International Journal of Architectural, Civil and Construction Sciences
ISSN: 2415-1734
Vol:8, No:7, 2014

Elitist Self-Adaptive Step-Size Search in Optimum Sizing of Steel Structures

Oğuzhan Hasançebi, Saeid Kazemzadeh Azad

***Abstract*—**This paper covers application of an elitist self-adaptive step-size search (ESASS) to optimum design of steel skeletal structures. In the ESASS two approaches are considered for improving the convergence accuracy as well as the computational efficiency of the original technique namely the so called self-adaptive step-size search (SASS). Firstly, an additional randomness is incorporated into the sampling step of the technique to preserve exploration capability of the algorithm during the optimization. Moreover, an adaptive sampling scheme is introduced to improve the quality of final solutions. Secondly, computational efficiency of the technique is accelerated via avoiding unnecessary analyses during the optimization process using an upper bound strategy. The numerical results demonstrate the usefulness of the ESASS in the sizing optimization problems of steel truss and frame structures.

***Keywords*—**Structural design optimization, optimal sizing, metaheuristics, self-adaptive step-size search, steel trusses, steel frames.

I. Introduction

DAILY life is full of instances which need decision making about the best possible solution. By using the shortest path to reach the destination, shopping with a certain budget, or ordering our daily tasks, implicitly we try to find an optimum solution. Generally, time and cost limitations are the two common limitations in real life optimization instances. Similar to frequent daily problems, the field of engineering design includes a wide range of optimization problems as well. Even, it can be mentioned that engineering design without optimization is indeed not meaningful [1]. In particular the optimum design of a structural system is an attempt to find the best arrangement of solution variables that yields a minimum weight or cost design. Furthermore, for practical aspects the final design should satisfy a set of design constraints imposed with respect to a standard code. Basically, the main categories of traditional structural optimization techniques are mathematical programming [2] and optimality criteria [3], [4] approaches. The well known shortcomings of traditional optimization methods are that these techniques are gradient-based and therefore typically work on the basis of continuous solution variables. Furthermore, computing the gradients of highly nonlinear objective functions of practical instances becomes another difficulty when dealing with these techniques. The most recent category of structural optimization techniques is referred to as non-traditional stochastic search methods or metaheuristics. These algorithms, such as genetic algorithms, particle swarm optimization, ant colony optimization, etc., are basically nature inspired approaches, which borrow their working principles from natural phenomena [5]. Different from traditional optimization techniques, metaheuristic algorithms do not perform any gradient based search and are able to handle both discrete and continuous solution variables. In addition, the stochastic nature of metaheuristics makes it more probable to find a near optimum solution even for complicated practical optimization instances. Since the optimization approaches based on metaheuristics are robust and successful in locating the optimal solutions, these algorithms can efficiently be employed for solving practical structural optimization problems. The state-of-the-art reviews of metaheuristics as well as their applications in structural design optimization can be found in [6]-[8].

Although meta-heuristic algorithms are generally conceived to be successful in locating promising solutions for challenging engineering optimization problems, the slow rate of convergence towards the optimum and the need for a high number of structural analyses are known as the main shortcomings of these techniques in practical structural design optimization. Mostly response computations of designs sampled during a search process mostly occupies 85-95% workload of a metaheuristic technique [9], and therefore large number of structural analyses substantially increases the total computing effort. One solution to this is to reduce the total computational time by taking advantage of high performance computing methods, such as parallel or distributed computing techniques [9]. The idea in this approach is to distribute the total workload of the optimization algorithm amongst multiprocessors of a single computer or within a cluster of computers connected to each other via local area network. Another approach, which is more straightforward and easier to apply, is to develop efficient strategies for diminishing the number of structural analyses required in the optimization process. The latter, can be performed through developing efficient optimization techniques capable of locating reasonable solutions using less computational effort. Recently, an upper bound strategy (UBS) is proposed in Kazemzadeh Azad et al. [10], where unnecessary structural analyses are avoided during the course of optimization through a simple and efficient mechanism. The key issue in the UBS is to identify those candidate solutions which have no chance to

Oğuzhan Hasançebi, Associate Professor, Middle East Technical University, Department of Civil Engineering, Ankara, Turkey (phone: +90 312 210 2456; fax: +90 312 210 1262; e-mail: oguzhan@metu.edu.tr).

Saeid Kazemzadeh Azad, Ph.D. Candidate, Middle East Technical University, Department of Civil Engineering, Ankara, Turkey (phone: +90 537 910 8622; e-mail: saeid.azad@ metu.edu.tr).

improve the search during the iterations of the optimum design process. After identifying those non-improving solutions, they are directly excluded from the design population without any structural analysis performed, resulting in a significant saving in computational effort [10].

