INTERNATIONAL JOURNAL OF OPTIMIZATION IN CIVIL ENGINEERING
Int. J. Optim. Civil Eng., 2011; 1:29-46

A HYBRID MODIFIED GENETIC-NELDER MEAD SIMPLEX ALGORITHM FOR LARGE-SCALE TRUSS OPTIMIZATION

H. Rahami[1], A. Kaveh[2*, †], M. Aslani[1] and R. Najian Asl[1]
[1]*Faculty of Engineering, University of Tehran, Tehran, Iran*
[2]*Iran University of Science and Technology, Narmak, Tehran-16, Iran*

ABSTRACT

In this paper a hybrid algorithm based on exploration power of the Genetic algorithms and exploitation capability of Nelder Mead simplex is presented for global optimization of multi-variable functions. Some modifications are imposed on genetic algorithm to improve its capability and efficiency while being hybridized with Simplex method. Benchmark test examples of structural optimization with a large number of variables and constraints are chosen to show the robustness of the algorithm.

Received: September 2010; Accepted: May 2011

KEY WORDS: hybrid algorithm, optimization, genetic algorithms, Nelder Mead simplex, trusses

1. INTRODUCTION

Recently global optimization has been a concern of researchers, especially when the fitness function is dependent on a large number of variables or it is strictly confined to some constraints. Hybridizing is the only motivation that brings back the hope to reach this goal as long as using only one search algorithm leads to dim results. A combination of two algorithms, in which one explores a promising area likely to contain global minima, and the other exploits the area to find the desired point would be promising if properly performed. Global methods like simulated annealing, tabu search, genetic algorithm, etc are known as efficient search engines to find and localize areas containing global minima, but they are

[*]Corresponding author: A. Kaveh, Centre of Excellence for Fundamental Studies in Structural Engineering, Iran University of Science and Technology, Narmak, Tehran-16, Iran
[†]E-mail address: alikaveh@iust.ac.ir

very much time consuming while converging to a specified point (if not get trapped in local minimum). On the other hand local methods including Nelder-Mead Simplex, hill climbing etc, are good for exploitation of the search domain. In this study we present a hybrid algorithm based on exploitation (diversification) power of genetic algorithm and exploitation (intensification) feature of the Nelder-Mead Simplex.

The advantage of the simplex search method is that it is straightforward in an algorithmic sense and computationally efficient. However, as a result of using only local information, when they converge to a stationary point, there is no guarantee that the global optimum is found unless the domain in which the global minimum lies is provided. In contrast, a GA method explores the global search space without using local information of promising search directions. But its computational cost is comparatively high. A hybrid algorithm using both of these characteristics (exploration and exploitation) would be promising to find global minima in a broad types of problems. It is practically important to note that each of these algorithms and the new hybrid algorithm presented in this study do not require gradient or Hessian matrix calculations, and therefore it does not suffer from the weakness of classical optimization methods. The main goals of the present hybrid algorithm are as follows:

- Reliability: A proper functioning of exploration and exploitation of the search domain (sometimes referred to as “diversification” and “intensification”) in order to find the true global minima.
- Efficiency: Using simple but effective combination to reduce the total amount of function evaluation.

Up to now we introduced the concepts related to this subject. Such general aspects can also be found in Refs. [1-6].

Proper numerical example plays an important role to reveal the robustness of a new algorithm. The large number of design variables and their mixed (discrete/continuous) nature has rendered the structure optimization a perfect benchmark for large scale search algorithms. The emergence of Evolutionary Algorithms, with their ability to simultaneously examine numerous areas of vast, multimodal search spaces resulted in more reliable and noticeably less expensive designs.

After this introduction, the present paper is organized in the following order: Section 2 is devoted to the general description of the Genetic Algorithm and Nelder Mead Simplex. In Section 3 the hybrid mechanism is fully described and some modifications on the compatibility and efficiency of the algorithm are proposed. Numerical examples and the application of the algorithm are presented in Section 4. Concluding remarks form the content of Section 5.

2. GENETIC ALGORITHM AND NELDER-MEAD SIMPLEX

2.1. Genetic Algorithm

The genetic algorithm (GA) is a search heuristic that mimic the genetic processes of biological organisms. This heuristic is widely used to explore useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), and uses *stochastic direct* optimization methods. The term stochastic indicates that GA uses random operators, and therefore may result in a different (set of) solutions each time they are run. They are also direct, which means they only work with the value of the objective function itself and not its derivatives. GA makes a powerful, universal tool that can solve almost any type of optimization problems including constrained and unconstrained, single- and multi-variable, linear and nonlinear problems with continuous or discrete variables.

They take an initial population of artificial chromosomes and let them evolve toward optimal solution(s) according to the principles of "natural selection" and "survival of the fittest". The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness (measure of desirability) of every individual in the population is evaluated. Multiple individuals are then stochastically selected from the current population (based on their fitness), and modified using such genetic operators as recombination (crossover) and mutation to form a new population. The new population is then used as the “current population” in the next iteration of the algorithm. The average fitness of the new generation is expected to be higher than that of the previous generation, as the best individuals of the last generation have been given higher chances for breeding. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

Finding the optimal solution of complex high dimensional and multimodal problems often requires very expensive fitness function evaluations. As a result of this, existence of an efficient optimization algorithm seems to be vital. Working with a bunch of solution vectors in each generation, time efficiency has always been a concern in GAs. Other criticisms are also mentioned about the functionality of GA but the worst among these defects is getting trapped in the local optima of the objective function in which an unrealistic solution regarded as the global minima is achieved.

In order to make GA a more reliable algorithm and in relation with optimization efficiency (better solutions with fewer function evaluation) many mechanisms have been suggested. One is improvements upon the mechanism of the algorithm, such as modification of genetic operators, or the use of niche technique, etc; the other is hybridizing of GA with other optimization methods, especially the ones working in local search spaces such as the Nelder-Mead Simplex, simulated annealing (SA), etc.

