ASIAN JOURNAL OF CIVIL ENGINEERING (BUILDING AND HOUSING) VOL. 11, NO. 3 (2010)
PAGES 277-293

A CHARGED SYSTEM SEARCH WITH A FLY TO BOUNDARY METHOD FOR DISCRETE OPTIMUM DESIGN OF TRUSS STRUCTURES

A. Kaveh[*a] and S. Talatahari[b]
[a]Centre of Excellence for Fundamental Studies in Structural Engineering, Iran University of Science and Technology, Tehran, Iran
[b]Department of Civil Engineering, University of Tabriz, Tabriz, Iran

ABSTRACT

A discrete version of the Charged System Search algorithm (CSS) is developed to optimize truss structures with discrete variables. The discrete CSS algorithm, similar to its original version, is based on some laws from electrostatics and the Newtonian mechanics. Each agent in the CSS is considered as a charged sphere having a uniform volume charge density which can affect an electric force to the other ones. However, contrary to the original CSS, for the discrete CSS, the affected forces can be attractive or repulsive. In addition, a new approach is presented to handle the constraints, which is called the *fly to boundary* method. Some design examples are tested using the new method and the results are compared to those of other meta-heuristic algorithms to demonstrate the effectiveness of the present method.

Keywords: Charged system search; meta-heuristic algorithms; fly to boundary method; optimum discrete design; truss structures

1. INTRODUCTION

The competitive world has forced engineers and researchers to evince interest in economical designs and to develop optimization approaches. Recently, authors have presented a novel continuous optimization algorithm, so called Charged System Search (CSS) [1] and it is applied to skeletal structural optimization problems [2]. The CSS algorithm is based on the Coulomb and Gauss laws from electrical physics and the governing laws of motion from the Newtonian mechanics. This algorithm can be considered as a multi-agent approach, where each agent is a Charged Particle (CP). Each CP is considered as a charged sphere with a specified radius, having a uniform volume charge density which can insert an electric force to the other CPs.

For practical structural optimization problems, industrial cross sections are used which

[*] E-mail address of the corresponding author: alikaveh@iust.ac.ir (A. Kaveh)

have discrete values and as a result a discrete solution is preferred to the continuous one for this kind of the optimization problems [3]. When the design variables represent a selection from a set of parts, the problem is considered as a discrete one [4]. Many research results are available on optimum discrete design of structural problems utilizing different meta-heuristic algorithms [2-18]. In our previous research, a new optimization method has been introduced for structural optimization problems with continuous variables using the CSS algorithm [1,2]. The main contribution of the present paper is to present a discrete version of the CSS which not only preserves the positive characteristics of the original CSS, but also improves its capability in solving discrete problems. The simplest method to have a discrete result is to utilize a rounding function where in this method the agents are allowed to select only discrete values from the permissible list of cross sections. In other words, if any of the agents selects another value for a design variable, the algorithm changes its magnitude by the value of the nearest discrete cross section. However, utilizing this method reduces the exploration of the algorithm. In order to obviate this defect, contrary to the original CSS, both attractive and repulsive forces are considered. In this way, the power of the exploration of the algorithm is increased.

In order to handle the constraints, we have utilized a simple penalty approach in our previous researches [1,2]. Despite the popularity of penalty functions, there are several drawbacks associated with this method, and the main one is the requirement of a careful tuning of the penalty factors for accurate estimation of the degree of penalization to be applied in order to approach efficiently to the feasible region [19]. In order to deal with this problem, a new constraint handling approach is developed here which is called the fly to boundary method. This approach can be considered as an improved version of the fly-back and feasible-based approaches.

2. CHARGED SYSTEM SEARCH FOR TRUSS STRUCTURES WITH DISCRETE VARIABLES

2.1 Review of the continuous CSS algorithm

The Charged System Search contains a number of Charged Particle (CP) where each one treated as a charged sphere and can insert an electric force to the others. The pseudo-code for the CSS algorithm is summarized as follows [1]:

- **Step 1: Initialization.** The magnitude of charge for each CP is defined as

$$q_i = \frac{fit(i) - fitworst}{fitbest - fitworst} \,!;!\, i = 1,2,...,N \tag{1}$$

where *fitbest* and fitworst are the best and the worst fitness of all the particles; *fit*(*i*) represents the fitness of the agent *i*; and *N* is the total number of CPs. The separation distance r_{ij} between two charged particles is defined as follows:

$$r_{ij} = \frac{\| \mathbf{X}_i - \mathbf{X}_j \|}{\| (\mathbf{X}_i + \mathbf{X}_j)/2 - \mathbf{X}_{best} \| + \varepsilon} \tag{2}$$

where $\mathbf{X}_i$ and $\mathbf{X}_j$ are the positions of the ith and jth CPs, respectively, $\mathbf{X}_{best}$ is the position of the best current CP, and ε is a small positive number. The initial positions of CPs are determined randomly.

- **Step 2: CM creation.** A number of the best CPs and the values of their corresponding fitness functions are saved in the Charged Memory (CM).
- **Step 3: The probability of moving determination.** The probability of moving each CP toward the others is determined using the following function:

$$p_{ij} = \begin{cases} 1 & \dfrac{fit(i) - fitbest}{fit(j) - fit(i)} > rand \vee fit(j) > fit(i) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

- **Step 4: Forces determination.** The resultant force vector for each CP is calculated as

$$\mathbf{F}_j = q_j \sum_{i,i \neq j} \left(\frac{q_i}{a^3} r_{ij} \cdot i_1 + \frac{q_i}{r_{ij}^2} \cdot i_2 \right) p_{ij} (\mathbf{X}_i - \mathbf{X}_j) \; !;!! \begin{cases} j = 1,2,...,N \\ i_1 = 1, i_2 = 0 \Leftrightarrow r_{ij} < a \\ i_1 = 0, i_2 = 1 \Leftrightarrow r_{ij} \geq a \end{cases} \tag{4}$$

- **Step 5: Solution construction.** Each CP moves to the new position as

$$\mathbf{X}_{j,new} = rand_{j1} \cdot k_a \cdot \frac{\mathbf{F}_j}{m_j} \cdot \Delta t^2 + rand_{j2} \cdot k_v \cdot \mathbf{V}_{j,old} \cdot \Delta t + \mathbf{X}_{j,old} \tag{5}$$

$$\mathbf{V}_{j,new} = \frac{\mathbf{X}_{j,new} - \mathbf{X}_{j,old}}{\Delta t} \tag{6}$$

where k_a and k_v are the acceleration and the velocity coefficients, respectively; and $rand_{j1}$ and $rand_{j2}$ are two random numbers uniformly distributed in the range (0,1).