Self-adaptive step-size search (SASS) algorithm is a recently proposed optimization technique based on a self-adaptive hill-climbing strategy [11]. In addition to its ability for tackling practical optimization problems, the facts that it has a simple algorithmic structure and needs relatively a small number of parameters for implementation are amongst the advantageous features of this technique. In Nolle [12] the SASS algorithm is successfully employed to find the optimum profiles for a simulated rolling mill. Nolle [13] also applied this algorithm to automated Langmuir probe tuning problem and reported numerical results indicating the favorable application of the technique. Later, Nolle and Bland [14] demonstrated the promising performance of the SASS in automatic optimization of standard engineering design problems.

This study covers application of a recently developed ESASS algorithm [15] to discrete sizing of steel truss and frame structures. In the ESASS two approaches are considered for improving the convergence accuracy as well as the computational efficiency of the original technique. Firstly, an additional randomness is incorporated into the sampling step of the technique to preserve exploration capability of the algorithm during the optimization. Moreover, an adaptive sampling scheme is introduced to improve the quality of final solutions. Secondly, computational efficiency of the technique is accelerated via avoiding unnecessary analyses during the optimization process using an upper bound strategy. The numerical results demonstrate the usefulness of the ESASS in the sizing optimization problems of steel truss and frame structures.

II. Optimum Design of Steel Skeletal Structures

For a steel structure composed of N_m structural members collected in N_d member groups, the sizing optimization problem can be formulated as follows.

The objective is to find a vector of integer values $\mathbf{I}$ given in (1) representing the sequence numbers of steel sections assigned to N_d member groups:

$$\mathbf{I}^T = \left[I_1, I_2, ..., I_{N_d}\right] \tag{1}$$

to minimize the penalized weight ($f(\mathbf{X})$) of the structure

$$f(\mathbf{X}) = W(\mathbf{X}) + \phi(\mathbf{X}) \tag{2}$$

where,

$$W = \sum_{i=1}^{N_d} \rho_i A_i \sum_{j=1}^{N_t} L_j \tag{3}$$

In (1)-(3), A_i and ρ_i are the length and unit weight of the steel section adopted for member group *i*, respectively, N_t is the total number of members in group *i*, L_j is the length of the member *j* which belongs to group *i*, and $\phi(\mathbf{X})$ is the penalty function employed for handling the constraints. The optimization constraints consist of the limitations imposed on overall structural response and behavior of individual members which are addressed for each example in Section V.

III. The SASS Optimization Algorithm

Section III covers the optimum design procedure based on the SASS algorithm [14]. The algorithm has a relatively simple outline, which consists of the following steps:

Step 1. Initial population: Form an initial population by spreading *m* solution candidates over the design space. Each solution candidate $\mathbf{X}_i$ is referred to as a particle $\mathbf{P}_i$ $(i = 1,..., m)$ in the SASS algorithm and is considered as a vector of *n* design variables, i.e. $\mathbf{P}_i = (v_{i1}, v_{i2}, ..., v_{in})$.

Step 2. Evaluation of the initial population: Calculate the objective function value of each particle through (2). The fitness value of each particle is computed by either inverting its objective function value, or subtracting it from a constant number chosen large enough to yield always a positive value for all particles.

Step 3. Selecting a particle for improvement: Select a particle for improvement in an optimization cycle. In this process each particle $\mathbf{P}_i$ $(i = 1,..., m)$ is selected once according to its sequence number in the population, and the improvement of this particle is performed as discussed in the following steps.

Step 4. Defining a maximum step size vector ($\mathbf{S}_{maxi}$): For each particle $\mathbf{P}_i$ selected in the previous step, choose two different particles $\mathbf{P}_k$ and $\mathbf{P}_l$ randomly from the population to define the neighborhood of the particle $\mathbf{P}_i$ based on a maximum step size vector $\mathbf{S}_{maxi}$,

$$\mathbf{S}_{maxi} = (S_{maxi1}, S_{maxi2}, ..., S_{maxin}) \tag{4}$$

$$S_{\max ij} = \left| v_{kj} - v_{lj} \right| \quad \text{for} \quad j=1, 2, \ldots, n \tag{5}$$

where, each component of $\mathbf{S}_{maxi}$ is equal to the absolute value of the difference between the corresponding design variables in the particles $\mathbf{P}_k$ and $\mathbf{P}_l$.

