	45.68	39.74186	41.87088	41.87088
10	84.51596	84.51596	84.51596	75.48	85.80628	100.645	85.80628
11	45.67733	45.67733	45.67733	56.71	45.67733	53.2257	45.67733
12	128.3868	128.3868	136.1288	136.129	123.2256	68.38696	128.3868
13	140.6449	140.6449	126.4514	143.87	146.4513	170.9674	140.6449
14	92.90304	92.90304	94.19336	92.9	92.90304	118.0643	92.90304
15	140.6449	140.6449	140.6449	154.84	143.8707	123.2256	140.6449
16	74.1934	74.1934	58.90311	58.84	58.90311	58.90311	58.90311
17	115.4836	118.0643	109.6772	115.48	115.4836	109.6772	115.4836
18	45.67733	45.67733	47.35474	45.68	45.67733	45.67733	45.67733
19	39.74186	39.74186	39.74186	39.74	39.74186	58.90311	39.74186
20	75.48372	75.48372	75.48372	75.48	64.516	94.19336	74.1934
21	45.67733	45.67733	45.67733	45.68	45.67733	47.35474	45.67733
22	39.74186	41.87088	41.87088	41.87	39.74186	41.87088	41.87088
23	41.87088	45.67733	39.74186	58.84	47.35474	58.90311	45.67733
24	45.67733	45.67733	45.67733	53.16	45.67733	53.2257	45.67733
25	39.74186	39.74186	41.87088	39.74	39.74186	49.35474	39.74186
26	39.74186	39.74186	39.74186	39.74	39.74186	41.87088	41.87088
27	45.67733	45.67733	45.67733	45.68	45.67733	45.67733	47.35474
28	39.74186	39.74186	39.74186	53.16	39.74186	41.87088	41.87088
29	39.74186	39.74186	56.70956	68.39	39.74186	64.516	41.87088
30	45.67733	47.35474	45.67733	45.68	45.67733	57.09666	45.67733
31	39.74186	39.74186	39.74186	39.74	39.74186	64.516	41.87088
32	45.67733	45.67733	45.67733	45.68	49.35474	72.25792	45.67733
Best weight (m^3)	21.38	21.41	21.16	22.0607	21.225	22.8384	21.3352
Worst weight (m^3)	21.89	21.69	21.71	N/A	N/A	24.1290	22.0811
Average weight (m^3)	21.57	21.52	21.37	N/A	N/A	23.2414	21.5548

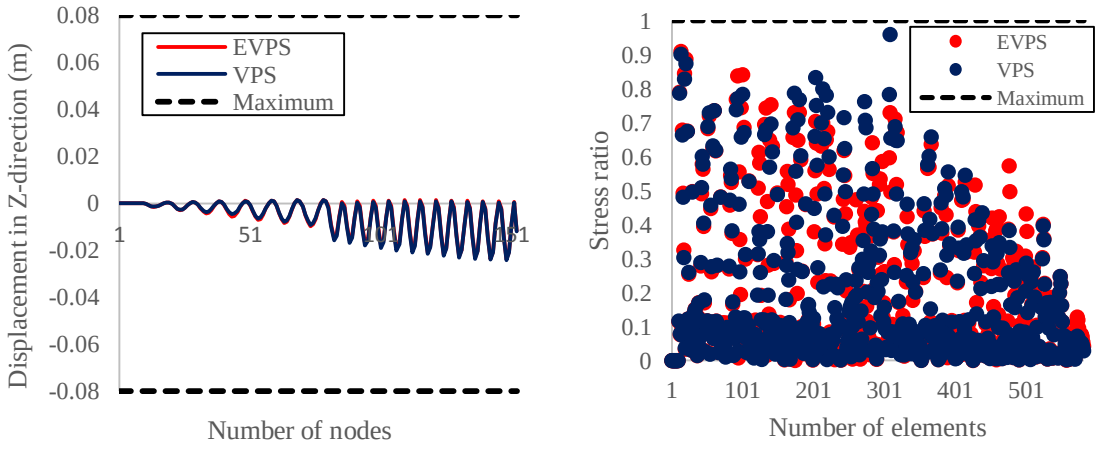
*This truss was optimized using the PSO algorithm by Kaveh and Talatahari [26] but the reported results in this article are obtained by authors.