- **Step 6: CP position correction.** If each CP swerves off the predefined bounds, correct its position using the harmony search-based handling approach as described in Ref. [15].
- **Step 7: CM updating.** The better new vectors are Included to the CM and the worst ones are excluded from the CM.
- **Step 8: Terminating criterion contro**l. Steps 3-7 are repeated until a terminating criterion is satisfied.

2.2 A discrete CSS

One way to solve discrete problems by using a continuous algorithm is to utilize a rounding function which changes the magnitude of a result to the nearest discrete value, as

$$\mathbf{X}_{j,new} = Fix\left(rand_{j1} \cdot k_a \cdot \frac{\mathbf{F}_j}{m_j} \cdot \Delta t^2 + rand_{j2} \cdot k_v \cdot \mathbf{V}_{j,old} \cdot \Delta t + \mathbf{X}_{j,old} \right) \tag{7}$$

where $Fix(\mathbf{X})$ is a function which rounds each elements of $\mathbf{X}$ to the nearest permissible discrete value. Using this position updating formula, the agents will be permitted to select discrete values. Although this change is simple and efficient, it may reduce the exploration of the algorithm [4]. Therefore, in order to maintain the exploration rate, here we perform two changes. Firstly, a new parameter so-called the kind of force is defined as

$$ar_{ij} = \begin{cases} +1 & \text{w.p. } k_t \\ -1 & \text{w.p. } 1-k_t \end{cases} \tag{8}$$

where ar_{ij} determines the type of the force, where +1 represents for the attractive force and −1 denotes for the repelling force and k_t is a parameter to control the effect of the kind of force. In general the attractive force collects the agents in a part of search space and the repelling force strives to disperse the agents. As a result, utilizing this new parameter the resultant force is redefined as

$$\mathbf{F}_j = q_j \sum_{i,i\neq j} \left(\frac{q_i}{a^3} r_{ij} \cdot i_1 + \frac{q_i}{r_{ij}^2} \cdot i_2 \right) ar_{ij}\, p_{ij} (\mathbf{X}_i - \mathbf{X}_j) ;!! \begin{cases} j = 1,2,...,N \\ i_1 = 1, i_2 = 0 \Leftrightarrow r_{ij} < a \\ i_1 = 0, i_2 = 1 \Leftrightarrow r_{ij} \geq a \end{cases} \tag{9}$$

The second change consists of assigning a big value for k_v instead of a small one. The effect of the pervious velocity of a CP is controlled based on the values of the k_v. Performing more searches in the early iterations may improve the exploration ability, however it must be deceased gradually. k_a can be considered as an exploitation controller parameter and k_v, controls the exploration process, and therefore a decreasing function will be a good selection as considered in Refs. [1,2]. However, here in order to maintain the exploration rate in a discrete search domain, it is selected as a big value (equal to 2). Figure 1 shows the flowchart of the discrete CSS algorithm.

2.3 Further explanation of discrete CSS

In order to further clarify the process of the discrete CSS, we consider the following Rosenbrock function [20], as one of the standard test functions in optimization,

$$f(\mathbf{X}) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2 \tag{10}$$

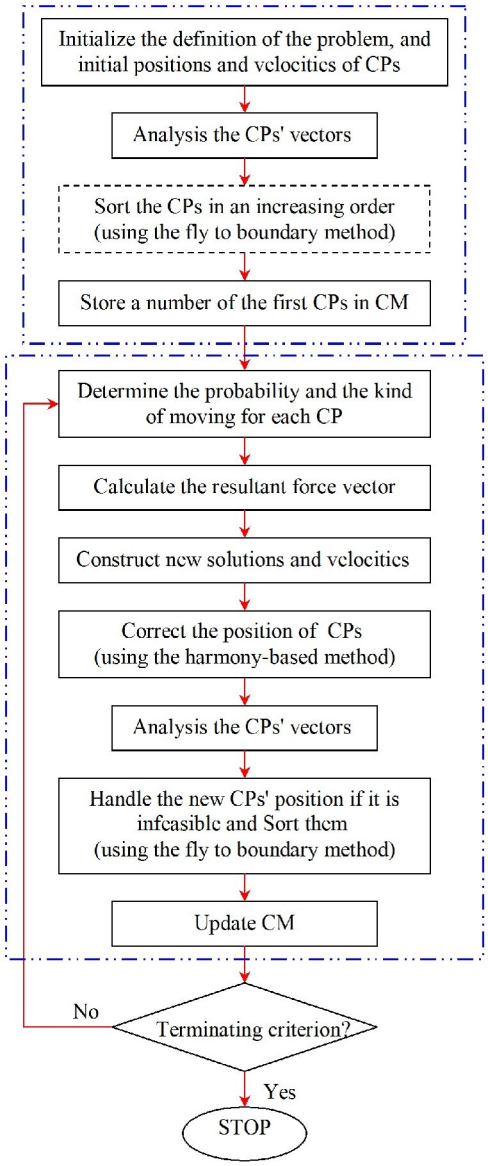

Figure 1. The Flowchart for the discrete CSS algorithm

Owing to a long narrow and curved valley that is present in this function, gradient-based algorithms may require a large number of iterations before the minimum vector (1.0, 1.0) is found, as shown in Figure 2. The permitted domain for the discrete CSS algorithm is taken from the set D = {- 3.00,- 2.95,- 2.90,- 2.85,- 2.80, . . . , 2.80, 2.85, 2.90, 2.95, 3.00}, which has 121 discrete values. For this example, the number of CPs is set to 25. Figure 3 shows the positions of the current CPs and the stored CPs in the CM for this benchmark example for different number of iterations. It can be seen that in the early iterations, the CPs

investigate the entire search space to discover a favorite space (global search). When this favorite space containing a global optimum is discovered, the movements of the CPs are limited in this space to provide more exploitation (local search).