Step 5. Sampling: Sample a new particle $\mathbf{P}_i'$ in the neighborhood of the selected particle $\mathbf{P}_i$ based on $\mathbf{S}_{maxi}$ using (6) and (7),

$$v_{ij}' = v_{ij} + step_{ij} \tag{6}$$

$$step_{ij} \in \left[-S_{\max ij}, S_{\max ij}\right] \quad \text{for} \quad j=1, 2, \ldots, n \tag{7}$$

where v_{ij} and v'_{ij} are the j-th design variable in the particles $\mathbf{P}_i$ and $\mathbf{P}'_i$ respectively, and $step_{ij}$ is any number randomly chosen between the range $\left[-S_{\max ij}, S_{\max ij}\right]$ using a uniform distribution.

Step 6. Evaluation of the sampled particle: Calculate the fitness value of the newly sampled particle $\mathbf{P}'_i$.

Step 7. Updating: Compare the sampled particle $\mathbf{P}'_i$ with the original particle $\mathbf{P}_i$ based on their fitness values. If $f(\mathbf{P}'_i) < f(\mathbf{P}_i)$ then $\mathbf{P}_i$ is updated and replaced by $\mathbf{P}'_i$, otherwise $\mathbf{P}_i$ is retained.

Step 8. Termination: Go to Step 3 until a stopping criterion is satisfied, which can be imposed as a maximum number of iterations or no improvement of the best design over a certain number of iterations. It should be noted that one cycle in SASS is composed of m iterations.

IV. The ESASS Optimization Algorithm

A reformulation of the SASS algorithm is proposed in [15] to improve the efficiency of the algorithm in structural design optimization problems. The resulting enhancement of the technique is referred to as elitist self-adaptive step-size search (ESASS) algorithm. The ESASS algorithm exhibits some superiority with respect to its standard variant in terms of both convergence accuracy and computational efficiency. In the following the enhancements in the ESASS algorithm are described in details.

In the SASS algorithm typically the components of step size vector $\mathbf{S}_{maxi}$ are large in the initial cycles due to a random generation of the initial population, and they tend to decrease adaptively with the convergence of the population as the search goes on. This self-adaptive nature of the algorithm is intended to provide a suitable search mechanism by sampling new particles in a restricted, yet more favorable region of the design space in the following cycles. However, when the performance of the algorithm is investigated through numerical examples, it is observed that the $\mathbf{S}_{maxi}$ values tend to become very small or even zero after a certain number of cycles, resulting in negligible or sometimes no changes in the generated particles. It follows that exploration ability of the algorithm vanishes in time, leading to degenerated or sometimes totally disabled search process by the SASS algorithm. As a remedy to this problem, (6) is somewhat modified in the proposed ESASS algorithm. An additional term ($rand_{ij}$) based on a standard normal distribution, $N(0,1)$, with a mean of zero and standard deviation of 1, is used in each iteration with a probability of R_p as follows:

$$v'_{ij} = v_{ij} + step_{ij} + rand_{ij} \tag{8}$$

$$rand_{ij} = \begin{cases} N(0,1) & \text{if } u_{ij} < R_p \\ 0 & \text{if } u_{ij} > R_p \end{cases} \tag{9}$$

where u_{ij} is a uniform random number selected between 0 and 1. The rationale behind (8) and (9) is to facilitate stochastic changes in the generation of new particles to keep alive the exploration capability of the algorithm especially when the $\mathbf{S}_{maxi}$ values are decreased to unnecessarily low values. However, not all components of the particle are subjected to stochastic change; instead this is controlled by the probability R_p. In addition, the use of a normally distributed random number in this formulation ensures that the small perturbations occur more often than the large ones.

On the other hand, some recently developed metaheuristic optimization algorithms based on elitist strategies have been found to be very efficient in locating optimum or near-optimum solutions while tackling complicated design optimization problems [16]-[18]. For instance, two enhanced metaheuristic algorithms [17], [18] that are specifically developed by the authors for handling sizing optimization problems work fundamentally on the basis of an elitist strategy where the new candidate solutions are generated in the vicinity of the current best design.

An attempt is made to utilize an elitist strategy in the ESASS algorithm where the sampling of new particles (Step 5) is encouraged in the neighborhood of the best-so-far particle in accordance with (10),

$$v'_{ij} = v_{ij}^{best} + step_{ij} + rand_{ij} \tag{10}$$

where v_{ij}^{best} refers to the j-th component of the best particle $\mathbf{P}_{best}$ found so far in the optimization process.

