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)
A. Kaveh and M. Ilchi Ghazaan

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:

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

\[
w_1 + w_2 + w_3 = 1 \tag{2}
\]

\[
D = (\frac{\text{iter}}{\text{iter}_{\text{max}}})^{-\alpha} \tag{3}
\]

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

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. \( \text{iter} \) is the current iteration number and \( \text{iter}_{\text{max}} \) is the total number of iterations for optimization process. \( \alpha \) is a constant. \( \text{rand1} \), \( \text{rand2} \) and \( \text{rand3} \) 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 \( \text{rand} \) (a random numbers uniformly distributed in the range of \([0,1]\)) and if \( p < \text{rand} \), 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 \( \text{HMCR} \) (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 \( \text{PAR} \) (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 \( \text{P} < \text{rand} \)

\( w_1 = 0 \) and \( w_2 = 1 - w_1 \)

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 stored
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:

\[
\text{Find} \quad \{X\} = [x_1, x_2, ..., x_{nm}]
\]

\[\text{to minimize} \quad W(\{X\}) = \sum_{i=1}^{nm} \rho_i A_i L_i \]  

(5)
\[ \begin{align*}
\text{subjected to: } & \quad g_j(\{X\}) \leq 0, \quad j = 1,2,...,nc \\
& \quad x_{i_{\text{min}}} \leq x_i \leq x_{i_{\text{max}}} 
\end{align*} \]

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; \(\rho_i, A_i\) and \(L_i\) denote the material density, cross-sectional area, and the length of the \(i\)th member, respectively. \(x_{\text{min}}\) and \(x_{\text{max}}\) 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, \(\alpha\), \(p\), \(w_1\) 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](image1)
![Top view](image2)
![Side view](image3)

Figure 2. Schematic of the spatial 942-bar tower
MATLAB CODE FOR VIBRATING PARTICLES SYSTEM ALGORITHM

