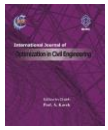
INTERNATIONAL JOURNAL OF OPTIMIZATION IN CIVIL ENGINEERING
Int. J. Optim. Civil Eng., 2017; 7(3):355-366



MATLAB CODE FOR VIBRATING PARTICLES SYSTEM ALGORITHM

A. Kaveh[*, †] and M. Ilchi Ghazaan
Centre of Excellence for Fundamental Studies in Structural Engineering, Iran University of Science and Technology, Narmak, Tehran, P.O. Box 16846-13114, Iran

ABSTRACT

In this paper, MATLAB code for a recently developed meta-heuristic methodology, the vibrating particles system (VPS) algorithm, is presented. The VPS is a population-based algorithm which simulates a free vibration of single degree of freedom systems with viscous damping. The particles gradually approach to their equilibrium positions that are achieved from current population and historically best position. Two truss towers with 942 and 2386 elements are examined for the validity of the present algorithm; however, the performance VPS has already been proven through truss and frame design optimization problems.

Keywords: vibrating particles system algorithm; MATLAB; meta-heuristic; structural optimization.

Received: 25 October 2016 Accepted: 30 January 2017

1. INTRODUCTION

Structural optimization can be classified as follows: 1. obtaining optimal size of structural members (sizing optimization); 2. finding the optimal form for the structure (shape optimization); 3. achieving optimal size and connectivity between structural members (topology optimization). Sizing optimization problems are very popular design problems and can be found frequently in papers [1-5].

Recent developments in meta-heuristic optimization algorithms have made these methods suitable even for complicated design problems and they have been widely employed for obtaining the optimal solutions of engineering design problems. Some of the most recent algorithms in this field are: teaching–learning-based optimization (TLBO) [6], water cycle algorithm (WCA) [7], colliding bodies optimization (CBO) [8], grey wolf optimizer (GWO)

[*]Corresponding author: Centre of Excellence for Fundamental Studies in Structural Engineering, Iran University of Science and Technology, Narmak, Tehran, P.O. Box 16846-13114, Iran. Tel.: +98 21 77240104; fax: +98 21 77240398
[†]E-mail address: alikaveh@iust.ac.ir (A. Kaveh)

[9], ant lion optimizer (ALO) [10], tug of war optimization (TWO) [11], whale optimization algorithm (WOA) [12] and water evaporation optimization (WEO) [13]. Further advances and applications of metaheuristics can be found in Kaveh [14,15].

In this study, a new nature-inspired meta-heuristic optimization algorithm, called vibrating particles system (VPS), is utilized in sizing optimization of tower truss structures and its MATLAB code is presented. This method was introduced by Kaveh and Ilchi Ghazaan [16] and it is inspired by the damped free vibration of single degree of freedom system. In VPS, The solution candidates are considered as particles that gradually approach to their equilibrium positions. Equilibrium positions are achieved from current population and historically best position.

The remainder of the paper is organized as follows. The VPS algorithm is briefly presented in Section 2. In order to show the capability of the proposed algorithm, two numerical examples are studied in Section 3. The last section concludes the paper. Computer code in MATLAB is provided in Appendix 1.

2. VIBRATING PARTICLES SYSTEM

A recent addition to meta-heuristic algorithms is the vibrating particles system that was introduced by Kaveh and Ilchi Ghazaan [16]. The VPS mimics the free vibration of single degree of freedom systems with viscous damping and by utilizing a combination of randomness and exploitation of obtained results, the quality of the particles improves iteratively as the optimization process proceeds. The pseudo code of VPS is provided in Fig. 1 and its code in MATLAB is presented in Appendix 1. The steps of this technique are as follows:

Level 1: Initialization

Step 1: The VPS parameters are set and the initial locations of all particles are determined randomly in the search space.

Level 2: Search

Step 1: The objective function value is calculated for each particle.

Step 2: For each particle, three equilibrium positions with different weights are defined that the particle tends to approach: 1. the best position achieved so far across the entire population (*HB*), 2. a good particle (*GP*) and 3. a bad particle (*BP*). In order to select the *GP* and *BP* for each candidate solution, the current population is sorted according to their objective function values in an increasing order, and then *GP* and *BP* are chosen randomly from the first and second half, respectively.