It should be noted that (8) and (10) offer two competitive formulations to be used in place of (6) for sampling new particles in searching the design space. Apparently, a more explorative search is provided with (8), whereas (10) motivates a more exploitative search by benefitting from previously visited best solution.

To combine these two useful search features in an efficient manner, an adaptive sampling scheme with the following pseudo-code is developed in the ESASS algorithm:

if ($u_i \le R_s$) **then**
- $\mathbf{P}_{selected} = \mathbf{P}_i$
- Sample new particle $\mathbf{P}'_i$ using (8)

else
- $\mathbf{P}_{selected} = \mathbf{P}_{best}$
- Sample new particle $\mathbf{P}'_i$ using (10)

where

$$R_s^{t+1} = \begin{cases} R_s^t & \text{if } \mathbf{P}_{best} \text{ is not improved at iteration } t \\ R_s^t + 0.01 & \text{if } \mathbf{P}_{best} \text{ is improved by Eq.(11) at iteration } t \\ R_s^t - 0.01 & \text{if } \mathbf{P}_{best} \text{ is improved by Eq.(13) at iteration } t \end{cases} \quad (11)$$

In the proposed adaptive sampling scheme a new particle $\mathbf{P}_i'$ is generated by applying either one of these two sampling equations (8) and (10) probabilistically. Here the sampling probability parameter $R_s \in [0,1]$ controls the sampling scheme to be implemented when generating a new particle. For each particle, a uniform random number u_i is generated anew between 0 and 1, and at times when $u_i \le R_s$, the new particle is sampled using (8), otherwise it is generated using (10). It follows that the probability of sampling a new particle with (8) and (10) is R_s and $1-R_s$, respectively. The R_s parameter is initially set to 0.5 to give an equal chance to either sampling scheme in the beginning. However, when iteration of a cycle is completed (i.e. when *m* number of particles are sampled and evaluated in a cycle) R_s is updated adaptively using (11), and this way the search is biased towards the sampling scheme that exhibits a better performance at the previous iteration.

In (11), R_s^t and $\underline{R_s^{t+1}}$ represent the sampling probability parameters at cycles *t* and *t*+1, respectively. Accordingly, if the best design $\mathbf{P}_{best}$ is improved by a particle sampled using (8) at the previous iteration, R_s is increased by 0.01; otherwise if $\mathbf{P}_{best}$ is improved by a particle sampled using (10), then R_s is lowered by 0.01. No update of R_s is carried out if $\mathbf{P}_{best}$ is not improved at the previous iteration.

On the other hand the computational efficiency of the ESASS algorithm is accelerated via the recently developed UBS method [10]. In this approach, basically the penalized weight of a current solution is considered as an upper bound limit for the net weight of a newly generated solution. Accordingly, a new solution with a net weight greater than this limit is excluded from the structural analysis stage. This strategy is used in the ESASS algorithm as follows. Here, after a new particle $\mathbf{P}_i'$ is sampled in Step 5 in the vicinity of a selected particle $\mathbf{P}_i$ or $\mathbf{P}_{best}$, first the net weight of $\mathbf{P}_i'$, i.e. $W(\mathbf{P}_i')$, is calculated only; not the penalized weight. This computation is straightforward and can be done with a trivial computational effort. If $\mathbf{P}_i'$ has a net weight smaller than or equal to the penalized weight of the selected particle $f(\mathbf{P}_{selected})$, the structural analysis of the new particle is processed and its penalized weight is computed. In the opposite case, i.e. $W(\mathbf{P}_i') > f(\mathbf{P}_{selected})$, however, the upper bound rule is activated and $\mathbf{P}_i'$ is automatically excluded from the structural analysis phase required for response computations in Step 6, since such a candidate is unlikely to improve the selected design.

V. Numerical Examples

Section V presents application of the ESASS algorithm to sizing optimization of steel skeletal structures. The design examples covered here include a 135-member steel frame and a 200-bar planar truss. The optimum solutions found for these structures with the ESASS algorithm are compared to those achieved using other contemporary metaheuristic algorithms. It is worth mentioning that a population size of 50 is used for the ESASS algorithm.

A. 135-Member Steel Frame

The first optimization instance is a 3-story steel frame depicted in Fig. 1, composed of 135 members including 66 beam, 45 column and 24 bracing elements. The stability of structure is provided through moment-resisting connections as well as inverted V-type bracing systems along the *x* direction. For practical fabrication requirements the 135 members of the frame are collected under 10 member groups. Here, the columns are grouped into four sizing variables in a plan level as corner, inner, side x-z and side y-z columns, and they are assumed to have the same cross-section over the three stories of the frame. On the other hand, all the beams in each story are grouped into one sizing variable, resulting in three beam-sizing design variables. Similarly, all the bracings in each story are grouped into one sizing variable, resulting in three bracing-sizing design variables for the frame. Further member grouping details for this example can be found in [10].