2.2. Nelder Mead simplex

The Nelder–Mead simplex (NMS) search method is based upon the work of Spendley et al. [7]. A simplex is a geometrical figure consisting in n-dimentions, of $(n+1)$ points: $x_0,...,x_n$. If any point of a simplex is taken as the origin, then other points define vectorer directions that span the n-dimention vector scace. Through a sequence of elementary geometric transformation (reflection, contraction, expansion and multi-contraction), the initila simplex moves, expands or contrasts. To select appropriate transformation, the method only uses the values of the function to be optimizaed at the vertices of the simplex considered. After each transformation, the current worst vertex is replaced by a better one. The trial movements in

Figure 1 are generated according to these opeartors (x_h: Represents the vertex where the objective function is the highest and x_l represents the vertex where the objective function is the lowest).

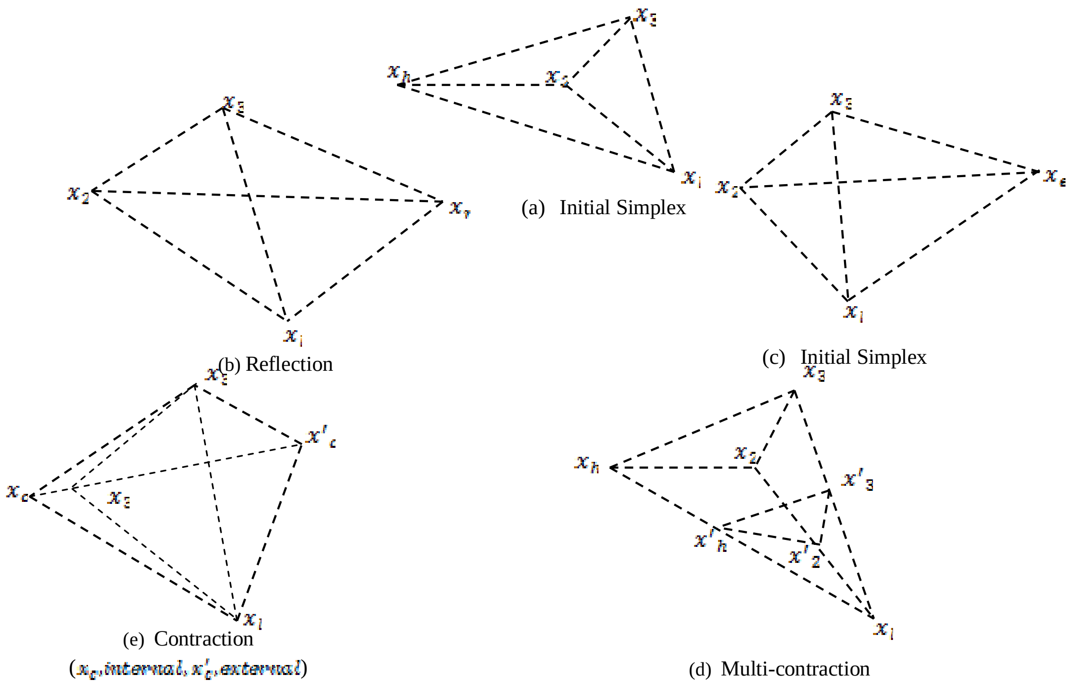

Figure 1. Simplex procedures (reflection, contraction, expansion and multi-contraction)

Operators in simplex method are defined as follow: $(\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i)$

- Reflection: $x_r = (1+\alpha)\bar{x} - \alpha x_0$
- Expansion: $x_e = (1-\gamma)\bar{x} + \gamma x_r$
- Contraction: $x_c = (1-\beta)\bar{x} + \beta x_0$

α, β *and* γ are real parameters that control the operators of the Simplex.

Convergence critertion is a measure of how a solution has moved from iteration (k) to $(k+1)$. So the algorithm stops whenever [8]:

$$\frac{1}{n}\sum_{i=1}^{n}\left\| x_i^{k+1} - x_i^k \right\|^2 < \varepsilon \tag{1}$$

Where x^{k+1} indicates the vertex replacing x^k at the iteration $(k+1)$, and ε is a small real positive number.

The Nelder-Mead simplex method has been applied in physics [9], crystallography [10], biology [11], chemistry [12] and health care [13]. Fletcher [14] considers the Nelder-Mead technique as one of the most successful methods that merely compare function values. There have been abundant studies devoted to various modifications of the Nelder-Mead simplex method appearing in the literature, such as Barton and Ivey [15], Nazareth and Tzeng [16], etc. However, the simple unmodified Nelder-Mead version of the simplex algorithm is best suited to our needs to hybridize with GA.

3. HYBRID ALGORITHM GA-NMS

3.1. Structure of the GA-NMS

In this section the hybrid algorithm GA-NMS is presented in detail. The main idea behind this algorithm is to combine the advantages of each algorithm in a way to avoid their disadvantages. It should also be pointed out that this hybrid algorithm is coded to deal with highly challenging problems as it will be shown in Section 4, and for regular problems considering all of these specifications are not valuable. First the general algorithm is presented and then some modifications are performed to improve the compatibility and efficiency of the algorithm.

3.1.1. Initialization

In the first step the parameters of the algorithm are initialized, these parameters include the simple GA parameters like rate of the Crossover (R_c), Mutation probability (P_M), the length of the Chromosomes and the number of population, and the controlling parameters that determines the interference of each algorithm (GA and Simplex). Recommendation on the magnitude of GA parameters is beyond the scope of this research, because they are highly dependent on the type of the problem being studied. However some modifications will be presented to improve their efficiency.