Step 3: The positions are updated by:

$$x_i^j = w_1.[D.A.rand1 + HB^j] + w_2.[D.A.rand2 + GP^j] + w_3.[D.A.rand3 + BP^j] \tag{1}$$

$$w_1 + w_2 + w_3 = 1 \tag{2}$$

$$D = \left(\frac{iter}{iter_{\max}}\right)^{-\alpha} \tag{3}$$

$$A = [w_1.(HB^j - x_i^j)] + [w_2.(GP^j - x_i^j)] + [w_3.(BP^j - x_i^j)] \tag{4}$$

where x_i^j is the *j*th variable of particle *i*. w_1, w_2 and w_3 are three parameters to measure the relative importance of *HB*, *GP* and *BP*, respectively. *iter* is the current iteration number and $iter_{max}$ is the total number of iteration for optimization process. α is a constant. *rand*1, *rand*2 and *rand*3 are random numbers uniformly distributed in the range of [0,1].

A parameter like *p* within (0, 1) is defined and it is specified whether the effect of *BP* must be considered in updating position or not. For each particle, *p* is compared with *rand* (a random numbers uniformly distributed in the range of [0,1]) and if p < rand, then $w_3 = 0$ and $w_2 = 1 - w_1$.

Step 4: If any component of the system violates a boundary, it must be regenerated by harmony search-based side constraint handling approach. In this technique, there is a possibility like *HMCR* (harmony memory considering rate) that specifies whether the violating component must be changed with the corresponding component of the historically best position of a random particle or it should be determined randomly in the search space. Moreover, if the component of a historically best position is selected, there is a possibility like *PAR* (pitch adjusting rate) that specifies whether this value should be changed with the neighboring value or not.