For design purpose, the frame is subjected to the following 10 load combinations to ASCE 7-98 [19]:

(1) $1.4D$
(2) $1.2D + 1.6L$
(3) $1.2D + 1.0E_x + 0.5L$
(4) $1.2D + 1.0E_{ex} + 0.5L$
(5) $1.2D + 1.0E_y + 0.5L$
(6) $1.2D + 1.0E_{ey} + 0.5L$
(7) $0.9D + 1.0E_x$
(8) $0.9D + 1.0E_{ex}$
(9) $0.9D + 1.0E_y$
(10) $0.9D + 1.0E_{ey}$

where *D* and *L* denote the dead and live loads, respectively; E_x and E_y are the earthquake loads applied to the center of mass in *x* and *y* directions, respectively; and E_{ex} and E_{ey} are the earthquake loads applied considering the effect of accidental eccentricity of the center of mass in *x* and *y* directions, respectively. Based on ASCE 7-98 [19] the amount of eccentricity is set to 5% of the dimension of the structure perpendicular to the direction of the applied earthquake load.

The live loads acting on the floor and roof beams are 12 and 7 kN/m, respectively. The dead loads consist of the self-weight of the structure in addition to the uniformly distributed loads of 20 and 15 kN/m applied on floor and roof beams, respectively.

The earthquake loads, are calculated based on the equivalent lateral force procedure outlined in ASCE 7-98 [19].

Here, the resulting seismic base shear (V) is taken as $V = 0.15W_s$ where W_s is the total dead load of the building. The computed base shear is distributed to each floor based on the following equation:

$$F_x = \frac{w_x h_x^k V}{\sum_{i=1}^{n} w_i h_i^k} \quad (12)$$

where F_x is the induced lateral seismic force at level *x; w* is portion of the total gravity load assigned to the related level (i.e. level *i* or *x*); and *h* is the height from base to the related level. The parameter *k* is determined based on the structure period. It is equal to 1 for structures with a period of 0.5 sec or less; and 2 for structures with a period of 2.5 sec or more. For structures with a period in range of 0.5 to 2.5 sec, *k* is calculated through linear interpolation [19]. It is worth mentioning that the period of the structure is calculated using the following equation given in ASCE 7-98 [19].

$$T = C_T h_n^{3/4} \quad (13)$$

where C_T is taken as 0.0853 and h_n is the height of the building; namely 12 m for this example. Hence, the period of the structure, *T*, is 0.55 sec. Based on the period obtained the value of parameter *k* in (12) is taken as 1.025 for this example. It should be noticed that since the self-weight of the structure changes during the course of optimization, the dead and earthquake loads are not stationary.

The beam elements are continuously braced along their lengths by the floor system; and columns and bracings are assumed to be unbraced along their lengths. The effective length factor, *K*, is taken as 1 for all beams and bracings. The *K* factor is conservatively taken as 1.0 for buckling of columns about their minor (weak) direction, since the frame is assumed to be non-swaying in that direction owing to inverted V-type bracing systems. However, for buckling of columns about their major direction the *K* factor is calculated [10].

The maximum lateral displacement of the top story is limited to 0.03 m and the upper limit of interstory drift is taken as *h*/400, where *h* is the story height. The interstory drifts are calculated based on the displacement of center of mass of each story. The maximum lateral displacement of the top story is calculated with respect to the maximum displacements of the ends of the structure. Here, horizontal displacements of all joints of each story are constrained to each other based on a rigid diaphragm assumption.

Discrete sizing of the frame is previously carried out in [10] via some contemporary metaheuristics, i.e. the upper bound strategy (UBS) integrated big bang-big crunch algorithm (UBB-BC), as well as its two enhanced variants i.e. UBS integrated modified and exponential big bang-big crunch algorithms (UMBB-BC and UEBB-BC). Moreover, this instance is also solved in [20] using a UBS integrated particle swarm optimization algorithm (UPSO).