3.1.2. Exploration of the search domain

In this section the GA is ignited by the randomly selected chromosome. The algorithm takes these initial points to make up the first mating pool in which the first generation will be born afterwards. Experiments show that a properly coded GA can explore the search domain to the global solution at the first 10,000~30,000 function evaluations, although this magnitude is highly dependent on the complexity of the problem. We want to ignore the common convergence criteria of the GAs by gradually decreasing the interference of the GA in the solution of the algorithm. According to this consideration, in the first steps, the GA is the only working algorithm that travels around the search domain to fine out the optimum regions and may be stopped if one of these criteria is fulfilled.

3.1.3. Exploitation

Here we define a controlling parameter which defines the interference of each sub algorithm in the solution of the main algorithm. We want to transform the solution from an exploration one to exploitation. Thus by calling the interference rate of the GA and the Simplex as I_G, I_S respectively, and by considering that $I_G + I_S = 0.9$, we aim to transform the solution. The number of solution points that each algorithm works in this section is equal to the number of population multiplied by I_G or I_S. In other words if we set a ranked population, we send 0.1 of the best population from this generation to the next generation without any change. This mechanism which is called *elitism* seems to secure the place of the best chromosomes in this transmission. The other part of population is transformed from GA or Simplex according to the steps in Table 1: (A population of 100 is assumed to exist).

Table 1. Operation of controlling operators

	Step 1	**Step 2**	**Step 3**	**Step 4**	**Step 5**
Iteration number	100	200	250	300	400
I_G	0.9	0.5	0.4	0.2	0 (Stops)
I_S	0	0.4	0.5	0.7	0.9
Genetic population	100	60	50	40	0 (Stops)

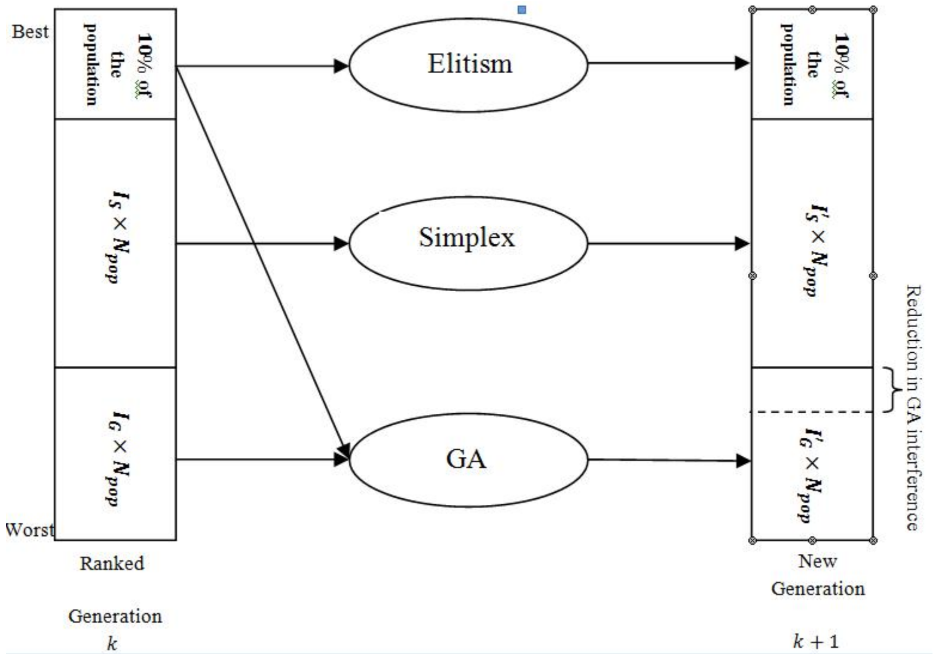

Figure 2. Schematic diagram of the GA-NMS algorithm

Figure 2 shows a schematic view of the GA-NMS algorithm. In addition in order to make better generation and mate some of the good features from the best population, we send the best 10% of the population to the GA in order to make better off-springs.

The operation finally changes to a pure exploitation which stops whenever the Simplex convergence critertion defined by Eq. (1) is satisfied. As we see the number of population that goes under the operation of Simplex gradually increases and on the other hand in a simple Simplex method we need at least n+1 vertices (n determines the number of variables). The number of vertices which can be used and the number of separate Simplex that can be run among these points are optional and they can be adjusted by user since they are mainly dependent on the number of population. Experiments show that if we divide the population in a way that leads to a maximum number of sub groups fitted to Simplex algorithm, better results will be achieved.

3.2. Modifications

3.2.1. Improving GA operators

3.2.1.1. Reproduction

A variety of methods have been introduced for reproduction in GA. Goldberg introduced the application of roulette wheel as the most famous tool for this operator. A very easy and more efficient way to choose parents is to randomly choose 2 chromosomes and compare their fitness. The chromosome with better fitness will be the candidate for being a parent. This method is called Tournament Selection. Selection pressure is easily adjusted by changing the tournament size. If the tournament size is larger, weak individuals have a smaller chance to be selected. This method can reproduce better generation as it explores a theme of elitism.

3.2.1.2. Dynamic mutation

A method of dynamic establishment of mutation probability in the genetic algorithm can be useful as we aim at exploration property of the GA. The comparison of the results shows an oscillating generations but very swift steps are performed to reach the region of solution. The mixture of reproduction method explained above and the Dynamic Mutation leads to a great combination hoping to achieve the exploration capacity of the GA. A comparison of the common mutation magnitude and the dynamic values is presented in Figure 3. The dynamic value is obtained by easily dividing the P_m by the generation number. A big value of P_m=0.1 in comparison of ubiquitous magnitudes was considered that gradually decreases in order to stabilize the generation.

3.2.2. Compatibility with constrained problems

A simple GA and the Simplex method were both compatible with unconstrained problems. By defining the application of penalty method we transform the problem into an unconstrained one. As we know a constrained problem is one in which the feasible region is defined by a set of implicit/explicit constraints. As an example to optimize weight of a truss we are searching for the lowest cross section of members and the constraints are stress and displacement of members. Here stress and displacement are indirect constraints as they are not applied directly on the cross section of the members i.e. the variables of the fitness function.