(a)



(b)



(c) (d)

Figure 5. Comparison of the allowable and existing constraints for the 582-bar truss using the VPS and EVPS algorithms. (a) Displacement in the X-direction. (b) Displacement in the Y-direction. (c) Displacement in the Z-direction. (d) Stress ratio

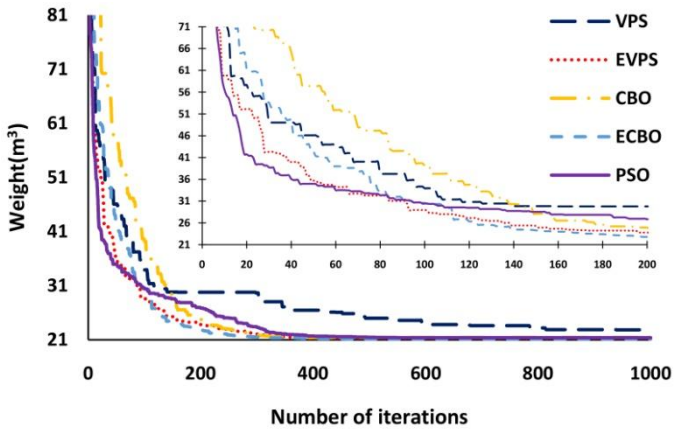


Figure 6. The convergence curves for the 582- bar tower truss

5. CONCLUSION

In this paper a code is developed for an enhanced version of the vibrating particles system (EVPS). Two well-studied weight optimization of steel truss structures problems are investigated to show the efficiently of the EVPS in comparison with its standard version VPS. In the paper, EVPS and VPS algorithms are used for continuous and discrete optimization problems. For the considered examples, the EVPS algorithm has reached better results compared to the VPS algorithm and speed of convergence is higher. The results of two benchmark problems illustrate that the EVPS has a good performance and it can be utilized for other optimization problems. The results show that the EVPS is also competitive with other meta-heuristic algorithms. Finally, MATLAB code for the EVPS algorithm is presented in Appendix 1 and a hypothetical objective function is presented in Appendix 2.