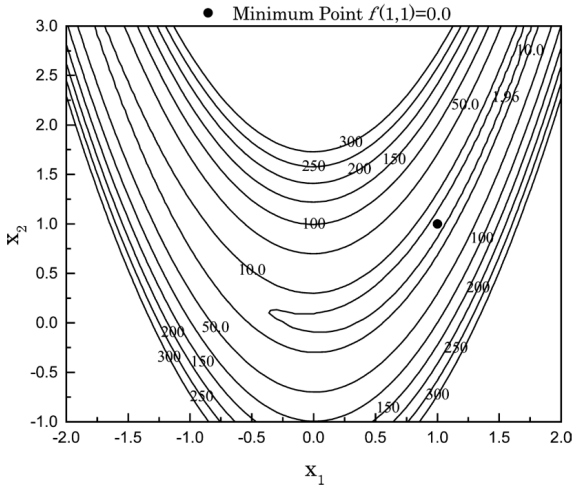

Figure 2. Rosenbrock function

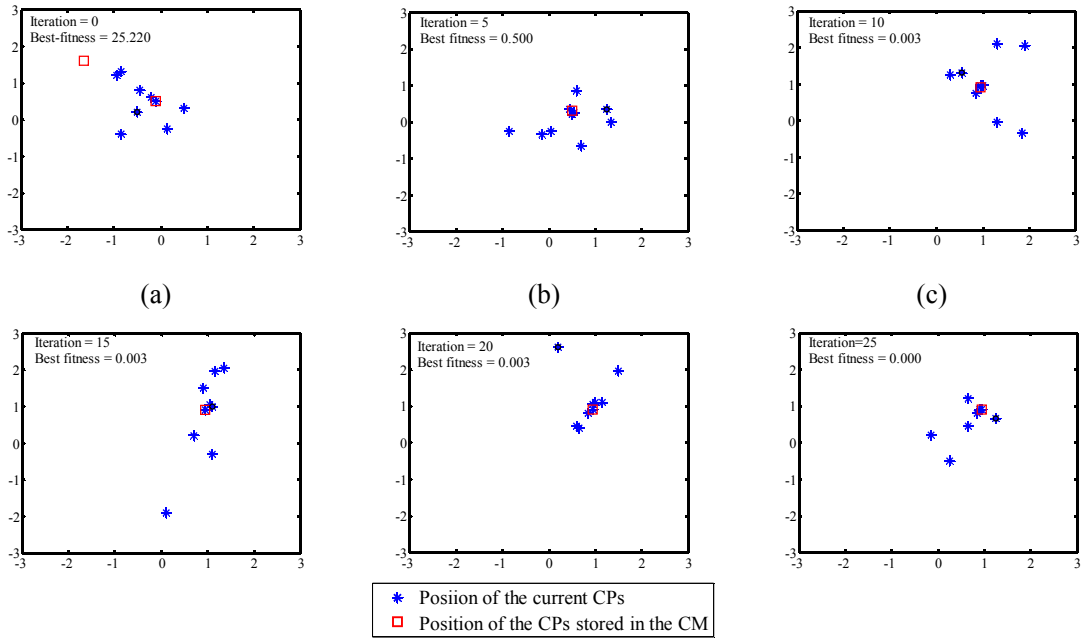

Figure 3. The positions of the current CPs and the stored CPs in the CM for the mathematical example

3. FLY TO BOUNDARY METHOD

The feasible-based constrained approach [21] deals with constrained search spaces by using the separation of constraints and objectives. In this method, the idea is to avoid the combination of the value of the objective function and the constraints of a problem to assign fitness, like when using a penalty function [19,21]. Authors have recently presented a modified feasible-based method which employs the following four rules as [22]

Rule 1: Any feasible solution is preferred to any infeasible solution.
Rule 2: Infeasible solutions with slight violations of the constraints are treated as feasible ones.
Rule 3: Between two feasible solutions, the one with better objective function value is preferred.
Rule 4: Between two infeasible solutions, the one having smaller sum of constraint violations is preferred.

Although the feasible-based mechanism is a sufficiently powerful and reliable approach, it has some disadvantages. Assume that in the *k*th iteration all the solutions in the memory of the optimization algorithm have feasible values. If a new solution generated in the subsequent iterations is infeasible, according to the rule 1, it is not employed. This solution is compared to those stored in the memory only and only if it is feasible, according to the rules 2 and 3. The last rule after that iteration (iteration *k* in which all solutions in the memory are feasible) is not used at all. This process is completely similar to the fly-back mechanism. In the fly-back method, any generated solution is compared to the memorized one, if it is feasible (similar to rule two in the feasible-based method) while it is ignored if it is not feasible (similar to rule one in the feasible-based method). However, the fly-back approach has some difficulties in finding the first valid solutions for all agents, since according to this method the agents must be initialized in the allowable region.

