Misan Journal of Engineering Sciences ISSN-E: 2957-4250
Vol. 1, No. 2, December 2022 ISSN: 2957-4242

A Hybrid Stochastic Algorithm with Domain Reduction for Discrete Structural Optimization

Mustafa Al-Bazoon*[1], Jasbir S. Arora[2]

[1]Department of Civil Engineering, The University of Misan, Amarah, Maysan 62001, Iraq
[2]Iowa Technology Institute, The University of Iowa, Iowa City, IA 52242, USA
**Corresponding author; E-mail: mustafa-jasim@uomisan.edu.iq*

(Received 3 Feb , Revised 12 June, Accepted 7 Sep)

Abstract: In recent years, many nature-inspired metaheuristic optimization algorithms have been proposed in an effort to develop efficient and robust algorithms. The drawback in most of them is the large number of simulations required to obtain good designs. To reduce the number of structural analyses to reach the best design, a new two-phase algorithm is proposed and evaluated. This hybrid algorithm is based on the well-known Harmony Search (HS) algorithm and recently developed Colliding Bodied Optimization (CBO). HS analyzes and improves one design in every iteration whereas CBO generates and analyzes a new population of designs in every iteration. Based on the observed behavior of these two algorithms, a Hybrid Harmony Search - Colliding Bodies Optimization (HHC) is proposed. The first phase of HHC uses the Improved Harmony Search (IHS) algorithm. A new design domain reduction technique is also incorporated in IHS that dramatically reduces the number of possible combinations of discrete variables. This improves the performance of the IHS algorithm. The second phase uses the Enhanced Colliding Bodies Optimization (ECBO). ECBO receives final designs from the first phase to enhance them further. This makes the second phase need fewer iterations in comparison with the ECBO alone. The performance of the proposed algorithms is evaluated using some benchmark discrete structural optimization problems, although the method is applicable to continuous-variable problems as well. The results show HHC with design domain reduction to be quite effective, robust, and needs a smaller number of structural analyses to solve optimization problems in comparison with IHS, ECBO, and some other metaheuristic optimization algorithms. HHC with design domain reduction is shown to be quite robust in the sense that different runs for a problem obtain the same final design. In comparison with HIS and ECBO, HHCD reduces the number of structural analyses to find the best design to less than half. This is an important feature that leads to better confidence in the final solution from a single run of the algorithm for a problem.

Keywords: Hybrid metaheuristic algorithm; Global optimization; Discrete structural optimization; Harmony Search; Colliding Bodies Optimization.

1. Introduction

Calculus-based optimization algorithms were developed more than 50 years ago and a vast amount of literature is available on the subject. Linear programming (LP), nonlinear programming (NLP), and dynamic programming (DP) methods need gradient information to improve the solution estimate (to find a search direction). These methods search for the optimum point in a neighborhood of the current estimate. In comparison with metaheuristic algorithms, these methods converge much faster and can find higher-accuracy solutions.

Gradient-based methods are most appropriate for continuous variables and continuous functions. Many engineering problems have non-smooth functions in their formulation. For example, designing steel frames according to American Institute of Steel Construction codes' requirements imposes some non-differentiable equations (empirical equations based on experimental studies). As a result, gradient-based optimization methods cannot be used to solve such problems. On the other hand, stochastic, metaheuristic, or nature-inspired algorithms use only simulation results to reach the final solution, such as the well-known Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and many others [1]. The search is not near the current point and the discrete variables and non-differentiable functions can be treated routinely. They use an organized random search in the entire design space instead of a gradient-based search in the neighborhood of the current point.

Therefore, they are likely to converge to a global optimum point rather than a local optimum. Different runs for the same problem can take different paths to the final solution or even a different solution. The methods are suitable for both continuous and discrete variables and with one or more objective functions.

Just like gradient-based algorithms, stochastic algorithms have drawbacks. They require considerable computation time to obtain a reasonable solution. The computation time depends on the number of the design variables and the range for each design variable; a larger design domain needs more iterations and thus more structural analyses. Increasing the number of iterations could be a good way to find a better design but there is no guarantee that a global optimum design will be found. Therefore, the best way is to run the algorithm more than once and choose the best solution from different runs. This, however, means that more computational effort is needed to solve a problem. A good metaheuristic algorithm has the ability to skip local optima, needs less number of simulations to find the best design, is applicable to different types of problems, and can obtain higher accuracy solutions [2].

In an effort to reduce the number of structural analyses to reach the final design, a Hybrid Harmony Search - Colliding Bodies Optimization (HHC) algorithm is proposed and evaluated in this study. This proposal is based on the following observations about the behavior of two algorithms while solving some structural design problems: (1) Improved version of the harmony search algorithm (IHS) [3] makes rapid improvements toward the final design in the initial iterations and then its progress slows down once it is in a neighborhood of the best design, and (2) The enhanced version of the colliding bodies optimization (ECBO) makes steady improvement towards the final design requiring more structural analyses to reach a neighborhood of the final design compared to IHS. Therefore, the basic idea to be explored for the proposed hybrid algorithm is to determine if a combination of the two algorithms can reduce the number of structural analyses to reach the final design. That is, since IHS algorithm can reach the neighborhood of the final design more rapidly, it will be used in phase one and its iterations will be terminated once the progress towards the final design slows down; in phase two, the improved designs from IHS will be passed on to the ECBO as its initial population (instead of random designs generated from the entire design domain) to improve the best design further. This may lead to fewer structural analyses to reach the final design. In addition, a new design domain reduction technique based on statistically analyzing some designs is added to IHS to increase the possibility of rapidly finding better designs.