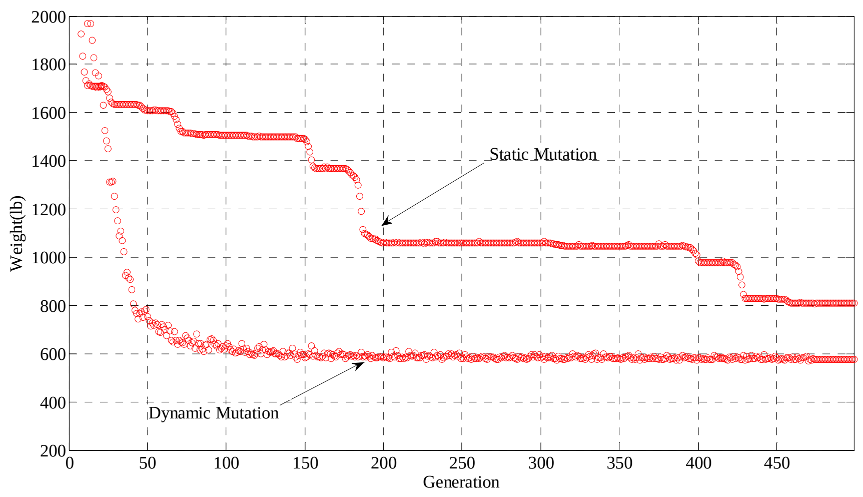

Figure 3. Comparison between static and dynamic mutations

To shift the problem to a normal form, we easily apply the penalty to the solution which violates the constraints. Thus automatically the fitness of these chromosomes are lowered. Many penalty methods are introduced and suggested for different problems. Here we apply the dynamic penalty as follow:

$$Obj.Function:\ f(\vec{x})\ \ ,Constrained\ to: g_i(\vec{x}) \leq 0\ , i = 1,\ldots, q \tag{2}$$

$$F(\vec{x}) = f(\vec{x}) \pm p(\vec{x}) \tag{3}$$

$$p(\vec{x}) = \alpha(\#iteration)^{\beta} \times \sum_{i=0}^{q} S_i(\vec{x})\ in\ which\ S_i(\vec{x}) = \begin{cases} 0 & g_i(\vec{x}) \leq 0 \\ \left|g_i(\vec{x})\right| & otherwise \end{cases} \tag{4}$$

The sensitivity of the problem to this penalty method is presented in Eqs. (1- 3) and its parameters have direct impact on the final result and are all beyond the scope of this research. Dynamic penalty enables algorithm to have a smooth rate of reaching to the global optimum. A wise tuning of the penalty parameters will tend the algorithm to find the global optimum and preferably search areas nearer to the boundaries of constraints where the validity of the solution is acceptable and the global point also lay there. In Figure 4 two convergence histories for a typical test case are included, one with high penalty orders and the other one with low penalty orders for coefficient in Eq. (3). Knowing the physics of the constraints enables us to adopt proper values, hence converging to more reliable results.

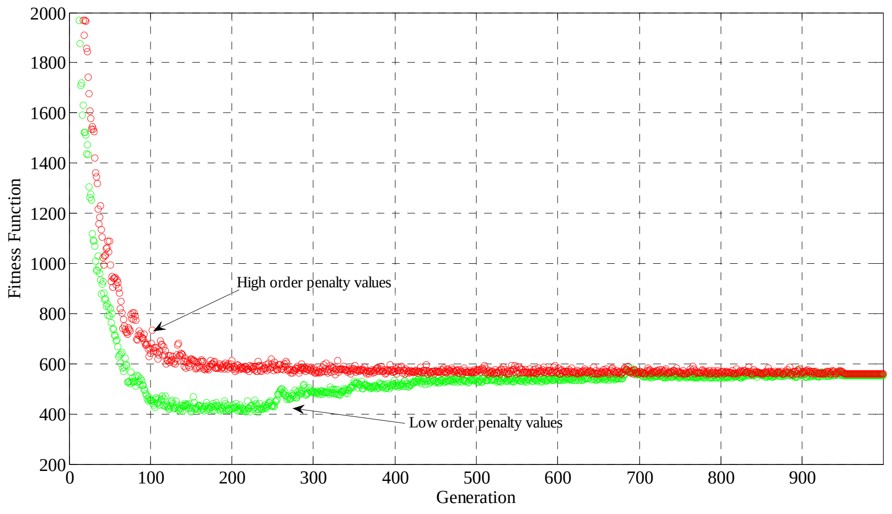

Figure 4. Comparison between the convegence history of penalty methods

4. NUMERICAL EXAMPLES

4.1. Introduction to structure optimization and dome and large-scale structures

To demonstrate the efficiency of this algorithm, weight minimization of structures is chosen as the test cases. These numerical examples are among those chosen in the literature to show the robustness of different algorithms. The nonlinearity of the problem, number of variables and constraints are the factors that determine the complexity of such problems. The test case problems are chosen in a variety of variables, ranging from 29~200 which are holding a number of constraints up to 3500. In all cases, the presented algorithm overcomes its rivals. For each test case the reported results in the literature are also provided.

From the analytical point of view the weight minimization problem of a trusses structure can be put in the following form:

Fitness function:

$$W(\vec{A}) = \rho g \sum_{j=1}^{N_l} A_j l_j \tag{5}$$

Constrained to:

$$u^l_{(x,y,z),k} \le u_{(x,y,z),k,ilc} \le u^u_{(x,y,z),k} \begin{cases} j = 1, N_e \\ k = 1, N_n \\ ilc = 1, N_c \end{cases} \& \quad \sigma^l_j \le \sigma_{j,ilc} \le \sigma^u_j \quad \& \quad A^l_j \le A_j \le A^u_j \tag{6}$$

Where:

- N_e is the number of elements in the structure;
- N_n is the total number of nodes;
- N_c is the number of independent loading conditions acting on the structure;

- g is the gravity acceleration value (9.81 m/s^2);
- ρ is the material density;
- $\vec{A} = A(A_1, A_1, A_1, \ldots, A_N)$ is the vector of the cross sectional areas values; A_j is the area of the *jth* element of the structure. A_j^l and A_j^u are lower and upper bounds of A_j;
- l_j is the length of *jth* element, which is:

$$l_j = \sqrt{(x_{j1} - x_{j2})^2 + (y_{j1} - y_{j2})^2 + (z_{j1} - z_{j2})^2} \tag{7}$$

Where $x_{j1,2}, y_{j1,2}, z_{j1,2}$ are the nodal coordinates of the *jth* element.