Here considering these points, a new constraint handling approach is developed. This method is called the fly to boundary approach and is based on fly-back and feasible-based methods. For most of the structural optimum design problems, the global minimum is located on or close to the boundary of the feasible design space. In other words, the constraints in the engineering problems determine the limits of the search space and often at least one constraint is active for the final optimum result. Therefore, the fly to boundary method attempts to lead the infeasible solutions toward the boundary space while with the feasible solutions it behaves like the feasible-based method. Related to the drawback of defining the initial feasible vectors for all agents, the selection is limited to some strong sections in the initialization stage for this method. Two rules are considered in the fly to boundary method as follows:

Rule 1: When the new solution is feasible, between the new and stored solutions, the one having better weight value is preferred.
Rule 2: If the new solution is infeasible, then one of the following sub-rules will be used:

2.1. When a solution has slight violations of the constraints, it will be treated as feasible one. However, the permitted violations will be decreased when the number of iterations increases. This efficiently guarantees that the optimization algorithm does not swerve from feasible space.

2.2. When the violations are big, for a predefined time, the following process will be repeated:

$$\begin{cases} \mathbf{X}_{j,old} = \dfrac{\mathbf{X}_{j,old} + \mathbf{X}_{j,new}}{2}, \mathbf{X}_{j,new} = \mathbf{X}_{j,new} \text{ if } \dfrac{\mathbf{X}_{j,old} + \mathbf{X}_{j,new}}{2} \in \text{feasible space} \\ \mathbf{X}_{j,old} = \mathbf{X}_{j,old}, \mathbf{X}_{j,new} = \dfrac{\mathbf{X}_{j,old} + \mathbf{X}_{j,new}}{2} \text{ if } \dfrac{\mathbf{X}_{j,old} + \mathbf{X}_{j,new}}{2} \notin \text{feasible space} \end{cases} \tag{11}$$

At last, the $\mathbf{X}_{j,old}$ is selected as the new solution.

When the new solution is infeasible and the previous one is feasible, the boundary of at least one constraint will be between these points. We want to find a point which is feasible while being very close to this boundary. Firstly, the centre of the new solution and the previous one is determined. If the new solution is feasible, the boundary will be between this point and $\mathbf{X}_{j,new}$ and thus $\mathbf{X}_{j,old}$ must be changed to this new point; and if the new point is infeasible, $\mathbf{X}_{j,new}$ will be changed to this point. This process is similar to the bisection method which is used for root finding of functions. We expect in both conditions the new point to be nearer to the boundary than the previous one. If this process is repeated for a definite number of times, the final result will not only be feasible, but will be on or very close to the boundary.

4. FORMULATION FOR DISCRETE OPTIMUM DESIGN OF STRUCTURES

Minimizing the structural weight *W* requires the selection of the optimum values of member cross-section d_i while satisfying the design constraints. The discrete optimal design problem of truss structures may be expressed as

$$\text{Find} \quad \mathbf{X} = [x_1, x_2, ..., x_{ng}]$$

$$\text{to minimize} \quad W(\mathbf{X}) = \sum_{i=1}^{nm} \gamma_i \cdot x_i \cdot L_i \tag{12}$$

subject to

$$x_i \in D_i, D_i = \{d_{i,1}, d_{i,2}, ..., d_{i,r(i)}\}$$
$$\delta_{\min} \le \delta_i \le \delta_{\max} \qquad i = 1,2,...., m$$
$$\sigma_{\min} \le \sigma_i \le \sigma_{\max} \qquad i = 1,2,...., nm$$

where **X** is a vector containing the design variables; D_i is an allowable set of discrete values for the design variable x_i; *ng* is the number of design variables or the number of member groups; $r(i)$ is the number of available discrete values for the *i*th design variable; $W(\mathbf{X})$ is the cost function which is taken as the weight of the structure; *nm* is the number of members forming the structure; *m* is the number of nodes; γ_i is the material density of member *i*; L_i is the length of the member *i*; σ_i and δ_i are the stress and nodal deflection, respectively; *min* and *max* mean the lower and upper bounds, respectively.

5. NUMERICAL EXAMPLES

Three truss examples with discrete variables are optimized utilizing the new method. Then, the results are compared to the solutions of other methods to demonstrate the efficiency of the present approach. For the proposed algorithm, a population of 25 individuals is used in all the examples; the constants *a* and k_t are set to 1 and 0.8, respectively. The acceleration coefficient k_a and the velocity coefficient k_v are considered as 1 and 2, respectively. The algorithms are coded in Matlab and the structures are analyzed using the direct stiffness method.

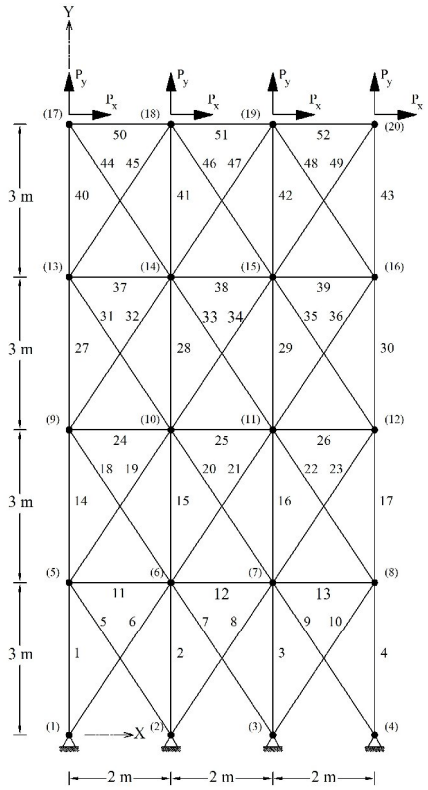

Figure 4. A 52-bar planer truss

5.1 A 52-bar planer truss

The 52-bar planar truss structure shown in Figure 4 has been analyzed by Wu and Chow [6], Lee and Geem [9], Li et al. [14] and Kaveh and Talatahari [4]. The members of this structure are divided into 12 groups: (1) A_1–A_4, (2) A_5–A_{10}, (3) A_{11}–A_{13}, (4) A_{14}–A_{17}, (5) A_{18}–A_{23}, (6) *A24–A26*, (7) *A27–A30*, (8) *A31–A36*, (9) *A37–A39*, (10) A_{40}–A_{43}, (11) A_{44}–A_{49}, and (12) A_{50}–A_{52}. The material density is 7860.0 kg/m^3 and the modulus of elasticity is 2.07×10^5 MPa. The members are subjected to stress limitations of ± 180 MPa. Both of the loads, *Px* = 100 kN and *Py* = 200 kN are considered. The discrete variables are selected from Table 1.