A new stopping criterion is also introduced in addition to a limit on the number of iterations for terminating phase one iterations. That is, when the algorithm is not able to find a better design for a certain number of iterations, it is terminated.

A major motivation for this work is to investigate procedures that can reduce the number of structural analyses to reach the final designs for the class of structural optimization problems that cannot be solved using gradient-based algorithms. This becomes critically important while solving more complex structural optimization applications, such as nonlinear static response problems, nonlinear dynamic response problems and multidisciplinary problems. Each simulation of such problems can take enormous computational effort making meta-heuristic methods very time-consuming.

Some benchmark discrete truss optimization problems are solved using the proposed algorithm. These well-known examples are solved previously in the literature using different metaheuristic algorithms. The results are discussed and compared with the available results in the literature to study the performance of the proposed algorithm.

2. Formulation of Discrete Structural Optimization Problems

In many practical design cases, design variables are discrete because members must be selected from the available sizes in a catalog. The formulation of the discrete design variables optimization problem is slightly different from continuous design variable optimization. In general, the problem can be stated as:

$$Find \quad \boldsymbol{X} = [x_1, x_2, \dots, x_{nvar}]; \quad x_j \in D_j; \quad j = 1,2, \dots, nvar \tag{1}$$

$$\text{to minimize} \quad f(\boldsymbol{X}) \tag{2}$$

$$\text{subject to} \quad g_k(\boldsymbol{X}) \leq 0; \quad k = 1,2, \dots, p \tag{3}$$

where X is the vector of design variables with nvar unknowns, D_j is a set of discrete values for the jth design variable, f(X) is a cost function (in this study, f(X) is the total weight of the structure), and $g_k(X)$ is a constraint function.

One way of treating constraints in metaheuristic algorithms is to combine constraints with the cost

function to define a merit function (also called the penalty function) that is then minimized:

$$F(\boldsymbol{X}) = f(\boldsymbol{X})[1 + \psi G(\boldsymbol{X})]^{\xi} \quad (4)$$

$$G(\boldsymbol{X}) = \sum_{k=1}^{p} max\,(0, g_k(\boldsymbol{X})) \quad (5)$$

where $\mathrm{G(X)}$ is a constraint violation function, $\psi \geq 1$ is exploration penalty coefficient ($\psi = 1$ unless another value is mentioned), $\xi > 1$ is penalty function exponent (in this study, $\xi = 2$), and $\max\,(0, \mathrm{g_k(X)}) \geq 0$ is the violation value of the kth inequality constraint. The present problem has just inequality constraints. However, if equality constraints are present in the problem formulation, they are treated by including their violations in Eq. (5).

3. Metaheuristic Optimization Algorithms

Over the years, many metaheuristic optimization algorithms have been explored. More recent techniques are based on observations about some natural phenomena, such as survival of the fittest and genetic inheritance in Genetic Algorithms (GA), which is inspired by the basic mechanism of natural evolution developed by Goldberg and Holland [4]; Simulated Annealing (SA) proposed by Kirkpatrick et al. [5]; Particle Swarm Optimization (PSO) proposed by Kennedy and Eberhart [6]; Ant Colony Optimization (ACO) introduced by Dorigo et al. [7]; Harmony Search (HS) algorithm invented by Geem et al. [8]; Big Bang–Big Crunch algorithm (BB–BC) introduced by Erol and Eksin [9]; Colliding Bodies Optimization (CBO) proposed by Kaveh and Mahdavi [10]; and Ray Optimization (RO), developed by Kaveh and Khayat [11].

In this study, HS algorithm and its improved version and CBO and its enhanced version are summarized since the proposed hybrid algorithm HHC uses these procedures.

3.1 Harmony Search Algorithm

Geem, Kim, and Longanathan [8] presented the HS algorithm based on the music improvisation process of jazz musicians. The following five steps describe the HS algorithm:

Step 1: Parameter setting

The algorithm initially generates a set of random designs from the design domain. Then in every iteration, a new design is generated and analyzed. If this design is better than the current population's worst design, it replaces that design; otherwise, another design is generated. The process is continued until a limit on the number of iterations is reached.

HS has four parameters that need to be initialized before starting the algorithm. There are no general guidelines for their selection; they are selected depending on the problem [12]. Thus, the best way is to try different values to find the best combination for an application. The parameters are:

Harmony memory size (HMS). It is the initial number of candidate solutions selected randomly from the design domain. For example, if HMS is 10, the algorithm starts by selecting 10 designs and for every design evaluates the merit function F if the problem is constrained or the objective function f if the problem is unconstrained. This information is saved in a matrix called harmony memory (HM).