$u_{(x,y,z),k,ilc}$ are the displacements of the *kth* node in the directions x, y, z with the lower and upper limits $u^l_{(x,y,z),k}$ and $u^u_{(x,y,z),k}$ in corresponding of the *ilcth* loading condition;

- $\sigma_{j,ilc}$ is the stress on the *jth* element, with the lower and upper limits σ_k^l and σ_k^u, in correspondence of the *ilcth* loading condition.

4.2. Large-scale truss optimization

Three numerical examples chosen from size optimum design of large-scale truss structures are studied to test and verify efficiency of the GA-NMS algorithm, as well as to illustrate its applicability for optimum design of practical structures. The studied trusses consist of a 200 bar truss and the 942 bar truss. These benchmark problems are also studied by different researchers including Farshi and Ziazi[17],Venkaya et al [18], Adeli and Cheng [19], Erbatur and Hasancebi [20]. The general features of these trusses are listed in Table 2.

Table 2. General properties of large-scale trusses

Property / type of the truss	**200-bar truss**	**942-bar truss**
Number of variables	29-200	59
Minimum cross section	0.1 *in*^{2}	1 *in*^{2}
Material density	0.283 $\frac{\text{1b}}{\text{in}^2}$	0.1 $\frac{\text{1b}}{\text{in}^2}$
Modulus of elasticity	30000 *ksi*	10000 *ksi*

4.2.1. A 200-bar truss

Due to the symmetry of the structure, variables linking are adopted and the areas are grouped into twenty-nine groups and 29 optimization variables are considered according to Figure 5. The following three independent loading conditions are imposed on the structure:

(a) 1000 lbf in positive x-direction at nodes 1,6,15,20,29,34,43,48,57,62 and 71;

(b) 10000 lbf in negative y-direction at nodes 1, 2, 3,4,5,6,8,10,

12,14,15,16,17,18,19,20,22,24,26,28,29,30,31,32,33,34,36,38,40,42,43,
44,45,46,47,48,50,52,54,56,58,59,60,61,62,64,66,68,70,71,72,73,74,75;

(c) The loading conditions (a) and (b) acting together;
Table 3 presents the optimal results for this case study.