Table 1: The available cross-section areas of the AISC code

No.	in.^{2}	mm^2	No.	in.^{2}	mm^2
1	0.111	(71.613)	33	3.840	(2477.414)
2	0.141	(90.968)	34	3.870	(2496.769)
3	0.196	(126.451)	35	3.880	(2503.221)
4	0.250	(161.290)	36	4.180	(2696.769)
5	0.307	(198.064)	37	4.220	(2722.575)
6	0.391	(252.258)	38	4.490	(2896.768)
7	0.442	(285.161)	39	4.590	(2961.284)
8	0.563	(363.225)	40	4.800	(3096.768)
9	0.602	(388.386)	41	4.970	(3206.445)
10	0.766	(494.193)	42	5.120	(3303.219)
11	0.785	(506.451)	43	5.740	(3703.218)
12	0.994	(641.289)	44	7.220	(4658.055)
13	1.000	(645.160)	45	7.970	(5141.925)
14	1.228	(792.256)	46	8.530	(5503.215)
15	1.266	(816.773)	47	9.300	(5999.988)
16	1.457	(939.998)	48	10.850	(6999.986)
17	1.563	(1008.385)	49	11.500	(7419.430)
18	1.620	(1045.159)	50	13.500	(8709.660)
19	1.800	(1161.288)	51	13.900	(8967.724)
20	1.990	(1283.868)	52	14.200	(9161.272)
21	2.130	(1374.191)	53	15.500	(9999.980)
22	2.380	(1535.481)	54	16.000	(10322.560)
23	2.620	(1690.319)	55	16.900	(10903.204)
24	2.630	(1696.771)	56	18.800	(12129.008)
25	2.880	(1858.061)	57	19.900	(12838.684)
26	2.930	(1890.319)	58	22.000	(14193.520)
27	3.090	(1993.544)	59	22.900	(14774.164)
28	1.130	(729.031)	60	24.500	(15806.420)
29	3.380	(2180.641)	61	26.500	(17096.740)
30	3.470	(2238.705)	62	28.000	(18064.480)
31	3.550	(2290.318)	63	30.000	(19354.800)
32	3.630	(2341.931)	64	33.500	(21612.860)

Table 2 and Figure 5 provide the comparison of optimal design results and convergence rates of the 52-bar planar truss structure, respectively. From Table 2, it can be observed that the PSOPC can not find a good result while the CSS can find the best result. The HPSO and HPSACO algorithms achieve good optimal results after 100,000 and 5,300 analyses iterations, respectively. However, CSS needs less than 5,000 analyses to reach to a good solution.

Table 2: Optimal design comparison for the 52-bar spatial truss

Optimal cross-sectional areas (cm^2)							
Element group		**Wu and Chow [6]**	**Lee and Geem [9]**	**Li *et al.* [14]**		**Kaveh and Talatahari [4]**	**Present study**
		GA	**HS**	**PSOPC**	**HPSO**	**HPSACO**	**CSS**
1	$A_1 \sim A_4$	4658.055	4658.055	5999.988	4658.055	4658.055	4658.055
2	$A_5 \sim A_{10}$	1161.288	1161.288	1008.380	1161.288	1161.288	1161.288
3	$A_{11} \sim A_{13}$	645.160	506.451	2696.380	363.225	494.193	388.386
4	$A_{14} \sim A_{17}$	3303.219	3303.219	3206.440	3303.219	3303.219	3303.219
5	$A_{18} \sim A_{23}$	1045.159	940.000	1161.290	940.000	1008.385	940.000
6	$A_{24} \sim A_{26}$	494.193	494.193	729.030	494.193	285.161	494.193
7	$A_{27} \sim A_{30}$	2477.414	2290.318	2238.710	2238.705	2290.318	2238.705
8	$A_{31} \sim A_{36}$	1045.159	1008.385	1008.380	1008.385	1008.385	1008.385
9	$A_{3\,7} \sim A_{39}$	285.161	2290.318	494.190	388.386	388.386	494.193
10	$A_{40} \sim A_{43}$	1696.771	1535.481	1283.870	1283.868	1283.868	1283.868
11	$A_{44} \sim A_{49}$	1045.159	1045.159	1161.290	1161.288	1161.288	1161.288
12	$A_{50} \sim A_{52}$	641.289	506.451	494.190	792.256	506.451	494.193
Weight (*kg*)		1970.142	1906.76	2146.63	1905.49	1904.83	1897.62