Harmony memory consideration ratio (HMCR): Its value ranges between 0 and 1. It is the probability of selecting design variables from the current HM to generate a new design. Variables selected from the current HM may go through further adjustment depending on the pitch adjustment rate.

Pitch adjusting rate (PAR): Its value ranges between 0 and 1 and it is the probability of mutation of the design variable selected from HM to a neighboring value.

Maximum improvisations ($\mathrm{MaxIer_{P1}}$): It is a limit on the number of iterations for HS.

Step 2: Initialization

The HS starts with HMS random designs to populate the harmony memory matrix HM as:

$$\mathbf{HM} = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^{nvar} \\ \vdots & & \ddots & \vdots \\ x_{HMS}^1 & x_{\mathrm{HMS}}^2 & \cdots & x_{HMS}^{nvar} \end{bmatrix} \quad (6)$$

where nvar is the number of design variables. Thus each row of this matrix represents a design point

and each column is associated with a design variable.

Step 3: Harmony improvisation

A new design point is improvised where each design variable is selected from either the current population of designs in the matrix HM or from its possible range of values. These selections are based on harmony memory consideration, pitch adjustment, and random numbers.

Using harmony memory consideration parameter HMCR, the new value for the jth design variable is chosen as x_i^j from either the jth column of HM or from the allowable values for this design variable [12]. For each design variable j (j = 1 to nvar), the row index i is selected randomly as follows:

$$x_i^j \in \{x_1^j, x_2^j, \dots, x_{HMS}^j\}; \quad if\ rn_{HMCR}^j \le HMCR \tag{7}$$

$$x_i^j \in D_j; \quad if\ rn_{HMCR}^j > HMCR \tag{8}$$

where rn_{HMCR}^j is a random number uniformly distributed over the interval [0,1] and D_j is the allowable set of values for the jth design variable.

Every design variable selected from harmony memory is examined further to determine whether it should be pitch-adjusted or not. The parameter PAR is used for this purpose as follows:

$$\text{Pitch adjusting decision for } x_i^j \begin{cases} \text{yes} & \text{if } rn_{PAR}^j \le PAR \\ \text{No} & \text{if } rn_{PAR}^j > PAR \end{cases} \tag{9}$$

where rn_{PAR}^j is a random number uniformly distributed over the interval [0,1]. If the pitch adjustment decision is "yes" x_i^j is replaced as follows:

$$\begin{aligned} x_{i,new}^j &= x_i^j + 1 \ \ if\ PAR_{rand}^j < 0.5 \\ x_{i,new}^j &= x_i^j - 1 \ \ if\ PAR_{rand}^j \ge 0.5 \end{aligned} \tag{10}$$

where PAR_{rand}^j is a random number uniformly distributed over the interval [0,1], and +1 and -1 mean moving to the next higher or lower allowable value for this variable [13].

Step 4: Update the harmony memory:

The new design from step 3 is evaluated. If it is better than the worst design in HM, the new design replaces the worst design in HM; otherwise, a new design is improvised.

Step 5: Termination criteria

If the limit on the number of iterations is reached, terminate the algorithm; otherwise, go to step 3.

3.2 Improved Harmony Search Algorithm (IHS)

The concept of IHS is the same as HS (the five steps in Section 3.1). However, the standard HS algorithm uses fixed values of HMCR and PAR. The main drawback of the standard HS algorithm is that it needs a large number of iterations to find an acceptable solution [3].

In IHS, HMCR and PAR are adjusted with every iteration using Eqs. (11) and (12) to improve the performance of the HS algorithm by eliminating its drawbacks [14].

$$HMCR(iter) = HMCR_{max} - \frac{(HMCR_{max} - HMCR_{min})}{MaxIer_{P1}} \times Iter_{P1} \tag{11}$$

$$PAR(iter) = \frac{(PAR_{max} - PAR_{min})}{\pi/2} \times arctan(Iter_{P1}) + PAR_{min} \tag{12}$$

where $Iter_{P1}$ is the current iteration, $HMCR_{max}$ and $HMCR_{min}$ are maximum and minimum harmony memory consideration ratios, respectively, PAR_{max} and PAR_{min} are maximum and minimum pitch adjacent ratios, respectively. Note that HMCR is a linearly decreasing function of iteration number. This increases the probability of selecting a design variable from its allowable range of values rather than from the harmony memory. Also, PAR is an increasing function of the iteration number that increases the probability of pitch adjustment for a design variable when it is selected from the HM matrix. These processes introduce more diversity into the population of design in the HM matrix.

Just like HS, there are no guidelines that one can follow to select IHS parameters. Therefore, the best way is to start with a set of values then try different values to find the best combination. In this study, $HMCR_{max}$ and PAR_{max} of 0.85 and $HMCR_{min}$ and PAR_{min} of 0.35 show good performance.