1st, 2nd and 3rd stories
13th, 14th and 15th stories
21st, 22nd and 23rd stories
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>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>W12×190</td>
<td>W12×170</td>
<td>27</td>
<td>W10×33</td>
<td>W8×24</td>
<td>53</td>
<td>W6×25</td>
<td>W8×24</td>
<td>54</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>2</td>
<td>W36×230</td>
<td>W36×260</td>
<td>28</td>
<td>W8×31</td>
<td>W12×26</td>
<td>55</td>
<td>W8×21</td>
<td>W10×22</td>
<td>56</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>3</td>
<td>W40×199</td>
<td>W44×262</td>
<td>29</td>
<td>W8×31</td>
<td>W10×22</td>
<td>57</td>
<td>W8×21</td>
<td>W8×21</td>
<td>58</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>4</td>
<td>W24×229</td>
<td>W30×235</td>
<td>30</td>
<td>W8×21</td>
<td>W8×21</td>
<td>59</td>
<td>W8×21</td>
<td>W10×22</td>
<td>60</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>5</td>
<td>W36×150</td>
<td>W36×245</td>
<td>31</td>
<td>W8×21</td>
<td>W8×21</td>
<td>61</td>
<td>W8×21</td>
<td>W14×20</td>
<td>62</td>
<td>W12×65</td>
<td>W14×38</td>
</tr>
<tr>
<td>6</td>
<td>W30×173</td>
<td>W24×229</td>
<td>32</td>
<td>W12×26</td>
<td>W10×22</td>
<td>63</td>
<td>W8×21</td>
<td>W10×22</td>
<td>64</td>
<td>W8×21</td>
<td>W14×61</td>
</tr>
<tr>
<td>7</td>
<td>W24×250</td>
<td>W40×199</td>
<td>33</td>
<td>W8×21</td>
<td>W8×21</td>
<td>65</td>
<td>W8×21</td>
<td>W10×22</td>
<td>66</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>8</td>
<td>W27×258</td>
<td>W14×193</td>
<td>34</td>
<td>W8×21</td>
<td>W10×22</td>
<td>67</td>
<td>W8×21</td>
<td>W8×21</td>
<td>68</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>9</td>
<td>W14×159</td>
<td>W40×174</td>
<td>35</td>
<td>W8×21</td>
<td>W8×21</td>
<td>69</td>
<td>W8×21</td>
<td>W10×22</td>
<td>70</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>10</td>
<td>W30×191</td>
<td>W24×162</td>
<td>36</td>
<td>W8×31</td>
<td>W12×58</td>
<td>71</td>
<td>W27×94</td>
<td>W12×58</td>
<td>72</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>11</td>
<td>W18×158</td>
<td>W14×145</td>
<td>37</td>
<td>W30×191</td>
<td>W30×211</td>
<td>73</td>
<td>W8×21</td>
<td>W10×22</td>
<td>74</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>12</td>
<td>W18×119</td>
<td>W18×119</td>
<td>38</td>
<td>W30×116</td>
<td>W14×109</td>
<td>75</td>
<td>W8×21</td>
<td>W10×22</td>
<td>76</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>13</td>
<td>W24×250</td>
<td>W12×279</td>
<td>39</td>
<td>W27×178</td>
<td>W24×131</td>
<td>77</td>
<td>W8×21</td>
<td>W10×22</td>
<td>78</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>14</td>
<td>W14×30</td>
<td>W8×21</td>
<td>40</td>
<td>W24×131</td>
<td>W21×101</td>
<td>79</td>
<td>W8×21</td>
<td>W10×22</td>
<td>80</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>15</td>
<td>W8×21</td>
<td>W10×22</td>
<td>41</td>
<td>W8×31</td>
<td>W12×58</td>
<td>81</td>
<td>W8×21</td>
<td>W10×22</td>
<td>82</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>16</td>
<td>W8×21</td>
<td>W12×26</td>
<td>42</td>
<td>W10×88</td>
<td>W10×77</td>
<td>83</td>
<td>W8×21</td>
<td>W10×22</td>
<td>84</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>17</td>
<td>W8×21</td>
<td>W10×22</td>
<td>43</td>
<td>W21×62</td>
<td>W12×50</td>
<td>85</td>
<td>W8×21</td>
<td>W10×22</td>
<td>86</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>18</td>
<td>W8×21</td>
<td>W10×22</td>
<td>44</td>
<td>W12×136</td>
<td>W27×114</td>
<td>87</td>
<td>W8×21</td>
<td>W10×22</td>
<td>88</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>19</td>
<td>W8×21</td>
<td>W10×22</td>
<td>45</td>
<td>W8×21</td>
<td>W10×22</td>
<td>89</td>
<td>W8×21</td>
<td>W10×22</td>
<td>90</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>20</td>
<td>W8×21</td>
<td>W10×22</td>
<td>46</td>
<td>W8×21</td>
<td>W10×22</td>
<td>91</td>
<td>W8×21</td>
<td>W10×22</td>