Table I presents a comparison of optimum solutions located using different algorithms. As can be seen from this table, the ESASS yields a design weight of 44.33 ton for this example. Other solutions obtained are 38.91 ton by UEBBBC, 45.67 ton by UMBB-BC, 47.3 ton by UBB-BC, and 55.66 ton by UPSO. These design weights are obtained using 1542 by ESASS, 1235 analyses by UEBBBC, 1794 analyses by UMBB-BC, 880 analyses by UBB-BC, and 1574 analyses by UPSO. It can be observed that the ESASS algorithm shows a promising performance which is comparable to the aforementioned contemporary enhanced optimization algorithms both in terms of solution quality as well as computational efficiency. Furthermore, the ESASS optimization algorithm needs few parameters for implementation.

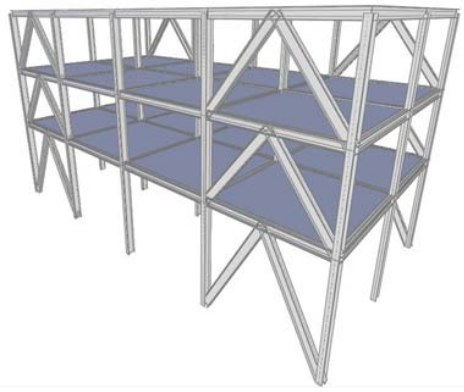

Fig. 1 135-Member Steel Frame

TABLE I
COMPARISON OF RESULTS FOR 135-MEMBER STEEL FRAME

Sizing variables	UPSO	UBB-BC	UMBB-BC	UEBB-BC	ESASS
1	W8X28	W10X39	W30X90	W21X62	W8X28
2	W33X118	W27X84	W14X48	W14X48	W36X160
3	W40X167	W40X149	W40X215	W36X150	W18X60
4	W14X53	W18X65	W27X84	W21X68	W14X53
5	W14X30	W21X44	W14X34	W18X40	W14X22
6	W24X55	W16X40	W12X35	W18X35	W16X31
7	W16X26	W10X22	W18X35	W16X26	W18X40
8	W14X30	W27X84	W21X44	W8X24	W8X24
9	W40X149	W16X26	W10X22	W16X26	W6X15
10	W27X84	W21X44	W6X15	W6X15	W16X50
Weight (ton)	55.66	47.3	45.67	38.91	44.33
Analyses	1574	880	1794	1235	1542

B. 200-Bar Truss Structure

Discrete sizing of the 200-bar truss structure shown in Fig. 2 is considered as the second design optimization example. The density of the material is 0.283 lb/in.^{3} (7833.41 kg/m^3) and the modulus of elasticity is 30,000 ksi (20,6842.8 MPa). The members are only subjected to the stress constraints with

limits of ±10 ksi (68.948 MPa). Here, the structure is subjected to three independent loading conditions:(i) 1.0 kip acting in the positive x direction at nodes 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, 71, (ii) 10 kips acting in the negative y direction at nodes 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, 26, . . . , 71, 72, 73, 74, and 75, (iii) Conditions 1 and 2 acting together.

Since the members of the truss are linked into 29 groups, there are totally 29 sizing design variables. This design optimization problem is formerly solved by GA in [21] using discrete design variables taken from the list S = [0.1, 0.347, 0.44, 0.539, 0.954, 1.081, 1.174, 1.333, 1.488, 1.764, 2.142, 2.697, 2.8, 3.131, 3.565, 3.813, 4.805, 5.952, 6.572, 7.192, 8.525, 9.3, 10.85, 13.33, 14.29, 17.17, 19.18, 23.68, 28.08, 33.7] (in.^{2}). For the sake of conformity, the same list of discrete sections is employed in this study for sizing the truss members.

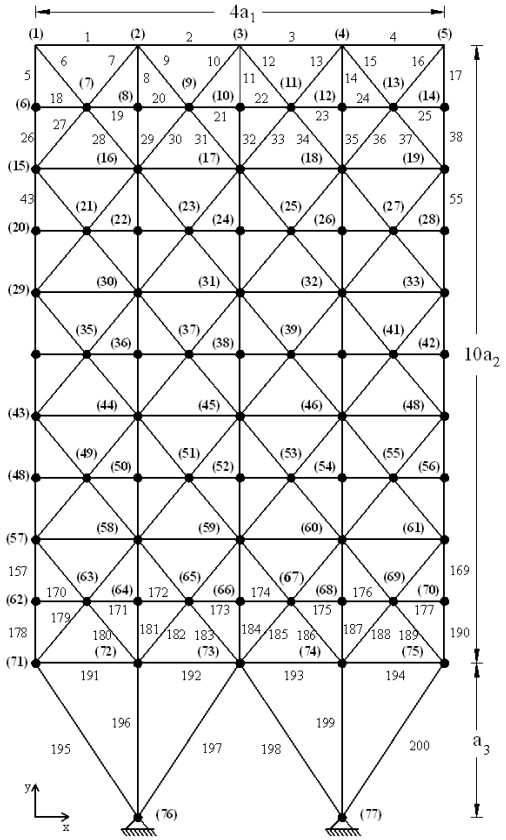

Fig. 2 200-Bar Truss (a_1 = 240 in., a_2 = 144 in., a_3 = 360 in.)