Level 3: Terminal condition check

Step 1: After the predefined maximum evaluation number, the optimization process is terminated.

```
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. (1)
                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 vibrating particles system algorithm

3. NUMERICAL EXAMPLES

Sizing optimization of skeletal structures can be stated as follows:

$$\begin{aligned} &Find && \{X\} = [x_1, x_2, .., x_{ng}] \\ &to\ minimize && W(\{X\}) = \sum_{i=1}^{nm} \rho_i A_i L_i \end{aligned} \quad (5)$$

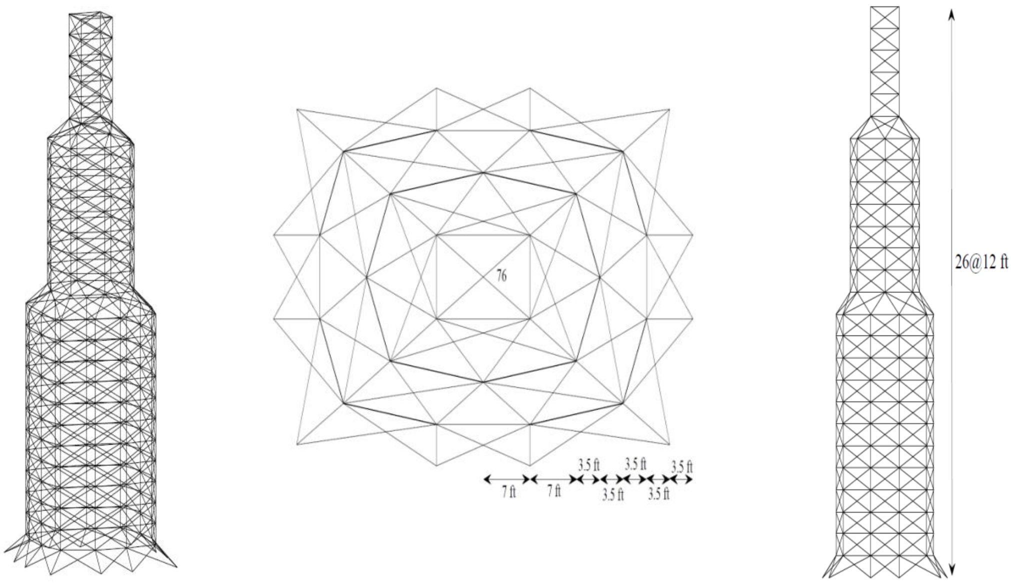
$$subjected\,to: \quad \begin{cases} g_j(\{X\}) \le 0, \quad j = 1,2,...,nc \\ x_{i\,min} \le x_i \le x_{i\max} \end{cases}$$

where [1] is a vector containing the design variables; *ng* is the number of design variables; *W*([1]) is the weight of the structure; *nm* is the number of elements of the structure; ρ_i, A_i and L_i denote the material density, cross-sectional area, and the length of the *i*th member, respectively. x_{imin} and x_{imax} are the lower and upper bounds of the design variable x_i, respectively. g_j([1]) denotes design constraints; *nc* is the number of constraints. The constraints are handled using the well-known penalty approach.

Two benchmark examples are provided to investigate the performance of the VPS algorithm. The values of population size, the total number of iteration, α , *p*, w_1 and w_2 are set to 20, 1500, 0.05, 70%, 0.3 and 0.3 for the examples, respectively. Twenty independent optimization runs are carried out for all the examples. The algorithm is coded in MATLAB and the structures are analyzed using the direct stiffness method by our own codes.

3.1 A spatial 942-bar tower

The schematic of a 942-bar tower truss is shown in Fig. 2 (the ground-level nodes being fixed). The elements are divided into 76 groups and member groups are presented in Fig. 3. A single load case is considered consisting of the lateral loads of 1.12 kips (5.0 kN) applied in both x- and y-directions and a vertical load of -6.74 kips (-30 kN) is applied in the z-direction at all nodes of the tower. A