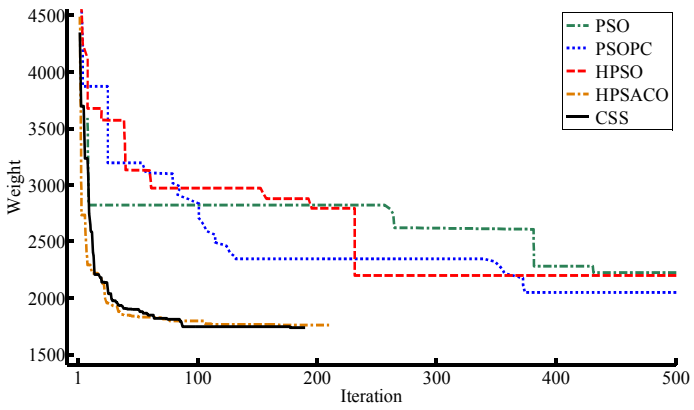

Figure 5. The convergence history for the 52-bar truss structure

5.2 A 72-bar spatial truss

For the 72-bar spatial truss structure shown in Figure 6, the material density is 0.1 lb/in^3 (2767.990 kg/m^3) and the modulus of elasticity is 10,000 ksi (68,950 MPa). The members are subjected to the stress limits of ±25 ksi (±172.375 MPa). The nodes are subjected to the displacement limits of ±0.25 in (±0.635 cm). The 72 structural members of this spatial truss are categorized as 16 groups using symmetry: (1) A_1–A_4, (2) A_5–A_{12}, (3) A_{13}–A_{16}, (4) A_{17}–A_{18}, (5) A_{19}–A_{22}, (6) A_{23}–A_{30}, (7) A_{31}–A_{34}, (8) A_{35}–A_{36}, (9) A_{37}–A_{40}, (10) A_{41}–A_{48}, (11) A_{49}–A_{52}, (12) A_{53}–A_{54}, (13) A_{55}–A_{58}, (14) A_{59}–A_{66} (15), A_{67}–A_{70}, and (16) A_{71}–A_{72}. The discrete variables are selected from Table 1. Table 3 lists the values and directions of the two load cases applied to the 72-bar spatial truss.

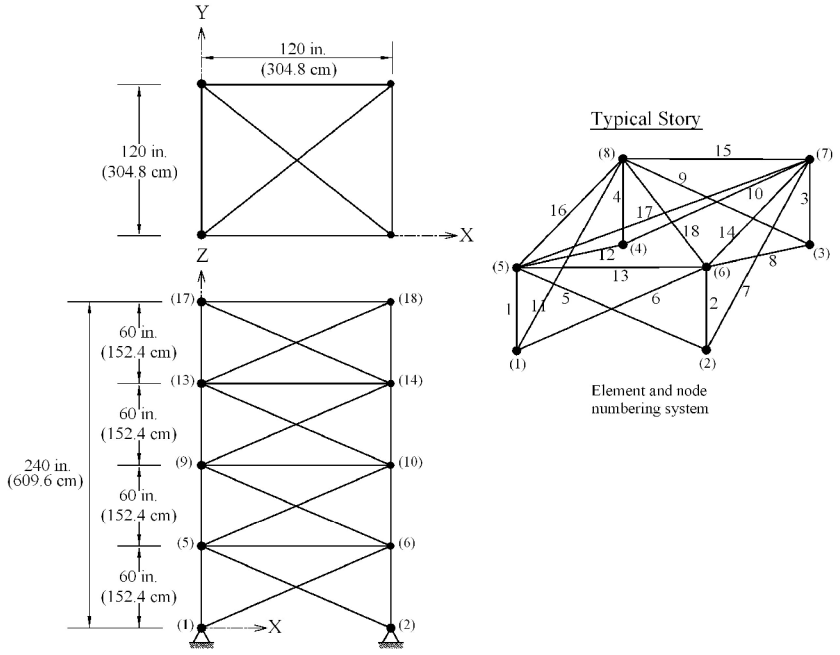

Figure 6. A 72-bar spatial truss

Table 3: Loading conditions for the 72-bar spatial truss

	Case 1			Case 2		
Node	P_X ***kips(kN)***	P_Y ***kips(kN)***	P_Z ***kips(kN)***	P_X	P_Y	P_Z ***kips(kN)***
17	5.0 (22.25)	5.0 (22.25)	–5.0 (22.25)	0.0	0.0	–5.0 (22.25)
18	0.0	0.0	0.0	0.0	0.0	–5.0 (22.25)
19	0.0	0.0	0.0	0.0	0.0	–5.0 (22.25)
20	0.0	0.0	0.0	0.0	0.0	–5.0 (22.25)

For this example, the best weight of the CSS algorithm is 393.05 lb (178.28 kg), while it is 393.38 lb (178.43 kg) for the HPSACO [4]. The PSOPC and HPSO algorithms do not get optimal results when the maximum number of iterations is reached [14]. The HPSO algorithm gets the optimal solution after 50,000 analyses [14] while it takes just 5,330 and 5,370 analyses for the HPSACO and CSS, respectively. The convergence history for this example is shown in Figure 7. Table 4 compares the results of the CSS algorithm to those of the previously reported methods in the literature.

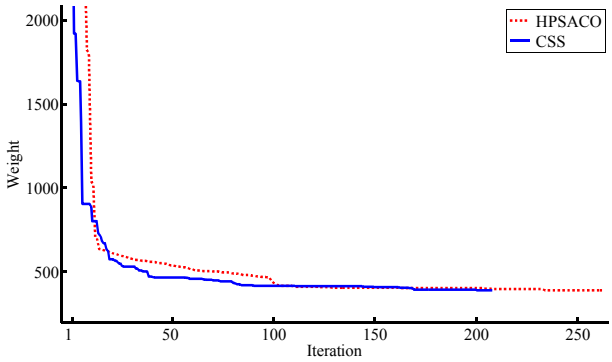

Figure 7. The convergence history for the 72-bar truss structure

Table 4: Optimal design comparison for the 72-bar spatial truss

Optimal cross-sectional areas ($in.^2$)						
Element group		**Wu and Chow [6]**	**Li et al. [14]**		**Kaveh and Talatahari [4]**	**Present study**
		GA	**PSOPC**	**HPSO**	**HPSACO**	**CSS**
1	$A_1 \sim A_4$	0.196	4.490	4.970	1.800	1.990
2	$A_5 \sim A_{12}$	0.602	1.457	1.228	0.442	0.442
3	$A_{13} \sim A_{16}$	0.307	0.111	0.111	0.141	0.111
4	$A_{17} \sim A_{18}$	0.766	0.111	0.111	0.111	0.111
5	$A_{19} \sim A_{22}$	0.391	2.620	2.880	1.228	0.994
6	$A_{23} \sim A_{30}$	0.391	1.130	1.457	0.563	0.563
7	$A_{31} \sim A_{34}$	0.141	0.196	0.141	0.111	0.111
8	$A_{35} \sim A_{36}$	0.111	0.111	0.111	0.111	0.111
9	$A_{37} \sim A_{40}$	1.800	1.266	1.563	0.563	0.563
10	$A_{41} \sim A_{48}$	0.602	1.457	1.228	0.563	0.563
11	$A_{49} \sim A_{52}$	0.141	0.111	0.111	0.111	0.111
12	$A_{53} \sim A_{54}$	0.307	0.111	0.196	0.250	0.111
13	$A_{55} \sim A_{58}$	1.563	0.442	0.391	0.196	0.196
14	$A_{59} \sim A_{66}$	0.766	1.457	1.457	0.563	0.563
15	$A_{67} \sim A_{70}$	0.141	1.228	0.766	0.442	0.442
16	$A_{71} \sim A_{72}$	0.111	1.457	1.563	0.563	0.766
Weight (*lb*)		427.203	941.82	933.09	393.380	393.05