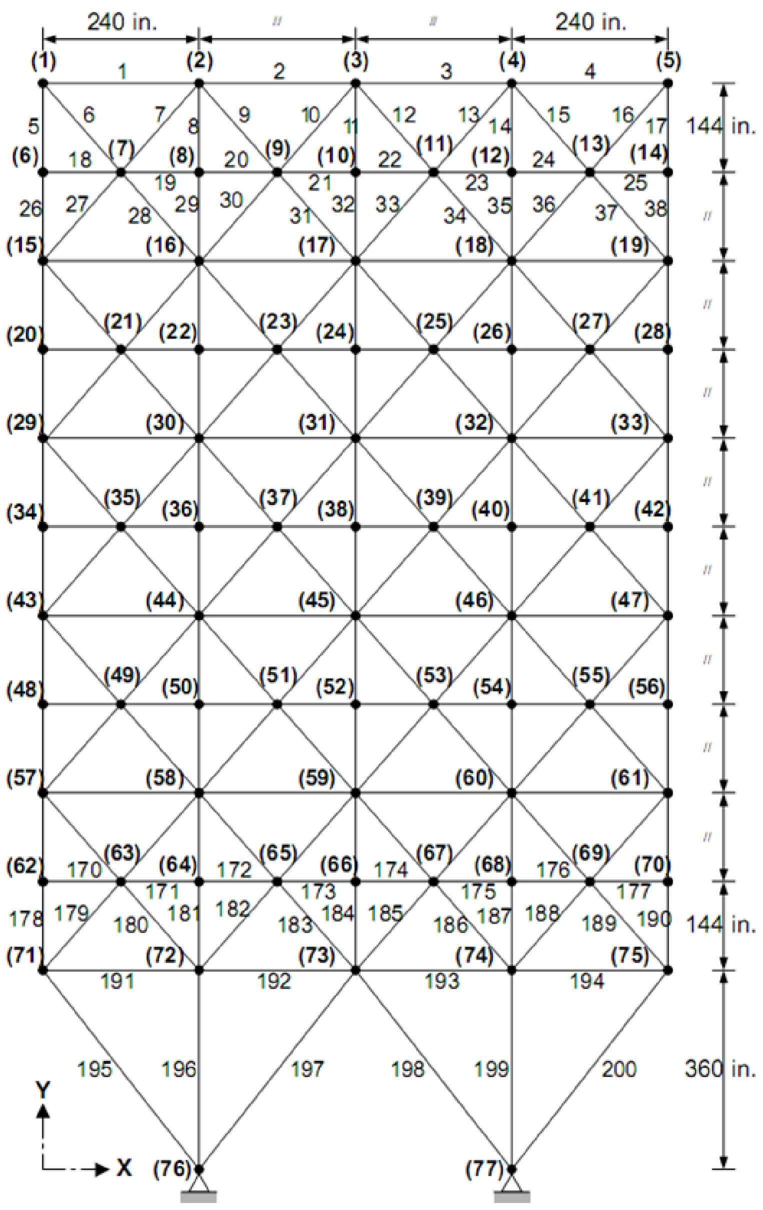

Figure 5. Two hundred-bar truss scheme

Table 3. Optimal designs of the 200-bar truss

Variables	Farshi and Ziazi [17]	This research
A_1(in.^{2})	0.147	0.147
A_2	0.945	0.935
A_3	0.1	0.1051
A_4	0.1	0.1062
A_5	1.9451	1.9445
A_6	0.2969	0.2982
A_7	0.1	0.1
A_8	3.1062	3.1064
A_9	0.1	0.1092
A_{10}	4.1052	4.1089
A_{11}	0.4039	0.4031
A_{12}	0.1934	0.1957
A_{13}	5.4289	5.4201
A_{14}	0.1	0.1
A_{15}	6.4289	6.4252
A_{16}	0.5745	0.5776
A_{17}	0.1339	0.1336
A_{18}	7.9737	7.9782
A_{19}	0.1	0.1
A_{20}	8.9737	8.9637
A_{21}	0.7053	0.7002
A_{22}	0.4215	0.4258
A_{23}	10.8675	10.8594
A_{24}	0.1	0.1
A_{25}	11.8674	11.8636
A_{26}	1.0349	1.029
A_{27}	6.6849	6.6801
A_{28}	10.8101	10.8158
A_{29}	13.8379	13.829
Weight (lb)	25456.57	25449.27

4.2.2. The 200 bar space truss structure with five independent loading conditions

The planar 200-bar truss structure shown in Figure 5 can be optimized also with 200 design variables by assigning a sizing variable to the cross-sectional of each element. The structure is designed to carry five independent loading conditions. Besides the three load cases listed in Section 4.2.1, the structure is also loaded by

(d) 1000 lbf acting in the negative x-direction at node points 5,14,19,28,33,42,47,56,61,70 and 75;

(e) Loading conditions (b)–described in Section 4.2.1.1 – and (d) acting together.

The optimization includes 3500 non-linear constraints on nodal displacements and member stresses. The displacements of all free nodes in both directions x and y must be less than ±0.5in. The allowable stress (the same in tension and compression) is 30,000psi. The lower bound of cross-sectional areas is 0.1 in^2.

Table 4 shows the results and a comparison of the available results in the literature.

Table 4. Optimal designs of the 200-bar truss with five loading conditions (Areas of elements on one side of symmetrical line)

Variables	Venkaya et al[18]	This Work	Variables	Venkaya et al[18]	This Work	Variables	Venkaya et al[18]	This Work	Variables	Venkaya et al[18]	This Work	Variables	Venkaya et al[18]	This Work
A_1	1.348	1.4067	A_{40}	0.233	0.1	A_{81}	5.737	5.7218	A_{120}	2.558	2.5469	A_{159}	0.21	0.1
A_2	1.313	1.1623	A_{43}	4.798	4.849	A_{82}	1.988	2.0443	A_{121}	0.237	0.2525	A_{160}	14.981	14.9598
A_5	3.402	3.4181	A_{44}	1.85	1.8053	A_{83}	0.201	0.1542	A_{122}	10.649	10.997	A_{161}	1.175	1.2812
A_6	1.771	1.73	A_{45}	0.127	0.1	A_{84}	7.22	7.4849	A_{123}	0.966	0.7787	A_{162}	1.251	1.1015
A_7	0.173	0.1328	A_{46}	4.318	4.2611	A_{85}	0.984	1.0679	A_{124}	0.991	1.1342	A_{163}	9.8	9.7349
A_8	1.497	1.5895	A_{47}	0.971	0.7747	A_{86}	0.797	0.7845	A_{125}	7.822	7.6835	A_{170}	0.116	0.3897
A_9	0.742	0.4548	A_{48}	0.749	0.6885	A_{87}	5.626	5.6498	A_{132}	0.116	0.1	A_{171}	0.816	0.4938
A_{10}	0.782	0.8027	A_{49}	3.346	3.162	A_{94}	0.116	0.1	A_{133}	0.634	0.4905	A_{172}	0.816	0.789
A_{11}	1.156	0.9263	A_{56}	0.116	0.2195	A_{95}	0.491	0.7661	A_{134}	0.634	0.3957	A_{173}	0.703	0.6152
A_{12}	0.116	0.3031	A_{57}	0.333	0.3364	A_{96}	0.491	0.652	A_{135}	0.512	0.6612	A_{178}	6.713	6.874
A_{13}	0.377	0.2921	A_{58}	0.333	0.1501	A_{97}	0.318	0.4281	A_{140}	7.285	7.4151	A_{179}	0.713	0.8576
A_{14}	0.377	0.387	A_{59}	0.208	0.36	A_{102}	6.688	6.5888	A_{141}	0.587	0.4113	A_{180}	4.281	4.1989
A_{15}	0.435	0.3559	A_{64}	5.662	5.605	A_{103}	0.533	0.2409	A_{142}	2.835	2.9147	A_{181}	16.104	16.187
A_{16}	4.575	4.713	A_{65}	0.519	0.55	A_{104}	2.151	2.2602	A_{143}	11.752	11.5395	A_{182}	1.309	1.3174
A_{17}	0.538	0.5823	A_{66}	1.95	1.8351	A_{105}	8.288	8.4646	A_{144}	1.049	0.8999	A_{183}	1.317	1.3123
A_{18}	1.895	1.9992	A_{67}	5.326	5.3958	A_{106}	0.884	0.567	A_{145}	1.011	0.921	A_{184}	10.95	11.2053
A_{19}	2.483	2.5937	A_{68}	0.813	0.832	A_{107}	0.984	1.0772	A_{146}	8.969	8.9478	A_{191}	5.073	4.8779
A_{20}	0.75	0.571	A_{69}	0.954	1.084	A_{108}	6.77	6.6058	A_{153}	2.495	2.439	A_{192}	3.243	3.3121
A_{31}	0.784	0.4193	A_{70}	4.495	4.7932	A_{115}	1.687	1.7259	A_{154}	1.024	1.0722	A_{195}	8.983	9.017
A_{32}	2.278	2.5342	A_{77}	1.391	1.2508	A_{116}	0.605	0.3831	A_{157}	5.695	5.7271	A_{196}	20.687	20.7437
A_{33}	1.294	1.2493	A_{78}	0.343	0.1197	A_{119}	6.274	6.4366	A_{158}	3.932	3.8349	A_{197}	9.594	9.6298
												Weight(lb)	**31020**	**30849**