3.3 Colliding Bodies Optimization (CBO)

3.3.1 Background Material

Kaveh and Mahdavi [10] developed this metaheuristic algorithm that is inspired by the laws of one-dimensional collision. The algorithm works with a population of designs at each iteration. Here each design in the population is considered as an object or body with mass and velocity

Using the laws of momentum and energy, a collision can be simulated between objects such as two balls in a billiard game or two cars in an accident. If there are no external forces acting on the system, the momentum of all objects before the collision equals the momentum of all objects after the collision. Conservation of linear momentum of two bodies in a one-dimensional collision is expressed as:

$$m_1 v_1 + m_2 v_2 = m_1 v_1' + m_2 v_2' \tag{13}$$

where m_1, v_1, and v_1' are mass, initial velocity and final velocity of the first object, respectively, and m_2, v_2, andv_2' are mass, initial velocity and final velocity of the second object, respectively.

For one-dimensional collision, let body 1 approach and collide with body 2; therefore $v_1 > v_2$. After the collision, the bodies separate; therefore $v_2' > v_1'$. The system loses some of its energy during the collision. The Coefficient of Restitution (COR) $\varepsilon \geq 0$ indicates how much kinetic energy remains in the system after collision that is defined as:

$$\varepsilon = \frac{\text{Velocity of separation after collision}}{\text{Velocity of approach before collision}} = \frac{v_2' - v_1'}{v_1 - v_2} \tag{14}$$

Using Eqs. (13) and (14) the velocities after collision are calculated as follows [10]:

$$v_1' = \frac{(m_1 - \varepsilon\, m_2) v_1 + (1 + \varepsilon)\, m_2 v_2}{m_1 + m_2} \tag{15}$$

$$v_2' = \frac{(m_2 - \varepsilon\, m_1) v_2 + (1 + \varepsilon) m_1 v_1}{m_1 + m_2} \tag{16}$$

There are two cases of collision:

i- A perfect elastic collision. There is no loss of kinetic energy in collision ($\varepsilon = 1$).

ii- An inelastic collision. There is part of the kinetic energy that is changed to some other form of energy ($\varepsilon < 1$). For the most real bodies, the value of ε is between 0 and 1.

3.3.2 Colliding Bodies Optimization

In Colliding Bodies Optimization (CBO), the Colliding Bodies (CBs) (the current population of designs) are divided into two equal groups: stationary and moving objects [10]. The moving objects move toward and collide the stationary objects causing:

1. Stationary objects to move to another position.
2. Moving objects to change their position.

After the collision, the position of both colliding and stationary bodies (population of designs) are updated using new velocities from Eqs. (15) and (16). The CBO can be executed in the following three steps:

Step 1: Initialization. Initializing an array of CBs (initial population of designs) with random positions as follow:

$$x_i^j = x_{j,min} + rn_i^j \times \left(x_{j,max} - x_{j,min}\right); \quad i = 1, 2, \ldots, 2n \text{ and } j = 1, 2, \ldots, nvar \tag{17}$$

where x_i^j is the jth variable of the ith design in the CB matrix, $x_{j,min}$ and $x_{j,max}$ are the lower and the upper bounds of jth design variable, rn_i^j is a random number between 0 and 1, $2n$ is the total number of CBs or the population size, and $nvar$ is number of design variables. To obtain discrete values for design variables, x_i^j is rounded to the nearest permissible discrete value.

Step 2: Search. This step is divided into 4 sub-steps:

1- CBs ranking: use the merit function $F(\boldsymbol{X})$ to compute CBs' masses (Eq. (18)), and sort the CBs' in a descending order based on their calculated masses:

$$m_i = \frac{1/F_i(\boldsymbol{X})}{\sum_{k=1}^{2n} 1/F_k(\boldsymbol{X})}\ ; \quad i = 1, 2, \dots, 2n \tag{18}$$

where m_i is the mass of the ith body (design), $F_i(X)$ and $F_k(X)$ are the merit function values of the ith and kth bodies (designs), respectively. This way the designs are sorted from the best to the worst. Note that a larger mass in Eq. (18) corresponds to a smaller value for the merit function.

2- Groups creation. CBs are equally divided into two groups:

(i) Stationary CBs: these are the upper half of CBs; these are better designs that are assigned zero velocities before collision:

$$\boldsymbol{v}_s = \boldsymbol{0};\ \ s = 1, 2, \dots, n \tag{19}$$

where v_s is the velocity of the sth CB in the stationary group.

(ii) Moving group: these are the lower part of CBs and they move toward the stationary CBs with velocity before the collision as:

$$\boldsymbol{v}_m = \boldsymbol{X}_m - \boldsymbol{X}_s; \quad m = n+1, \dots, 2n \text{ and } s = m - n \tag{20}$$

where v_m and X_m are the velocity and position of the mth CB in the moving group, respectively, and X_s is the sth CB position in the stationary group.