discrete set of standard steel sections selected from W-shape profile list based on area and radii of gyration properties is used as sizing variables. Cross-sectional areas of the elements are supposed to vary between 6.16 and 215 in^2 (i.e. between 39.74 and 1387.09 cm^2). Limitation on stress and stability of truss elements are imposed according to the provisions of the ASD-AISC [17].



3D view Top view Side view

Figure 2. Schematic of the spatial 942-bar tower

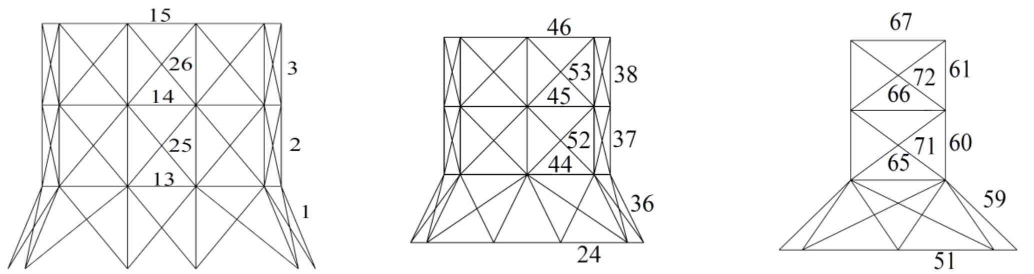


Figure 3. Member groups of spatial 942-bar tower

Table 1 presents the results obtained by the ECBO [18] and VPS. The proposed method obtained 3,296,202 m^3 which is better than 3,376,968 m^3 found by the ECBO. The average optimized weight and standard deviation on average weight of the VPS are, respectively, 3,346,822 m^3 and 41,617 m^3. The best designs have been located in 19,960 and 26,180 analyses for ECBO and VPS, respectively. Fig. 4 shows the convergence curves of the best results obtained by these algorithms.

Table 1: Comparison of optimized designs obtained for the spatial 942-bar tower problem

No.	Sections		No.	Sections		No.	Sections	
	ECBO [16]	VPS		ECBO [16]	VPS		ECBO [16]	VPS
1	W12×190	W12×170	27	W10×33	W8×24	53	W6×25	W10×22
2	W36×230	W36×260	28	W6×25	W8×24	54	W8×21	W10×22
3	W40×199	W44×262	29	W8×31	W12×26	55	W8×21	W10×22
4	W24×229	W30×235	30	W8×31	W10×22	56	W8×21	W10×22
5	W36×150	W36×245	31	W8×21	W8×21	57	W8×21	W8×21
6	W30×173	W24×229	32	W12×26	W10×22	58	W8×21	W10×22
7	W24×250	W40×199	33	W8×21	W8×21	59	W21×62	W14×43
8	W27×258	W14×193	34	W8×21	W10×22	60	W12×152	W24×117
9	W14×159	W40×174	35	W8×21	W8×21	61	W14×120	W18×119
10	W30×191	W24×162	36	W18×86	W16×89	62	W12×65	W14×38
11	W18×158	W14×145	37	W30×191	W30×211	63	W14×30	W10×77
12	W18×119	W18×119	38	W30×116	W14×109	64	W8×21	W14×61
13	W24×250	W12×279	39	W27×178	W24×131	65	W8×21	W10×22
14	W14×30	W8×21	40	W24×131	W21×101	66	W8×21	W10×22
15	W8×21	W10×22	41	W18×86	W10×88	67	W8×21	W8×21
16	W8×21	W12×26	42	W10×88	W10×77	68	W8×21	W10×22
17	W8×21	W10×22	43	W21×62	W12×50	69	W8×21	W10×22
18	W8×21	W10×22	44	W12×136	W27×114	70	W8×21	W10×22