4.2.3. A 942-bar truss

The 26-story tower space truss containing 942 elements and 244 nodes is considered in this section. 59 design variables are used regarding the symmetry of the structure. Figure 6 shows the geometry and the 59 element groups. The members are subjected to the stress limits of ±25 ksi (±172.375 MPa) and the four nodes of the top level in the x, y and z directions are subjected to the displacement limits of ±15.0in (±38.10cm). The allowable cross-sectional areas in this example are selected from 0.1 to 200in^2. The loading on the structure consists of:

(1) The vertical load at each node in the first section (by section we mean a storey of the structure) is equal to 3 kips**. (2)** The vertical load at each node in the second section is equal to 6 kips. **(3)** The vertical load at each node in the third section is equal to 9 kips. **(4)** The horizontal load at each node on the right side in the x direction is equal to 1 kips. **(5)** The horizontal load at each node on the left side in the x direction is equal to 1.5 kips. **(6)** The horizontal load at each node on the front side in the y direction is equal to 1 kips. **(7)** The horizontal load at each node on the back side in the y direction is equal to 1 kips.

Table 5 lists the available studies on this case and the results for this algorithm. It is obvious that like other cases the results from this algorithm outperforms its rivals.

Table 5. Optimal design of 942-bar tower space truss

Variables	**Adeli and Cheng [19]**	**Erbatur and Hasancebi [20]**	**This research**
A_1 (in^2.)	Not Available	1.00	2.7859
A_2	Not Available	1.00	1.3572
A_3	Not Available	3.00	5.0362
A_4	Not Available	1.00	2.2398
A_5	Not Available	1.00	1.2226
A_6	Not Available	17.00	14.9575
A_7	Not Available	3.00	2.9568
A_8	Not Available	7.00	10.9038
A_9	Not Available	20.00	14.4177
A_{10}	Not Available	1.00	3.7090
A_{11}	Not Available	8.00	5.7076
A_{12}	Not Available	7.00	4.9264
A_{13}	Not Available	19.00	14.1751
A_{14}	Not Available	2.00	1.9043
A_{15}	Not Available	5.00	2.8101
A_{16}	Not Available	1.00	1.0000
A_{17}	Not Available	22.00	18.8070
A_{18}	Not Available	3.00	2.6151
A_{19}	Not Available	9.00	12.5328
A_{20}	Not Available	1.00	1.1314
A_{21}	Not Available	34.00	30.5122
A_{22}	Not Available	3.00	3.3460
A_{23}	Not Available	19.00	17.0450
A_{24}	Not Available	27.00	18.0785
A_{25}	Not Available	42.00	39.2717
A_{26}	Not Available	1.00	2.6062
A_{27}	Not Available	12.00	9.8303
A_{28}	Not Available	16.00	13.1126
A_{29}	Not Available	19.00	13.6897
A_{30}	Not Available	14.00	16.9776

Variables	**Adeli and Cheng [19]**	**Erbatur and Hasancebi [20]**	**This research**
A_{31}	Not Available	42.00	37.6006
A_{32}	Not Available	4.00	3.0602
A_{33}	Not Available	4.00	5.5106
A_{34}	Not Available	4.00	1.8014
A_{35}	Not Available	1.00	1.1568
A_{36}	Not Available	1.00	1.2423
A_{37}	Not Available	62.00	62.7741
A_{38}	Not Available	3.00	3.3276
A_{39}	Not Available	2.00	4.2369
A_{40}	Not Available	4.00	1.7202
A_{41}	Not Available	1.00	1.0148
A_{42}	Not Available	2.00	5.6428
A_{43}	Not Available	77.00	78.0094
A_{44}	Not Available	3.00	3.2206
A_{45}	Not Available	2.00	3.5934
A_{46}	Not Available	3.00	4.7668
A_{47}	Not Available	2.00	1.1531
A_{48}	Not Available	3.00	2.1698
A_{49}	Not Available	100.00	99.6406
A_{50}	Not Available	4.00	4.1469
A_{51}	Not Available	1.00	2.1600
A_{52}	Not Available	4.00	4.1499
A_{53}	Not Available	6.00	11.2070
A_{54}	Not Available	3.00	11.0904
A_{55}	Not Available	49.00	35.9499
A_{56}	Not Available	1.00	2.1937
A_{57}	Not Available	62.00	66.1705
A_{58}	Not Available	1.00	3.3402
A_{59}	Not Available	3	4.0525
Weight (lb)	170000	143436	142295.75