<td>92</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>21</td>
<td>W8×21</td>
<td>W6×25</td>
<td>47</td>
<td>W8×21</td>
<td>W10×22</td>
<td>93</td>
<td>W8×21</td>
<td>W10×22</td>
<td>94</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>22</td>
<td>W8×21</td>
<td>W8×24</td>
<td>48</td>
<td>W8×21</td>
<td>W6×25</td>
<td>95</td>
<td>W8×21</td>
<td>W10×22</td>
<td>96</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>23</td>
<td>W8×21</td>
<td>W10×22</td>
<td>49</td>
<td>W8×21</td>
<td>W10×22</td>
<td>97</td>
<td>W8×21</td>
<td>W10×22</td>
<td>98</td>
<td>W8×21</td>
<td>W10×22</td>
</tr>
<tr>
<td>24</td>
<td>W24×117</td>
<td>W14×145</td>
<td>50</td>
<td>W8×21</td>
<td>W8×40</td>
<td>99</td>
<td>W8×21</td>
<td>W8×28</td>
<td>100</td>
<td>W8×21</td>
<td>W8×28</td>
</tr>
<tr>
<td>25</td>
<td>W12×50</td>
<td>W8×31</td>
<td>51</td>
<td>W27×94</td>
<td>W12×58</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>W14×30</td>
<td>W8×24</td>
<td>52</td>
<td>W10×22</td>
<td>W6×25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Comparison of optimized designs obtained for the spatial 942-bar tower problem
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>
<thead>
<tr>
<th>No.</th>
<th>Sections</th>
<th>No.</th>
<th>Sections</th>
<th>No.</th>
<th>Sections</th>
<th>No.</th>
<th>Sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>W14×730</td>
<td>75</td>
<td>W14×38</td>
<td>149</td>
<td>W8×21</td>
<td>6</td>
<td>W6×25</td>
</tr>
<tr>
<td>2</td>
<td>W14×730</td>
<td>76</td>
<td>W12×65</td>
<td>150</td>
<td>W14×34</td>
<td>10</td>
<td>W10×22</td>
</tr>
<tr>
<td>3</td>
<td>W14×730</td>
<td>77</td>
<td>W14×90</td>
<td>151</td>
<td>W10×22</td>
<td>10</td>
<td>W10×22</td>
</tr>
<tr>
<td>4</td>
<td>W14×665</td>
<td>78</td>
<td>W12×65</td>
<td>152</td>
<td>W12×30</td>
<td>8</td>
<td>W8×24</td>
</tr>
<tr>
<td>5</td>
<td>W14×730</td>
<td>79</td>
<td>W30×116</td>
<td>153</td>
<td>W8×21</td>
<td>12</td>
<td>W12×26</td>
</tr>
<tr>
<td>6</td>
<td>W14×730</td>
<td>80</td>
<td>W14×90</td>
<td>154</td>
<td>W10×22</td>
<td>8</td>
<td>W8×28</td>
</tr>
<tr>
<td>7</td>
<td>W14×730</td>
<td>81</td>
<td>W18×76</td>
<td>155</td>
<td>W8×24</td>
<td>8</td>
<td>W8×31</td>
</tr>
<tr>
<td>8</td>
<td>W40×215</td>
<td>82</td>
<td>W14×48</td>
<td>156</td>
<td>W27×146</td>
<td>12</td>
<td>W12×79</td>
</tr>
<tr>
<td>9</td>
<td>W14×665</td>
<td>83</td>
<td>W10×68</td>
<td>157</td>
<td>W14×48</td>
<td>10</td>
<td>W10×22</td>
</tr>
<tr>
<td>10</td>
<td>W14×500</td>
<td>84</td>
<td>W8×28</td>
<td>158</td>
<td>W8×21</td>
<td>10</td>
<td>W10×22</td>
</tr>
<tr>
<td>11</td>
<td>W12×279</td>
<td>85</td>
<td>W10×60</td>
<td>159</td>
<td>W14×34</td>
<td>8</td>
<td>W8×24</td>
</tr>
<tr>
<td>12</td>
<td>W33×318</td>
<td>86</td>
<td>W14×38</td>
<td>160</td>
<td>W8×21</td>
<td>10</td>
<td>W10×45</td>
</tr>
<tr>
<td>13</td>
<td>W14×605</td>
<td>87</td>
<td>W10×45</td>
<td>161</td>
<td>W10×22</td>
<td>33</td>
<td>W3×201</td>
</tr>
<tr>
<td>14</td>
<td>W14×730</td>
<td>88</td>
<td>W12×50</td>
<td>162</td>
<td>W6×25</td>
<td>14</td>
<td>W14×34</td>
</tr>
<tr>
<td>15</td>
<td>W14×455</td>
<td>89</td>
<td>W14×82</td>
<td>163</td>
<td>W8×21</td>
<td>12</td>
<td>W12×65</td>
</tr>
<tr>
<td>16</td>
<td>W33×221</td>
<td>90</td>
<td>W8×40</td>
<td>164</td>
<td>W8×24</td>
<td>12</td>
<td>W12×30</td>
</tr>
<tr>
<td>17</td>
<td>W44×335</td>
<td>91</td>
<td>W10×22</td>
<td>165</td>
<td>W10×22</td>
<td>10</td>
<td>W10×22</td>
</tr>
</tbody>
</table>
Figure 5. Schematic of the spatial 2386-bar tower

3D view  Top view  Side view

Figure 6. Member groups of 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:

```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
    end

    w2=0.3;w3=1-w1-w2;
end

% Handling the side constraints
for m=1:popSize
MATLAB CODE FOR VIBRATING PARTICLES SYSTEM ALGORITHM

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

REFERENCES