19	W8×21	W10×22	45	W8×21	W10×22	71	W8×24	W8×31
20	W8×21	W10×22	46	W8×21	W10×22	72	W8×24	W10×22
21	W8×21	W6×25	47	W8×21	W10×22	73	W8×21	W12×26
22	W8×21	W8×24	48	W8×21	W6×25	74	W8×21	W10×22
23	W8×21	W10×22	49	W8×21	W10×22	75	W8×21	W8×21
24	W24×117	W14×145	50	W8×21	W8×40	76	W8×21	W8×28
25	W12×50	W8×31	51	W27×94	W12×58	Best volume ($in.^3$)	3,376,968	3,296,202
26	W14×30	W8×24	52	W10×22	W6×25			

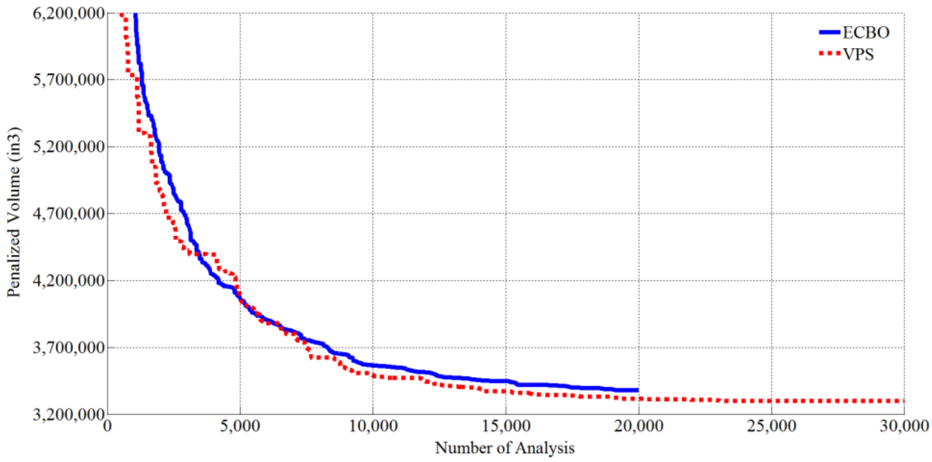


Figure 4. The convergence curves for the spatial 942-bar tower

3.2 A spatial 2386-bar tower

The schematic of a 2386-bar tower truss is shown in Fig. 5 (the ground-level nodes being fixed). The elements are divided into 220 groups and member groups are presented in Fig. 6. The Performance constraints and other conditions are the same as those of the first example.

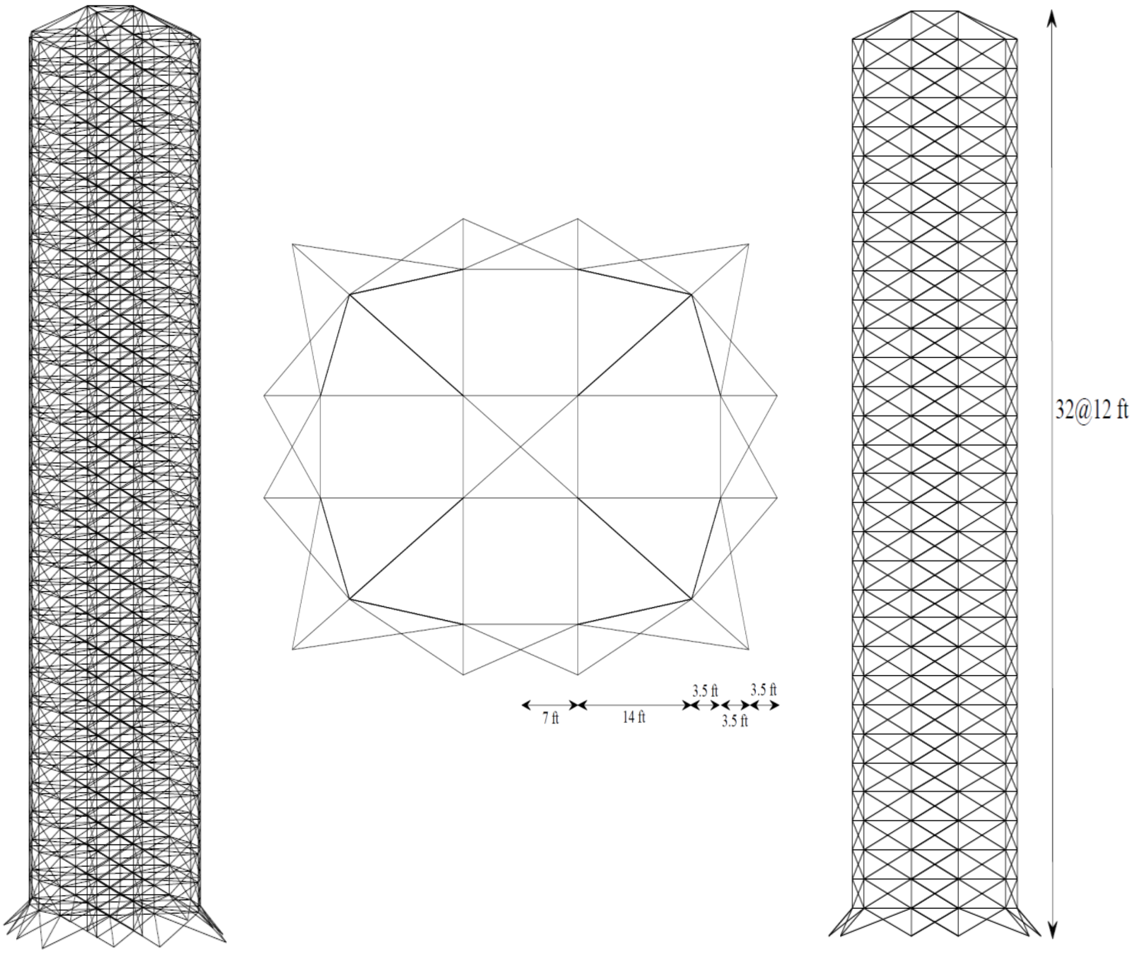
The designs optimized by ECBO [18] and VPS are compared in Table 2. The best designs are found by ECBO and VPS as 14,086,857 m^3 and 12,989,713 m^3, respectively. The average optimized weight and standard deviation on average weight of the VPS are 13,371,681 m^3 and 267,601 m^3, respectively. The best designs are achieved after 29,670 and 29,980 analyses by ECBO and VPS, respectively. Fig. 7 compares the best convergence histories of the algorithms.

Table 2: Comparison of optimized designs obtained for the spatial 2386-bar tower problem

No.	Sections		No.	Sections		No.	Sections	
	ECBO [16]	VPS		ECBO [16]	VPS		ECBO [16]	VPS
1	W14×730	W14×665	75	W14×38	W16×36	149	W8×21	W6×25
2	W14×730	W14×605	76	W12×65	W10×68	150	W14×34	W10×22
3	W14×730	W14×665	77	W14×90	W10×60	151	W10×22	W10×22
4	W14×665	W14×665	78	W12×65	W14×34	152	W12×30	W8×24
5	W14×730	W14×605	79	W30×116	W14×43	153	W8×21	W12×26
6	W14×730	W14×665	80	W14×90	W8×35	154	W10×22	W8×28
7	W14×730	W14×665	81	W18×76	W21×62	155	W8×24	W8×31
8	W40×215	W14×665	82	W14×48	W12×45	156	W27×146	W12×79
9	W14×665	W14×605	83	W10×68	W12×26	157	W14×48	W10×22
10	W14×500	W14×665	84	W8×28	W12×50	158	W8×21	W10×22
11	W12×279	W14×665	85	W10×60	W6×25	159	W14×34	W8×24
12	W33×318	W14×426	86	W14×38	W10×22	160	W8×21	W10×45