Discrete sizing of the 200-bar truss structure is performed using the ESASS algorithm and the optimum design found is presented in Table II. For comparison purpose the results for this example using the ARCGA [22] and MABC [23] algorithms are also presented in Table II. It can be seen from Table II that the ESASS obtains a design weight of 28075.488 lb which is lighter than the design weights obtained by the other methods. Moreover, the number of structural analyses performed using the ESASS algorithm is only 11156, which is significantly less than those of the GA (i.e. 51360), ARCGA (i.e. 25000), and MABC (i.e. 40000) algorithms. The optimization history showing the variation of the best penalized weight during the cycles of the ESASS algorithm is illustrated in Fig. 3.

TABLE II
OPTIMAL CROSS SECTIONAL AREAS (IN.^{2}) FOR THE 200-BAR TRUSS STRUCTURE

Sizing variables	GA [21]	ARCGA	MABC	ESASS
1	0.347	0.1	0.1	0.1
2	1.081	1.081	1.333	0.954
3	0.1	0.1	0.1	0.1
4	0.1	0.1	0.1	0.1
5	2.142	2.142	2.697	2.142
6	0.347	0.347	0.347	0.347
7	0.1	0.1	0.1	0.1
8	3.565	3.131	3.131	3.131
9	0.347	0.1	0.1	0.1
10	4.805	4.805	4.805	4.805
11	0.44	0.347	0.44	0.347
12	0.44	0.1	0.539	0.1
13	5.952	5.952	5.952	5.952
14	0.347	0.1	0.1	0.1
15	6.572	6.572	6.572	6.572
16	0.954	0.539	1.081	0.44
17	0.347	1.081	0.347	0.539
18	8.525	7.192	8.525	7.192
19	0.1	0.539	0.1	0.44
20	9.3	8.525	9.3	8.525
21	0.954	1.333	0.954	0.954
22	1.764	1.081	1.764	1.174
23	13.33	10.85	13.33	10.85
24	0.347	0.1	0.44	0.44
25	13.33	13.33	13.33	10.85
26	2.142	1.488	2.142	1.764
27	4.805	5.952	3.813	8.525
28	9.3	13.33	8.525	13.33
29	17.17	14.29	19.18	13.33
Weight (lb)	28544.014	28347.594	28366.365	28075.488
Analyses	51360	25000	40000	11156

VI. CONCLUSION

In this paper, application of the recently developed ESASS algorithm to optimum design of steel skeletal structures is presented. Basically, in the ESASS two strategies are considered for improving the convergence accuracy as well as the computational efficiency of the original technique. On the one hand, an additional randomness is incorporated into the sampling step of the technique to preserve exploration

capability of the algorithm during the optimization. On the other hand, an adaptive sampling scheme is introduced to improve the quality of final solutions. Furthermore, as a result of integrating the UBS with the ESASS algorithm the total number of required structural analyses is reduced. The numerical investigations using steel truss and frame structures indicate a promising performance of the ESASS algorithm with an acceptable level of comparability to the contemporary enhanced optimization algorithms both in terms of solution quality as well as computational efficiency.

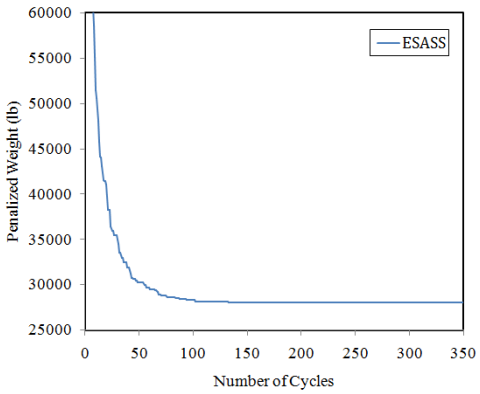

Fig. 3 Optimization History of 200-Bar Truss Using the ESASS

REFERENCES

[1] S. Kazemzadeh Azad, O. Hasançebi, O.K. Erol, "Evaluating efficiency of big-bang big crunch algorithm in benchmark engineering optimization problems", Int. J. Optim. Civil. Eng., 1(3), 495–505, 2011.