REFERENCES

1. Kaveh A, Hosseini P. A simplified dolphin echolocation optimization method for optimum design of trusses, *Int J Optim Civil Eng* 2014; **4**(3): 381-97.
2. Kaveh A, Hosseini Vaez S, Hosseini P. Enhanced vibrating particles system algorithm for damage identification of truss structures, *Sci Iran* 2017.
3. Kaveh A, Hoseini Vaez SR, Hosseini P. Modified dolphin monitoring operator for weight optimization of frame structures, *Periodica Polytech Civil Eng* 2017; **61**(4): 770-9.
4. Kaveh A. *Advances in Metaheuristic Algorithms for Optimal Design of Structures, Springer International Publishing*, 2nd edition, Switzerland, 2017.
5. Kaveh A. *Applications of Metaheuristic Optimization Algorithms in Civil Engineering*, Springer International Publishing, Switzerland, 2017.
6. Goldberg DE, Holland JH. Genetic algorithms and machine learning, *Machine Learn* 1988; **3**(2):95-99.
7. Kennedy J, Eberhart R. Particle swarm optimization In: Neural Networks, *Proceedings IEEE International Conference on 1995*; pp. 1942-1948.
8. Kaveh A, Talatahari S. A novel heuristic optimization method: charged system search, *Acta Mech* 2010; **213**(3): 267-89.
9. Kaveh A, Khayatazad M. A new meta-heuristic method: ray optimization, *Comput Struct* 2012; **112**: 283-94.
10. Kaveh A, Mahdavi VR. Colliding bodies optimization: a novel meta-heuristic method, *Comput Struct* 2014; **139**: 18-27.
11. Kaveh A, Zolghadr A. Optimal design of cyclically symmetric trusses with frequency constraints using cyclical parthenogenesis algorithm, *Adv Struct Eng* 2017; 1369433217732492.
12. Dorigo M, Birattari M, Stutzle T. Ant colony optimization, *IEEE Comput Intell Magaz* 2006; **1**(4): 28-39.
13. Kaveh A, Farhoudi N. A new optimization method: dolphin echolocation, *Adv Eng Softw* 2013, **59**: 53-70.
14. Atashpaz-Gargari E, Lucas C. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, In: *Evolutionary Computation 2007 CEC 2007 IEEE Congress on IEEE* 2007; pp. 4661-4667.
15. Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: harmony search, *Simulation* 2001; **76**(2): 60-8.
16. Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer, *Adv Eng Softw* 2014; **69**: 46-61.
17. Kaveh A, ILchi Ghazaan M. A new meta-heuristic algorithm: vibrating particles system, *Sci Iran: Transaction A: Civil Eng* 2017; **24**(2): 551.
18. Kaveh A, Ilchi Ghazaan M. Vibrating particles system algorithm for truss optimization with multiple natural frequency constraints, *Acta Mech* 2017; **228**(1): 307-22.
19. Kaveh A, Ilchi Ghazaan M. Matlab code for vibrating particles system algorithm, *Int J Optim Civil Eng* 2017; **7**(3): 355-66.
20. Load and Resistance Factor Design (LRFD). American Institute of Steel Construction (AISC), Load and resistance factor design, Chicago, 2001.

21. Camp CV. Design of space trusses using Big Bang–Big Crunch optimization, *J Struct Eng* 2007; **133**(7): 999-1008.
22. Camp CV, Bichon BJ. Design of space trusses using ant colony optimization. *J Struct Eng* 2004; **130**(5): 741-51.
23. Kaveh A, Mahdavi VR. Colliding bodies optimization method for optimum design of truss structures with continuous variables, *Adv Eng Softw* 2014; **70**: 1-12.
24. Kaveh A, Ilchi Ghazaan M. Enhanced colliding bodies optimization for design problems with continuous and discrete variables, *Adv Eng Softw* 2014; **77**: 66-75.
25. Jalili S, Kashan AH, Hosseinzadeh Y. League Championship Algorithms for Optimum Design of Pin-Jointed Structures, *J Comput Civil Eng* 2016; **31**(2):04016048.
26. Hasançebi O, Çarbaş S, Doğan E, Erdal F, Saka M. Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures, *Comput Struct* 2009; **87**(5): 284-302.
27. Kaveh A, Talatahari S. A particle swarm ant colony optimization for truss structures with discrete variables, *J Construct Steel Res* 2009; **65**(8): 1558-68.