13	W14×605	W14×665	87	W10×45	W10×22	161	W10×22	W33×201
14	W14×730	W14×426	88	W12×50	W10×22	162	W6×25	W14×34
15	W14×455	W14×605	89	W14×82	W16×36	163	W8×21	W12×65
16	W33×221	W14×550	90	W8×40	W6×25	164	W8×24	W12×30
17	W44×335	W36×245	91	W10×22	W12×26	165	W10×22	W10×22

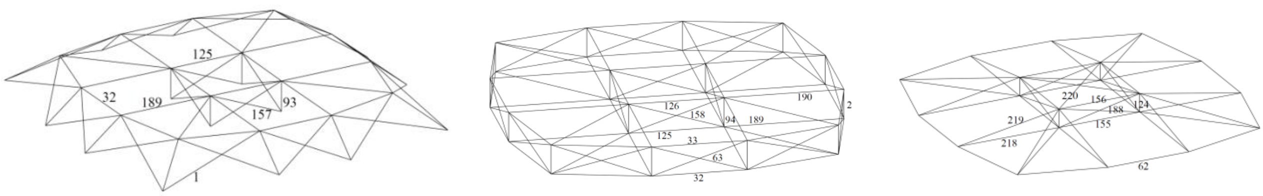
18	W14×426	W33×291	92	W8×21	W10×22	166	W8×24	W10×22
19	W33×221	W33×263	93	W8×21	W10×22	167	W8×21	W6×25
20	W24×229	W30×292	94	W12×40	W8×21	168	W8×21	W8×28
21	W14×145	W33×221	95	W12×40	W14×82	169	W14×34	W14×30
22	W12×252	W18×158	96	W8×21	W10×22	170	W10×22	W8×24
23	W27×194	W18×158	97	W10×39	W12×79	171	W8×31	W10×22
24	W36×245	W12×136	98	W14×30	W21×93	172	W6×25	W8×21
25	W27×161	W14×109	99	W14×48	W14×30	173	W10×22	W12×26
26	W33×118	W44×335	100	W10×88	W10×22	174	W8×21	W8×21
27	W33×201	W18×86	101	W12×50	W14×48	175	W6×25	W10×22
28	W8×21	W14×30	102	W14×34	W8×24	176	W8×24	W10×22
29	W14×90	W10×33	103	W14×43	W12×26	177	W8×21	W8×24
30	W8×21	W8×21	104	W12×65	W8×31	178	W8×21	W12×45
31	W8×35	W14×61	105	W12×53	W12×26	179	W8×21	W8×24
32	W30×211	W14×605	106	W12×26	W12×45	180	W8×21	W14×38
33	W14×120	W14×120	107	W8×21	W14×38	181	W8×21	W10×22
34	W16×67	W10×22	108	W6×25	W14×38	182	W10×22	W12×26
35	W10×100	W14×30	109	W10×39	W10×22	183	W6×25	W10×22
36	W12×26	W10×22	110	W8×28	W10×39	184	W8×21	W12×58
37	W8×31	W6×25	111	W10×39	W16×89	185	W8×21	W14×34
38	W33×118	W10×22	112	W8×21	W14×34	186	W14×30	W10×22
39	W10×68	W10×22	113	W10×22	W8×21	187	W10×22	W6×25
40	W8×21	W12×26	114	W10×49	W14×38	188	W14×605	W14×605
41	W8×35	W10×22	115	W10×33	W12×30	189	W16×36	W8×21
42	W14×74	W10×22	116	W8×31	W10×22	190	W8×24	W8×24
43	W8×24	W12×26	117	W10×22	W8×28	191	W14×38	W10×22
44	W14×120	W12×26	118	W8×21	W6×25	192	W8×21	W10×22
45	W8×24	W8×24	119	W8×28	W12×58	193	W10×22	W8×24
46	W10×39	W8×21	120	W14×30	W24×279	194	W8×21	W12×26
47	W16×36	W6×25	121	W12×26	W14×38	195	W8×21	W10×22
48	W8×21	W10×49	122	W10×49	W8×31	196	W8×21	W10×22
49	W8×21	W8×21	123	W8×21	W10×22	197	W8×28	W6×25
50	W12×40	W10×22	124	W18×86	W10×22	198	W8×21	W12×30
51	W14×34	W12×26	125	W33×118	W18×158	199	W8×21	W8×24
52	W12×26	W10×22	126	W8×21	W8×21	200	W10×22	W10×22