[2] F. Erbatur, M.M. Al-Hussainy, "Optimum design of frames", Comput. Struct, 45(5–6), 887–891, 1992.

[3] E.I. Tabak, P.M. Wright "Optimality criteria method for building frames", J. Struct. Div., ASCE, 107(7), 1327–1342, 1981.

[4] M.P. Saka, "Optimum design of steel frames with stability constraints" Comput. Struct., 41(6), 1365–1377, 1991.

[5] X-S. Yang, "Nature-inspired metaheuristic algorithms" Luniver Press, 2008.

[6] L. Lamberti, C. Pappalettere, "Metaheuristic design optimization of skeletal structures: a review", Computational Technology Reviews, 4, 1-32, 2011. doi:10.4203/ctr.4.1

[7] M.P. Saka, "Optimum design of steel frames using stochastic search techniques based in natural phenomena: a review", in B.H.V. Topping, (Editor), "Civil Engineering Computations: Tools and Techniques", Saxe-Coburg Publications, Stirlingshire, UK, Chapter 6, pp 105-147, 2007.

[8] M.P. Saka, E. Doğan, "Recent developments in metaheuristic algorithms: a review" in B.H.V. Topping, (Editor), "Computational Technology Reviews", Saxe-Coburg Publications, Stirlingshire, UK, 31-78, 2012.

[9] O. Hasançebi, T. Bahçecioğlu, Ö. Kurç, M.P. Saka, "Optimum design of high-rise steel buildings using an evolution strategy integrated parallel algorithm", Comput. Struct., 89, 2037-2051, 2011.

[10] S. Kazemzadeh Azad, O. Hasançebi, S. Kazemzadeh Azad, "Upper Bound Strategy for Metaheuristic Based Design Optimization of Steel Frames", Adv. Eng. Software, 57, 19–32, 2013.

[11] L. Nolle, "On a hill-climbing algorithm with adaptive step size: towards a control parameter-less black-box optimisation algorithm", In: B. Reusch (Ed.), Computational Intelligence, Theory and Applications, Springer, pp. 587–595, 2006.

[12] L. Nolle, "SASS applied to optimum work roll profile selection in the hot rolling of wide steel", Knowledge-Based Systems, 20, 203–208, 2007.

[13] L. Nolle, "SASS Applied to Automated Langmuir Probe Tuning", In: Proceedings of Asia Modelling Symposium 2007, Phuket, Thailand, pp. 421–425, 2007.

[14] L. Nolle, J.A. Bland, "Self-adaptive step-size search for automatic optimal design", Knowledge-Based Systems, 29, 75–82, 2012.

[15] S. Kazemzadeh Azad, O. Hasançebi, "An Elitist Self-Adaptive Step-Size Search for Structural Design Optimization, Applied Soft Computing, in press, 2014.

[16] O. Hasançebi, S. Kazemzadeh Azad, "An efficient metaheuristic algorithm for engineering optimization: SOPT", Int. J. Optim. Civil Eng. 2, 479–487, 2012.

[17] O. Hasançebi, S. Kazemzadeh Azad, "Discrete size optimization of steel trusses using a refined big bang–big Crunch Algorithm", Eng. Optim., 46, 61–83, 2014.

[18] O. Hasançebi, S. Kazemzadeh Azad, "An exponential big bang-big crunch algorithm for discrete design optimization of steel frames", Comput. Struct. 110–111, 167–179, 2012.

[19] ASCE 7-98, "Minimum design loads for buildings and other structures: Revision of ANSI/ASCE 7-95", American Society of Civil Engineers, 2000.

[20] S. Kazemzadeh Azad, O. Hasançebi, "Improving Computational Efficiency of Particle Swarm Optimization for Optimal Structural Design", Int. J. Optim. Civ. Eng., 3, 563–574, 2013.

[21] V. Togan, A.T. Daloglu, "An improved genetic algorithm with initial population strategy and self-adaptive member grouping", Comput. Struct., 86, 1204–1218,2008.

[22] K. Koohestani, S. Kazemzadeh Azad, "An Adaptive real-coded genetic algorithm for size and shape optimization of truss structures", The First International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering, B.H.V. Topping, Y. Tsompanakis (Eds.), Civil-Comp Press, Stirlingshire, UK, Paper 13, 2009.

[23] A. Hadidi, S. Kazemzadeh Azad, S. Kazemzadeh Azad, "Structural optimization using artificial bee colony algorithm", The 2nd International Conference on Engineering Optimization (EngOpt), Lisbon, Portugal, 2010.