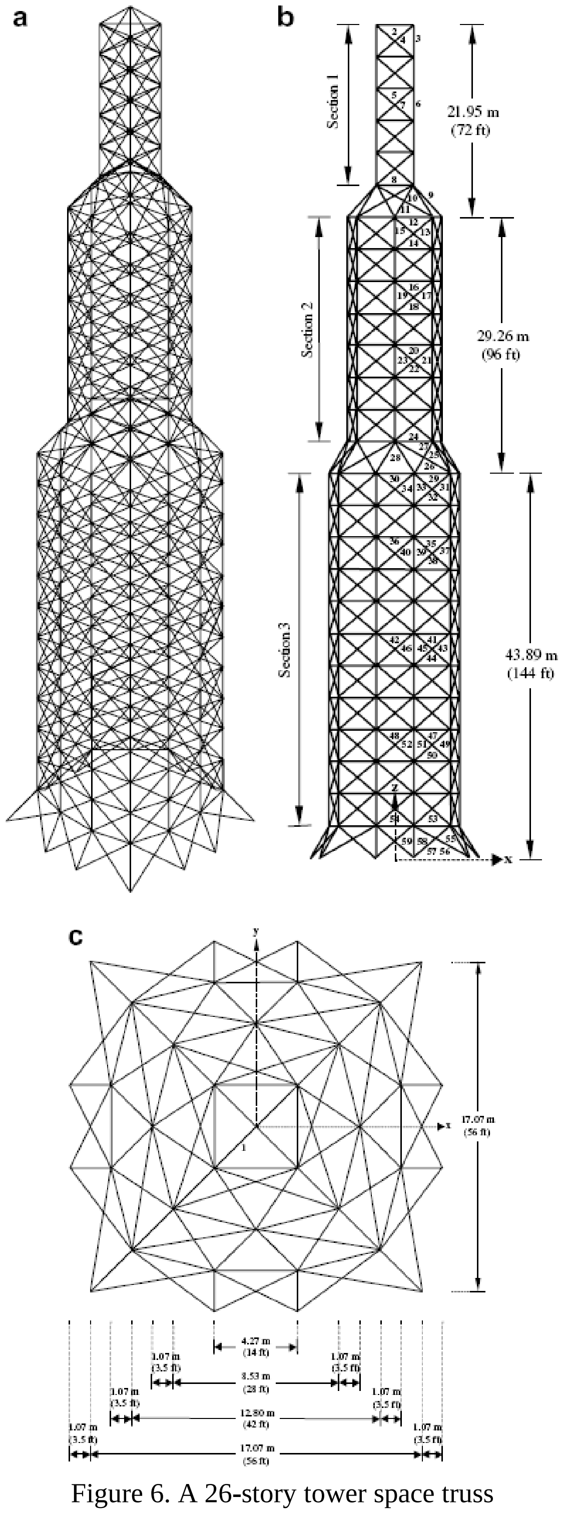

Figure 6. A 26-story tower space truss

5. CONCLUSION

In this paper a hybrid algorithm is presented for finding the global minimum of the objective function for both constrained and unconstrained problems. A controlled mixture of main characteristics of two different algorithms (exploration of Genetic algorithm and exploitation of Nelder-Mead simplex) led to a new algorithm which has promising results in the challenging field of structural optimization. Moreover some modifications are implemented on the GA operators to improve its convergence rate and reaching better results. Optimization costs in terms of CPU usage or convergence time decreased since simple-coded algorithm like Nelder-Mead simplex were used in hybridizing with modified GA.

Acknowledgement: The first author is grateful to the Iran National Science Foundation for the support.

REFERENCES

1. Fan S-KS, Zahara E. A hybrid simplex search and particle swarm optimization for unconstrained optimization, *Eur J Oper Res* 2007; **181**: 527–48.
2. Ren Z-W, San Y, Chen J-F. Hybrid Simplex-improved Genetic Algorithm for Global Numerical Optimization, *Acta Automat* 2007; **33**(1): 91-5.
3. Chelouah R, Siarry P**.** A hybrid method combining continuous tabu search and Nelder–Mead simplex algorithms for the global optimization of multiminima functions, *Eur J Oper Res* 2005; **161**(3): 636-54.
4. Nelder JA, Mead R. A Simplex method for function minimization, *Comput J* 1965 ;7: 308–13.
5. Olsson DM, Nelson LS. The Nelder–Mead simplex procedure for function minimization, *Technometrics* 1975; **17**: 45–51.
6. Chen DH, Saleem Z, Grace DW. A new simplex procedure for function minimization, *Int J Model Simul* 1986; **6**: 81–5.
7. Spendley W, Hext GR, Himsworth FR. Sequential application of simplex designs in optimization and evolutionary operation, *Technometrics* 1962; **4**: 441–61.
8. Arora JS. *Introduction to Optimum Design*, Second Edition, The University of Iowa, Elsevier Academic Press, 2004.
9. Balakrishnan J, Gunasekaran MK, Gopal ESR Critical dielectric – constant measurements in the binary liquid system – methanol + normal heptane, *Inter J PA Physics* 1984; **22**: 286–98.
10. Busing WR, Matsui M. The application of external forces to computational model of crystals, *Acta Crystallogr A* 1984; **40**: 532–40.
11. Schulze W, Rehder U. Organization and morphogenesis of the human seminiferous epithelium, *Cell Tissue Rev* 1984; **237**: 395–417.
12. Silver GL. Space modification: An alternative approach to chemistry problem involving geometry, *J Comput Chem* 1981;**2**: 478–90.

13. Sthapit PR, Ottoway JM, Fell GS. Determination of lead in matural and tap waters by flame atomic-fluorescence spectrometry, *Analyst* 1984; **109**: 1061–75.
14. Fletcher R. *Practical Methods of Optimization*, John Wiley & Sons, Chichester, 1987.
15. Barton RR, Ivey JS. Nelder–Mead simplex modifications for simulation optimization, *Manage Sci* 1996; **42**: 954-73.
16. Nazareth L, Tzeng P. Gilding the lily: A variant of the Nelder–Mead algorithm based on golden-section search, *Comput Optim Appl* 2002; **22**: 133–4.
17. Farshi B, Ziazi A. Sizing optimization of truss structures by method of centers and force formulation, *Int J Solids Struct* 2010;**47**: 2508–24.
18. Venkayya VB, Khot NS, Reddy VS. Optimization of Structures Based on the Study of Energy Distribution, Proceedings of the Second Conference on Matrix Methods in Structural Mechanics, AFFDL TR-68-150, NTIS No. AD-703-685, 11 1-153 (December 1969).
19. Adeli H, Cheng N-T. Concurrent genetic algorithms for optimization of large structures. *J Aerospace Eng* 1994; **7**: 276–96.
20. Erbatur F, Hasancebi O. Optimal design of planar and space structures with genetic algorithms. *Comput Struct* 2000;**75**: 209–24.