53	W8×21	W10×22	127	W10×22	W21×182	201	W12×26	W10×22
54	W8×21	W10×22	128	W12×26	W8×31	202	W8×21	W12×26
55	W8×21	W12×26	129	W10×22	W10×22	203	W8×21	W10×22
56	W8×21	W10×22	130	W8×24	W14×48	204	W6×25	W10×22
57	W8×21	W8×31	131	W8×21	W16×36	205	W10×22	W8×24
58	W8×24	W10×22	132	W8×21	W12×30	206	W8×21	W6×25
59	W14×34	W6×25	133	W8×21	W8×21	207	W10×22	W8×21
60	W10×22	W10×22	134	W8×21	W8×21	208	W8×21	W8×24
61	W16×36	W10×22	135	W8×21	W10×22	209	W8×21	W12×26
62	W8×35	W8×21	136	W12×26	W10×22	210	W8×21	W10×22
63	W33×318	W10×112	137	W10×22	W10×49	211	W8×21	W12×26
64	W12×136	W16×89	138	W10×22	W12×106	212	W6×25	W10×22
65	W21×147	W10×68	139	W8×21	W10×22	213	W8×21	W10×22
66	W18×86	W12×79	140	W10×22	W6×25	214	W8×21	W10×22
67	W10×88	W10×60	141	W8×21	W10×22	215	W8×21	W8×24
68	W14×82	W14×61	142	W8×21	W10×22	216	W8×21	W10×22
69	W12×152	W14×43	143	W8×21	W10×22	217	W12×26	W8×21

70	W10×49	W16×67	144	W8×21	W10×22	218	W8×31	W10×22
71	W10×60	W21×62	145	W14×30	W12×30	219	W12×58	W12×53
72	W12×136	W12×58	146	W10×22	W12×40	220	W14×99	W27×178
73	W16×89	W14×61	147	W8×21	W14×550	Best volume (in.^{3})	14,086,857	12,989,713
74	W14×90	W21×62	148	W8×21	W10×22			



3D view Top view Side view

Figure 5. Schematic of the spatial 2386-bar tower



1st story 2nd story 32nd story

Figure 6. Member groups of spatial 2386-bar tower

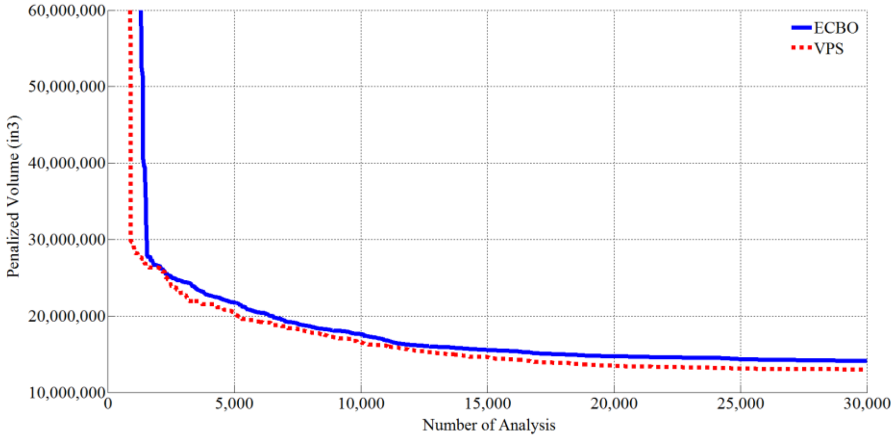


Figure 7. The convergence curves for the spatial 2386-bar tower

4. CONCLUSION

MATLAB code for the VPS algorithm is presented and two numerical examples chosen from size optimum design of truss towers are studied to test and verify the efficiency of the proposed method. Their results are compared with those of the ECBO algorithm. The VPS algorithm finds superior optimal designs for all the problems investigated, illustrating the capability of the present method in solving constrained problems. Besides, the average optimized results and standard deviation on averages results obtained by VPS are acceptable. It can be seen from convergence history diagrams that the convergence rate of the VPS algorithm is higher than that of the ECBO.

APPENDIX 1: VPS IN MATLAB

The VPS code in MATLAB:

```
% VIBRATING PARTICLES SYSTEM - VPS

% clear memory
clear all

% Initializing variables
popSize=20;                          % Size of the population
nVar=29;                             % Number of optimization variables
maxIt=200;                           % Maximum number of iteration
xMin=-500;                           % Lower bound of the variables
xMax=500;                            % Upper bound of the variables
alpha=0.05;                          % Parameter in Eq. (3)
w1=0.3;w2=0.3;w3=1-w1-w2;            % Parameters in Eq. (1)
p=0.2;                              % With the probability of (1-p) the effect
of BP is ignored in updating
PAR=0.1;HMCR=0.95;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]=FEM(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);

       % Updating particle positions
       D=(iter/maxIt)^(-alpha); % Eq. (3)
       for m=1:popSize
           temp1=m;
           temp2=m;
           while temp1==m
               temp1=ceil(rand*0.5*popSize);
           end
           while temp2==m
               temp2=popSize-ceil(rand*0.5*popSize)+1;
           end
           if p<rand
               w3=0;
               w2=1-w1;
           end
           for n=1:nVar
               A=(w1*(sortedHBV(1,2+n)-
position(m,n)))+(w2*(position(sortedAgentCost(temp1,2),n)-
position(m,n)))+(w3*(position(sortedAgentCost(temp2,2),n)-position(m,n)));    %
Eq. (4)
               comp1=(D*rand*A)+sortedHBV(1,2+n);
               comp2=(D*rand*A)+position(sortedAgentCost(temp1,2),n);
               comp3=(D*rand*A)+position(sortedAgentCost(temp2,2),n);
               position(m,n)=(w1*comp1)+(w2*comp2)+(w3*comp3); % Eq. (1)
           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

    end

    disp(sortedHBV(1,:))
```

REFERENCES

1. Degertekin SO. Optimum design of steel frames using harmony search algorithm, *Struct Multidisc Optim* 2008; **36**: 393-401.
2. Hasancebi O, Carbas S, Dogn E, Erdal F, Saka MP. Comparison of non-deterministic search techniques in the optimum design of real size steel frames, *Comput Struct* 2010; **88**: 1033-48.
3. Camp CV, Farshchin M. Design of space trusses using modified teaching - learning based optimization, *Eng Struct* 2014; **62–63**: 87–97.
4. Kaveh A, Ilchi Ghazaan M. A comparative study of CBO and ECBO for optimal design of skeletal structures, *Comput Struct* 2015; **153**: 137-47.
5. Gholizadeh S, Ebadijalal M. Seismic design optimization of steel structures by a sequential ECBO algorithm, *Int J Optim Civil Eng* 2017; **7(2)**: 157-71.
6. Rao RV, Savsani VJ, Vakharia DP. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, *Comput Aided Des* 2011; **43**(3): 303–15.
7. Eskandar H, Sadollah A, Bahreininejad A, Hamdi M. Water cycle algorithm-A novel metaheuristic optimization method for solving constrained engineering optimization problems, *Comput Struct* 2012; **110**: 151-66.
8. Kaveh A, Mahdavi VR. Colliding bodies optimization: A novel meta-heuristic method, *Comput Struct* 2014; **139**: 18-27.
9. Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer, *Adv Eng Softw* 2014; **69**: 46-61.
10. Mirjalili S. The ant lion optimizer, *Adv Eng Softw* 2015; **83**: 80-98.
11. Kaveh A, Zolghadr A. A novel metaheuristic algorithm: tug of war optimization, *Int J Optim Civil Eng* 2016; **6(4)**: 469-92.
12. Mirjalili S, Lewis A. The whale optimization algorithm, *Adv Eng Softw* 2016; **95**: 51-67.
13. Kaveh A, Bakhshpoori T. Water Evaporation Optimization: A novel physically inspired optimization algorithm, *Comput Struct* 2016; **167**: 69-85.

14. Kaveh A. *Advances in Metaheuristic Algorithms for Optimal Design of Structures*, 2nd edition, Springer, Switzerland, 2017.
15. Kaveh A. *Applications of Metaheuristic Optimization in Civil Engineering*, Springer, Switzerland, 2017.
16. Kaveh A, Ilchi Ghazaan M. Vibrating particles system algorithm for truss optimization with multiple natural frequency constraints, *Acta Mech* 2017; **228**: 307-22.
17. American Institute of Steel Construction (AISC), Manual of steel construction–allowable stress design 9th ed, Chicago, AISC, 1989.
18. Kaveh A, Ilchi Ghazaan M. Optimum design of large-scale truss towers using cascade optimization, *Acta Mech* 2016; **227**(9): 2645-56.