3- Evaluation after the collision. After the collision, velocities of stationary and moving CBs are calculated based on inelastic one dimensional collision of two bodies using Eqs. (15) and (16):

$$\boldsymbol{v}'_s = \frac{(1+\varepsilon)\, m_m\, \boldsymbol{v}_m}{m_m + m_s};\ \ s = 1, 2, \dots, n \text{ and } m = s + n \tag{21}$$

and the velocity of the moving CBs is obtained as:

$$\boldsymbol{v}'_m = \frac{(m_m - \varepsilon\, m_s)\, \boldsymbol{v}_m}{m_m + m_s};\ \ m = n+1, \dots, 2n \text{ and } s = m - n \tag{22}$$

$$\varepsilon = 1 - \frac{Iter_{P2}}{MaxIter_{P2}} \tag{23}$$

where v'_s is the velocity of the sth CB of the stationary group after collision; v_m and v'_m are the velocity of the mth CB of the moving group before and after collision, respectively; m_s is the mass of the sth CB of the stationary group; m_m is the mass of the mth CB of the moving group; ε is the COR parameter; $Iter_{P2}$ is the current iteration of ECBO; and $MaxIter_{P2}$ is the limit on number of iterations for ECBO. Note that ε is a decreasing function of the iteration number.

4- CBs updating. The new position of CBs are calculated as follows:

$$\boldsymbol{X}_s^{new} = \boldsymbol{X}_s + [\boldsymbol{rn}_s]\boldsymbol{v}'_s;\ \ s = 1, 2, \dots, n \tag{24}$$

$$\boldsymbol{X}_m^{new} = \boldsymbol{X}_m + [\boldsymbol{rn}_m]\boldsymbol{v}'_m;\ \ m = n+1, \dots, 2n \tag{25}$$

where $\boldsymbol{X}_s^{new}$ and $\boldsymbol{X}_m^{new}$ are the new positions of the stationary and moving bodies, respectively, $\boldsymbol{X}_s$ and $\boldsymbol{X}_m$ are the old positions of the stationary and moving bodies, respectively, rn_s and rn_m are diagonal matrices with diagonal elements as random numbers between -1 and 1. To obtain discrete values of designs, $\boldsymbol{X}_s^{new}$ and $\boldsymbol{X}_m^{new}$ are rounded to the nearest permissible discrete values.

Step 3: Terminating criterion control

If the limit on the number of iterations ($MaxIter_{P2}$) is reached, the algorithm is terminated. Otherwise, go to Step 2.

3.3.3 Enhanced Colliding Bodies Optimization (ECBO)

This metaheuristic algorithm is an enhancement of the standard CBO. It uses memory to save some good designs and a mechanism to escape from local optima to get better solutions faster. This is done by adding two more sub-steps to step 2 of the standard CBO as follows [15]:

1- *Saving*: this sub-step is added between sub-steps i and ii in step 2 of the standard CBO. In this sub-step, some historically good designs (having smaller merit function values) and their related information are saved in a matrix called Colliding Memory (CM). The good designs saved in CM replace the worst designs in the current population at the beginning of every iteration. After that, the CM is also updated. The number of designs saved is CMS.

2- *Escaping from local optima*: this sub-step is added after the last sub-step of step 2 of the standard CBO. In this sub-step, a parameter called Pro within [0, 1] is introduced. For each colliding body, rnp_i $(i = 1, 2, \dots, 2n)$, which is a random number uniformly distributed within [0, 1], is compared with Pro. If $Pro > rnp_i$, one component of the ith CB is selected randomly and its value is regenerated as:

$$x_i^j = x_{j,min} + rnp \times \left(x_{j,max} - x_{j,min}\right);\ \ i = 1,2,\ldots,2n \tag{26}$$

where x_i^j is the jth variable of the ith design, rnp is a random number between 0 and 1, and $x_{j,min}$ and $x_{j,max}$ are the lower and upper bounds of the jth variable, respectively. The reason to change just one component of ith CB is to protect the structures of CBs. This mechanism was shown to give diversity leading to better designs [15].

4 HHC: Hybrid Improved Harmony Search-Enhanced Colliding Bodies Algorithm

4.1 Motivation for Hybrid Algorithm

Compared to other metaheuristic algorithms, ECBO is simple, requires just one algorithmic parameter, and performs well in terms of the quality of the solution. IHS is easy to implement and it works fine with any kind of problem. However, both have some shortcomings that were observed while solving some problems. IHS needs specification of several algorithmic parameters that can affect the performance of the algorithm. ECBO makes steady progress towards the neighborhood of the final design whereas IHS makes quite rapid progress towards a similar neighborhood. Therefore, IHS requires fewer structural analyses compared to ECBO to reach a neighborhood of the final design. However, after reaching the neighborhood of the final design, the progress of IHS is quite slow to reach the final design whereas ECBO continues to make good progress towards the solution.

The basic idea of the proposed HHC algorithm is to use IHS in Phase 1 to reach the neighborhood of the solution quickly and then switch to the ECBO to reach the final design. This way ECBO starts with some improved designs in Phase 2. This combination could lead to the final solution in fewer structural analyses which will be very useful while solving more complex problems, such as dynamic response optimization problems with discrete variables and non-differentiable functions.