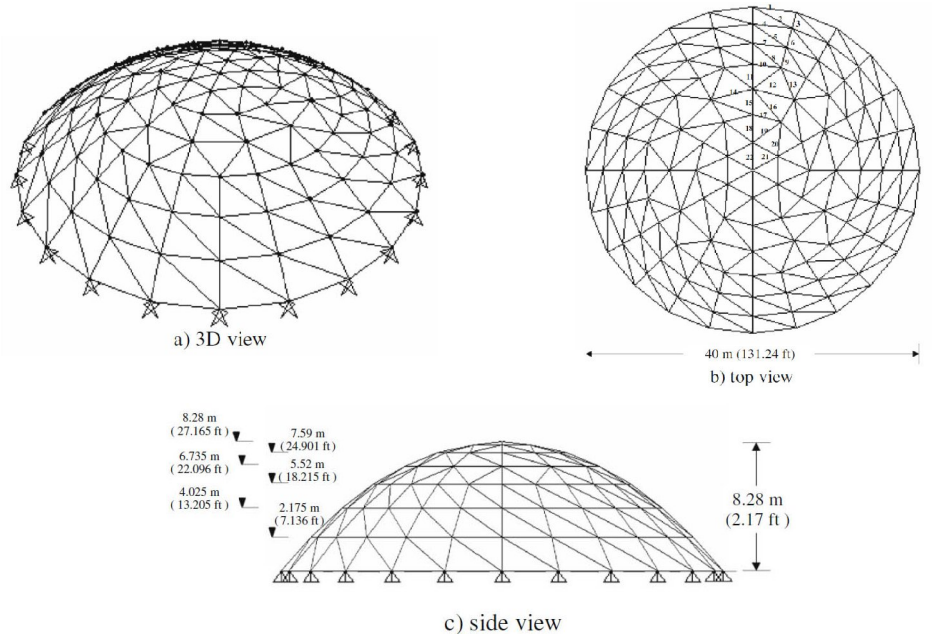

Figure 8. A 354-member braced truss dome (a) 3D view (b) top view and (c) side view

This example has been optimized using 6 meta-heuristic algorithms, previously [17]. Simulated annealing technique results in the least weight compared to other heuristic algorithm, which is 14,760.8 kg. The final designs achieved by evolution strategies and particle swarm optimization methods are both 14,816.3 kg, and are only 0.3% different from the one located by SA. Ant colony optimization, tabu search, harmony search and genetic algorithms achieved 4%, 8.7%, 8.8%, 12.6% heavier designs, respectively [17]. Figure 9 compares the convergence history between these algorithms with the CSS.

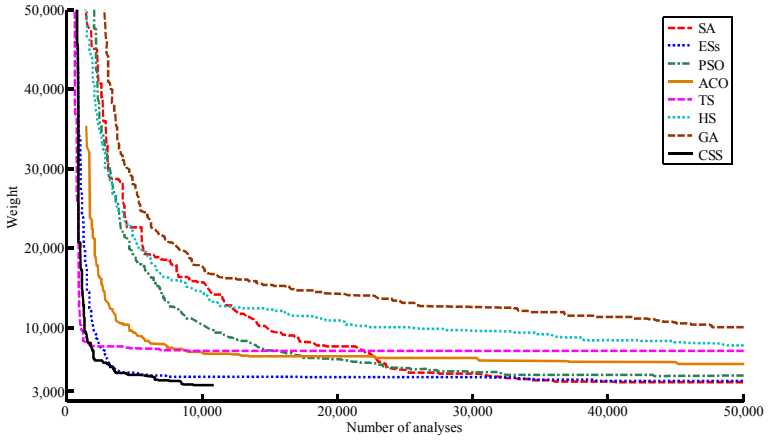

Figure 9. Comparison of the convergence history for the 354-member braced truss dome structure

The best result of the CSS algorithm has the weight of 14,377.4 kg which is 2.5% lighter than the best result of the above-mentioned methods. Table 5 contains the comparison of optimal pipe section design results of the 354-bar dome shaped truss for the CSS and simulated annealing algorithms. For the CSS result, the stress and stability limitations contrary to node displacements are active constraints.