APPENDIX 1: EVPS CODE IN MATLAB (THIS CODE IS APPLIED IN VPS CODE [19])

```
The EVPS code in MATLAB
% Enhanced VIBRATING PARTICLES SYSTEM - EVPS
% clear memory
clear all
% Initializing variables
popSize=50; % Size of the population
nVar=4; % Number of optimization variables
maxIt=1000; % Maximum number of iteration
xMin=-100; % Lower bound of the variables
xMax=100; % Upper bound of the variables
alpha=0.05;
w1=0.3;
w2=0.3;
w3=1-w1-w2;
p=0.2; % With the probability of (1-p) the effect of BP
is ignored in updating
PAR=0.1;
HMCR=0.95;
Memorysize=4;
result=zeros(popSize,nVar+1);
Memory=zeros(Memorysize,nVar+1);
neighbor=0.1;  % Parameters for handling the side
constraints
% Initializing particles
position=(xMin+rand(popSize,nVar).*(xMax-xMin));
```

```
% Search
agentCost=zeros(popSize,3); % Array of agent costs
HBV=zeros(popSize,nVar+2); % Historically best matrix
for iter=1:maxIt
% Evaluating and storing
for m=1:popSize
[penalizedWeight,weight]=SphereFun(position(m,:));       %
Evaluating the objective function for each particle
agentCost(m,1)=penalizedWeight;
agentCost(m,2)=m;
agentCost(m,3)=weight;
end
sortedAgentCost=sortrows(agentCost);
for m=1:popSize
if iter==1 || agentCost(m,1)<HBV(m,1)
HBV(m,1)=agentCost(m,1);
HBV(m,2)=agentCost(m,3);
for n=1:nVar
HBV(m,n+2)=position(m,n);
end
end
end
sortedHBV=sortrows(HBV);
result(iter,1)=sortedHBV(1,1);
result(iter,2:nVar+1)=sortedHBV(1,3:nVar+2);
if iter==1
    for i=1:Memorysize
        Memory(i,1)=sortedHBV(i,1);
        Memory(i,2:nVar+1)=sortedHBV(i,3:nVar+2);
    end
end
if Memory(Memorysize,1)>sortedHBV(1,1)
    Memory(Memorysize,1)=sortedHBV(1,1);
    Memory(Memorysize,2:nVar+1)=sortedHBV(1,3:nVar+2);
end
Memory=sortrows(Memory);
% Updating particle positions
D=(iter/maxIt)^(-alpha);
for m=1:popSize
temp1=ceil(unifrnd(1,popSize/2,1));
temp2=ceil(unifrnd(popSize/2,popSize,1));
temp11=round(unifrnd(1,Memorysize,1));
    if p<rand
        w3=0;
        w2=1-w1;
    end
```

```
for n=1:nVar
    if rand<w3
       A(m,n)=(-
1)^(ceil(randn()))*(position(sortedAgentCost(temp2,2),n)-
position(m,n));
position(m,n)=D*A(m,n)*rand+position(sortedAgentCost(temp
2,2),n);
    elseif rand<w2
        A(m,n)=(-
1)^(ceil(randn()))*(position(sortedAgentCost(temp1,2),n)-
position(m,n));
position(m,n)=D*A(m,n)*rand+position(sortedAgentCost(temp
1,2),n);
    else
        A(m,n)=(-1)^(ceil(randn()))*(Memory(temp11,1+n)-
position(m,n));
        position(m,n)=D*A(m,n)*rand+Memory(temp11,1+n);
    end
end
w2=0.3;w3=1-w1-w2;
end
% Handling the side constraints
for m=1:popSize
    for n=1:nVar
if position(m,n)<xMin || position(m,n)>xMax
temp1=rand;temp2=rand;temp3=ceil(rand*popSize);
if temp1<=HMCR && temp2<=(1-PAR)
position(m,n)=sortedHBV(temp3,2+n);
elseif temp1<=HMCR && temp2>(1-PAR)
position(m,n)=sortedHBV(temp3,2+n)+neighbor;
if position(m,n)>xMax
position(m,n)=sortedHBV(temp3,2+n)-2*neighbor;
end
else
position(m,n)=xMin+(rand*(xMax-xMin));
end
end
end
end
iter
sortedHBV(1,1)
end
% Saving the results
xlswrite('result.xls',result);
```

APPENDIX 2: A HYPOTHETICAL OBJECTIVE FUNCTION

```
Function [penalizedWeight,weight]=SphereFun(x)
%% If there is any violation:
%Penalty=sum of violations
%CP=Coefficient of penalty
%penalizedWeight=sum(x.^2)*(1+CP*Penalty);
penalizedWeight=sum(x.^2);
weight=sum(x.^2);
end
```