4.2 Phase 1: Improved Harmony Search (IHS)

IHS is used in Phase 1 to obtain a good set of designs quickly for Phase 2. Two additional steps are added to IHS:

i- *Stopping Criteria*: In addition to a maximum number of iteration criterion discussed in step 5 in section 3.1, a new merit function improvement criterion is added. That is, when there is no or slight improvement in the current merit function value after many iterations, this phase is terminated. The pseudo-code of this criterion is as follows:

If$_1$ $Iter_{P1} \geq r_1 \times MaxIter_{P1}$
  If$_2$ $(Merit(Iter_{P1}) - Merit(Iter_{P1} - r_2 \times MaxIter_{P1}))/Merit(Iter_{P1}) \leq \varepsilon_{P1}$
    Terminate Phase 1
  End$_2$
End$_1$

$$MaxIter_{P1} = 10 \times nvar \times \text{number of elements in the discrete set} \tag{27}$$

where $Iter_{P1}$ is the current iteration, $MaxIter_{P1}$ is the limit number of iterations for Phase 1.

Note that the parameters r_1, r_2 and ε_{P1} are selected so that premature termination of the algorithm does not occur. They do not affect performance of the algorithm in any other way. The limit on number of iterations, $MaxIter_{P1}$ in Eq. (27), is dependent on the number of design variables and the number of elements in the discrete set. When the number of design variables and/or the number of elements in the discrete set increase, the search space enlarges. Therefore, metaheuristic algorithms need more iterations. Thus, Eq. (27) is used instead of a fixed number for each problem.

ii- *Domain reduction*: During the first few iterations (compared with total number of iterations), IHS improves initial designs rapidly. Although, at this stage, the best design may be far from the final design, it was observed that some of the design variables in **HM** have the same or about the same values from iteration to iteration. These design variables are most likely at their best values in this phase. That is, the allowable range for these design variables can be reduced based on their mean

value and standard deviation. In other words, the design domain can be reduced based on the current state of **HM**. Section 4.4 provides more details for this step.

4.3 Phase 2: Enhanced Colliding Bodies Optimization (ECBO)

ECBO starts with $2n$ random designs and it keeps colliding them in search for a better solution, as explained earlier. Thus, if the initial population is not reasonably good, the algorithm most likely needs more iterations to find the final design. In each iteration, ECBO needs to evaluate the problem $2n$ times, where $2n$ is the population size.

In HHC, some better designs generated by Phase 1 are passed on to the **CB** matrix. Then ECBO collides those designs to enhance them further. That is, starting with better designs, the total number of iterations for the ECBO algorithm can be reduced to obtain the final design. This is quite beneficial since ECBO needs to evaluate the problem $2n$ times in one iteration. For example, if the population size is 50 in ECBO and the number of iterations to enhance the initial population is 100, then ECBO alone needs 5000 structural analyses to improve the starting population. However, this improvement may be done with fewer structural analyses by replacing the initial population with some better designs of Phase 1 results. Using the same population sizes of 50, 75, and 100 for both phases (passing all Phase 1 designs to Phase 2) did not improve the performance of the algorithm in terms of the quality of the solutions and the number of structural analyses needed to obtain the final designs. Overall, passing just some better designs of Phase 1 to Phase 2 makes the algorithm obtain the final design more often with a smaller number of structural analyses.

In this phase, the stopping criterion is a maximum number of iterations as follows:

$$MaxIter_{P2} = nvar \times \text{number of elements in the discrete set} \tag{28}$$

Similar to Eq. (27), Eq. (28) is based on number of design variables and number of elements in the discrete set.

4.4 Domain Reduction Technique

This additional step is added to Phase 1 to increase the possibility for IHS to find better designs faster to enhance the general performance of HHC in terms of the number of structural analyses required to find the best design reliably. Domain reduction can be done by looking at the standard deviation of each design variable values for some better designs in the **HM** matrix. When a design variable has a small standard deviation, its upper and lower limits in the allowable set of discrete values D_j for the jth design variable is changed as follows:

$$x_{j,min} = x_{j,avg} - x_{j,sd} \tag{29}$$

$$x_{j,max} = x_{j,avg} + x_{j,sd} \tag{30}$$

$$x_{j,avg} = \frac{1}{nd}\sum_{i=1}^{nd}(x_i^j) \tag{31}$$

$$x_{j,sd} = \sqrt{\frac{1}{nd-1}\sum_{i=1}^{nd}(x_i^j - x_{j,avg})^2} \tag{32}$$

where $x_{j,min}$ and $x_{j,max}$ are the lower and upper bounds of the jth design variable, respectively, $x_{j,avg}$ is the average of jth design variable, $x_{j,sd}$ is the standard deviation of jth design variable, and nd is the number of designs that are considered in calculating the average and the standard deviation. Designs that are considered in this step to find the new upper and lower limits are important. Therefore, this step starts only after a certain number of iterations so that IHS has already improved initial designs enough. IHS with domain reduction shows a better convergence behavior over IHS without domain reduction. Domain reduction makes IHS obtain better designs in fewer iterations.