Table 5: Optimal design comparison for the 354-bar dome shaped truss

Element group	**Optimal pipe sections**	
	Hasançebi et al. [17] (SA)	**Present work (CSS)**
1	P2 (6.90 cm^2)	P2 (6.90 cm^2)
2	P3 (14.39 cm^2)	P3.5 (17.29 cm^2)
3	P4 (20.45 cm^2)	P3 (14.39 cm^2)
4	P3.5 (17.29 cm^2)	P3 (14.39 cm^2)
5	P3 (14.39 cm^2)	P3 (14.39 cm^2)
6	P3 (14.39 cm^2)	P3 (14.39 cm^2)
7	P3 (14.39 cm^2)	P3 (14.39 cm^2)
8	P2.5 (10.97 cm^2)	P3 (14.39 cm^2)
9	P3 (14.39 cm^2)	P2.5 (10.97 cm^2)
10	P3 (14.39 cm^2)	P3 (14.39 cm^2)
11	P2.5 (10.97 cm^2)	P2.5 (10.97 cm^2)
12	P2.5 (10.97 cm^2)	P2.5 (10.97 cm^2)
13	P2.5 (10.97 cm^2)	P2.5 (10.97 cm^2)
14	P2.5 (10.97 cm^2)	P2.5 (10.97 cm^2)
15	P2.5 (10.97 cm^2)	P2.5 (10.97 cm^2)
16	P2.5 (10.97 cm^2)	P2.5 (10.97 cm^2)
17	PX2 (9.55 cm^2)	PX2 (9.55 cm^2)
18	PX2 (9.55 cm^2)	PX2 (9.55 cm^2)
19	P2 (6.90 cm^2)	P2 (6.90 cm^2)
20	P2 (6.90 cm^2)	P2 (6.90 cm^2)
21	P2 (6.90 cm^2)	P2 (6.90 cm^2)
22	P2 (6.90 cm^2)	P2 (6.90 cm^2)
Weight kg (lb)	14,760.8 (32,542.3)	14,377.4 (31,696.8)

6. CONCLUDING REMARKS

In this paper, a new discrete optimization approach based on the CSS algorithm is developed for optimal design of truss structures. The other contribution of the paper is the use of a new constraint handling approach, called a fly to boundary method. The fly to boundary method attempts to lead the infeasible solutions toward the boundary space while with the feasible solutions it behaves like the feasible-based method. Using this method changes the position of the infeasible solution to a new one being closer to the boundary.

Several standard truss examples from the literature are presented to demonstrate the effectiveness and robustness of the proposed approach compared with the other meta-heuristics. The results reveal that the CSS performs better than the others. Also, the convergence capability of the CSS method outperformed those of the other approaches. The application of the present CSS algorithm is not limited to truss structures and can also be applied to other types of structural optimization problems, such as frame structures, and plates and shell structures.

REFERENCES

1. Kaveh A, Talatahari S. A novel heuristic optimization method: charged system search, *Acta Mechanica*, 2010, DOI: 10.1007/s00707-009-0270-4.
2. Kaveh A, Talatahari S. Optimal design of skeletal structures via the charged system search algorithm, *Structural and Multidisciplinary Optimization*, 2010, DOI: 10.1007/s00158-009-0462-5.
3. Kaveh A, Talatahari S. A discrete particle swarm ant colony optimization for design of steel frames, *Asian Journal of Civil Engineering*, No. 6, **9**(2008) 563–75.
4. Kaveh A, Talatahari S. A particle swarm ant colony optimization for truss structures with discrete variables, *Journal of Constructional Steel Research*, Nos. 8-9, **65**(2009) 1558–68.
5. Rajeev S, Krishnamoorthy CS. Discrete optimization of structures using genetic algorithms, *Journal of Structural Engineering, ASCE*, No. 5, **118**(1992) 1233–50.
6. Wu SJ, Chow PT. Steady-state genetic algorithms for discrete optimization of trusses, *Computers and Structures*, No. 6, **56**(1995) 979–91.
7. Kameshki ES, Saka MP. Optimum geometry design of nonlinear braced domes using genetic algorithm, *Computers and Structures*, Nos. 1-2, **85**(2007) 71–9.
8. Camp CV, Bichon J, Stovall SP. Design of steel frames using ant colony optimization, *Journal of Structural Engineering*, ASCE, **131**(2005) 369–79.
9. Lee KS, Geem ZW, Lee SH, Bae KW. The harmony search heuristic algorithm for discrete structural optimization, *Engineering Optimization*, No. 7, **37**(2005) 663–84.
10. Kaveh A, Farahmand Azar B, Talatahari S. Ant colony optimization for design of space trusses, *International Journal of Space Structures*, No. 3, **23**(2008) 167–81.
11. Schutte JJ, Groenwold AA. Sizing design of truss structures using particle swarms, *Structural and Multidisciplinary Optimization*, **25**(2003) 261–9.
12. Lee KS, Geem ZW. A new structural optimization method based on the harmony

search algorithm, *Computers and Structures*, **82**(2004) 781–98.
13. Degertekin SO. Optimum design of steel frames using harmony search algorithm, *Structural and Multidisciplinary Optimization*, **36**(2008) 393–401.
14. Li LJ, Huang ZB, Liu F. A heuristic particle swarm optimization method for truss structures with discrete variables, *Computers and Structures*, Nos. 7–8, **87**(2009) 435–43.
15. Kaveh A, Talatahari S. Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures, *Computers and Structures*, Nos. 5-6, **87**(2009) 267–83.
16. Saka MP. Optimum design of steel sway frames to BS5950 using harmony search algorithm, *Journal of Constructional Steel Research*, No. 1, **65**(2009) 36–43.
17. Hasançebi O, Çarbas S, Dogan E, Erdal F, Saka MP. Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures, *Computers and Structures*, Nos. 5-6, **87**(2009) 284–302.
18. Kaveh A, Talatahari S. Size optimization of space trusses!using Big Bang–Big Crunch algorithm, *Computers and Structures*, Nos. 17-18, **87**(2009) 1129–40.
19. Coello, CAC. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, *Computer Methods in Applied Mechanics and Engineering*, **191**(2002) 1245–87.
20. Rosenbrock HH. An automatic method for finding the greatest or least value of a function, *Computer Journal*, No.3, **3**(1960):175–184.
21. Deb K. An efficient constraint handling method for genetic algorithms, *Computer Methods in Applied Mechanics and Engineering*, **186**(2000) 311–38.
22. Kaveh A, Talatahari S. Engineering Optimization with!Hybrid Particle Swarm and Ant Colony Optimization, *Asian Journal of Civil Engineering*, No. 6, **10**(2009) 611–28.
23. Kaveh A. Talatahari S. A Discrete Big Bang–Big Crunch Algorithm for Optimal Design of Skeletal Structures, *Asian Journal of Civil Engineering*, No.1, **11**(2010) 103-122.