Sometimes, the upper and lower limits of design variables become equal (the standard deviation is zero). That is, to avoid trapping in local optima, at least five elements are kept in the discrete set by modifying the upper and lower limits as follows:

$$x_{j,min} = x_{j,avg} - 2 \tag{33}$$

$$x_{j,max} = x_{j,avg} + 2 \tag{34}$$

where -2 and +2 imply two elements below and two elements above the average value. Also, when the best design in the current **HM** has a design variable is at the modified lower or the upper bound, the

current lower or upper bound is adjusted using Eq. (33) or Eq. (34), respectively. The proposed domain reduction technique is dynamic. That is, the lower and upper bounds are adjusted based on the best design and the nd better designs in **HM** at each iteration.

The domain reduction step starts when there are feasible and nearly feasible designs in the **HM** matrix (designs that have constraint violation of 5% or less). The minimum number of feasible or nearly feasible designs should not be one so that the upper and lower bounds become the same (in this study, 5% of the population is used as the minimum number of feasible or nearly feasible designs). After sorting of designs in **HM** from the best to the worst, the domain reduction procedure for each design variable is implemented using the following pseudo-code:

If$_1$ $Iter_{P1} \geq r_3 \times MaxIter_{P1}$
  If$_2$ the number of feasible or nearly feasible designs $\geq 5\%$ of HMS
    change the lower and upper limits using Eqs. (29) and (30), respectively.
    If$_3$ the number of elements in the discrete set of the jth design variable < 5
      change the lower and upper limits using Eqs. (33) and (34), respectively.
    End$_3$
    If$_4$ $x_{j,best} \leq x_{j,min}$ or $x_{j,best} \geq x_{j,max}$
      change the lower or upper limits using Eq. (33) or Eq. (34), respectively.
    End$_4$
  End$_2$
End$_1$

here r_3 is the percentage of $MaxIter_{P1}$ to start this criterion. r_3 should be selected so that IHS has already improved designs. Based on observing IHS convergence behavior, it is recommended to use $r_3 \geq$10% of $MaxIter_{P1}$. This way, the standard deviation can give more accurate results about the design variable state. $x_{j,best}$ is the value of the jth design variable of the best design in **HM**.

Reduction of the design variable bounds shrinks the feasible set for the problem. This increases the possibility of obtaining better designs at the end of Phase 1 with a reduced number of structural analyses. Including this technique in Phase 2 showed no improvement in the algorithm's performance since ECBO can efficiently treat larger domains for design variables. Therefore domain reduction scheme is not suggested for the ECBO. The first numerical example in Section 5.1 is used to show how this step reduces the design domain and enhances the performance of the HHC significantly. The pseudocode of HHCD is shown in Algorithm 1.

4.5 Evaluation of the Algorithms

Multiple runs for the same problem will be executed to study the performance of the algorithms. Several metrics will be used in evaluations:

1- *Average of the final merit function values obtained with different runs*. An average value that is closer to the best solution will indicate the ability of the algorithm to obtain the best design more often.
2- *Standard deviation of the final values of the merit functions obtained with different runs*. A smaller value of the standard deviation will imply robustness of the algorithm to obtain the best design with different runs for the problem.
3- *Average of the number of structural analyses needed to reach the final solution*. A smaller value will indicate more efficient algorithm.
4- *Standard deviation of the number of structural analyses*. A smaller value will indicate the robustness of the algorithm to obtain the final design in approximately same number of structural analyses with different runs for the same problem.

5. Numerical Examples

Before the proposed HHC algorithm can be used to solve more complex and larger problems, it needs to be tested to solve some standard test problems and study its performance. In the following sections, some of the popular discrete truss optimization examples are solved for minimum structural weight to compare the performance of HHC with other metaheuristic optimization algorithms. Structures are

analyzed using the finite element (direct stiffness) method and algorithms are coded using MATLAB©. For all problems, Phase 1 parameters are set as follows: HMS is 75, $HMCR_{max}$ and PAR_{max} are 0.85 and $HMCR_{min}$ and PAR_{min} are 0.35. Phase 2 parameters are set as follows: population size ($2n$) is 40, Pro is 0.5, and the number of designs to be saved in **CM** (*CMS*) is 4 ($2n/10$). Phase 1 improvement criterion ratios, r_1 and r_2, are 0.25 and 0.10, respectively. Domain reduction ratios, r_3 is 0.10 and ε_{P1} is 10^{-3}. These parameters are selected based on studying the convergence behavior of IHS. IHS obtains good and diverse designs (in comparison with initial random designs) after about 25% of the maximum number of iterations; then it converges very slowly. HHC's performance was evaluated using population sizes of 50, 75 and 100 for Phase 1 and population sizes of 20, 30, 40, and 50 for Phase 2 with Pro of 0.25, 0.4 and 0.5. The results showed that the combinations of these parameters worked. It is concluded that these parameters are not problem dependent and are kept fixed for all design examples.

Since the optimization algorithms are stochastic in nature, 50 independent optimization runs were performed for each example to test the performance of HHC. Also, five example problems were solved but the detailed results for three examples are presented here. The other two examples had similar results and trends. They are: planar 15-bar truss and planar 52-bar truss [15]. They are omitted to keep a reasonable length of the paper.

The number of maximum iterations varies based on the number of design variables and the number of elements in the discrete set (Eqs. 27 and 28). For IHS and ECBO, the maximum numbers of iterations were set to 50000 and 1000, respectively, to allow these algorithms to fully search for the best design. NSA (Number of Structural Analyses) is calculated as follows:

$$\mathrm{NSA_{HHC}} = HMS + NIter_{P1} + (NIter_{P2} - 1) \times \text{poplution size} \quad (35)$$

$$\mathrm{NSA_{ECBO}} = NIter \times \text{poplution size} \quad (36)$$

$$\mathrm{NSA_{IHS}} = HMS + NIter \quad (37)$$

where HMS is harmony memory size, $NIter$ is the number of iterations and the subscript refers to the phase. In Eq. (35), $(NIter_{P2} - 1)$ implies that the designs passed to Phase 2 do not need to be evaluated again.

5- To study the domain reduction effects on the behavior of HHC, all numerical examples were tested without the domain reduction step as well. In the next sections, HHC refers to the algorithm without the domain reduction step, while HHCD refers to the algorithm with the domain reduction step.

Algorithm 1. Pseudocode of HHCD.

Inputs: : Phase1: HMS = 75, $HMCR_{max}$ = 0.85, PAR_{max} = 0.85, $HMCR_{min}$ = 0.35, PAR_{min} = 0.35, r_1 =0.25, r_2 = 0.1, r_3 = 0.10 and ε_{P1} = 10^{-3}.
Phase2: n = 20, Pro =0 0.5, and CMS = 4.

Outputs: The location of the best design, its fitness value, and the number of analyses.

Phase 1: IHS

while$_1$ $Iter_{P1} \geq MaxIter_{P1}$
Generate new designs using Eqs. 7 to 12.
Stopping criteria
If$_1$ $Iter_{P1} \geq r_1 \times MaxIter_{P1}$
If$_2$ $(Merit(Iter_{P1}) - Merit(Iter_{P1} - r_2 \times MaxIter_{P1}))/Merit(Iter_{P1}) \leq \varepsilon_{P1}$
Terminate Phase 1
End$_2$
End$_1$
Domain reduction
If$_3$ $Iter_{P1} \geq r_3 \times MaxIter_{P1}$
If$_4$ the number of feasible or nearly feasible designs $\geq$ 5% of HMS
change the lower and upper limits using Eqs. (29) and (30), respectively.
If$_5$ the number of elements in the discrete set of the jth design variable < 5
change the lower and upper limits using Eqs. (33) and (34), respectively.
End$_5$
If$_6$ $x_{j,best} \leq x_{j,min}$ or $x_{j,best} \geq x_{j,max}$
change the lower or upper limits using Eq. (33) or Eq. (34),

respectively.

End$_6$

End$_4$

End$_3$

end while$_1$

Pass the best 40 designs to Phase 2

Phase 2: ECBO

while$_2$ $Iter_{P2} \geq MaxIter_{P2}$

Generate new designs using Eqs. 18 to 26.

end while$_2$

5.1 Planar 10-bar truss

Figure 1 shows the configuration of the 10-bar truss. This popular benchmark example has been solved by many researchers, e.g., Rajeev and Krishnamoorthy [17], Li et al. [16], Xiang et al. [18], Camp [19], and others. For all members, the modulus of elasticity is 10,000 ksi and material density is 0.1 lb/in3. The allowable displacement for all nodes in both vertical and horizontal directions equals ±2.0 in. All members are subjected to stress limitations of 25 ksi for both tension and compression. The structure is subjected to two vertical downward loads, P=100 kips, at joint 2 and 4. Cross-sectional areas of all members are the design variables that are selected form the discrete set of 42 elements as follows:

D=[1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97, 11.50, 13.50, 13.90, 14.20, 15.50, 16.00, 16.90, 18.80, 19.90, 22.00, 22.90, 26.50, 30.00, 33.50] (in^2). (38)

As noted earlier, this study case is used to also show how HHCD works. In this illustrative example, $MaxIter_{P1}$=10×10×42=4200 and $MaxIter_{P2}$=10×42=420. The rest of the internal parameters are set as mentioned earlier. Phase 1 (IHS) starts with 75 random designs that are evaluated using Eq. (4). Table 1 gives the best design among these 75 initial designs. The total structural weights (and merit function values) for HHC and HHCD are 4888.346 lb (38724.564) and 4929.869 lb (26734.436) with values of the violation parameter G as 1.815 and 1.329, respectively. After iteration 420 (0.1× $MaxIter_{P1}$), the domain reduction process starts. Table 1 shows that at the end of Phase 1 some design variables' bounds were reduced based on the method discussed in Section 4.3. Due to this step, the possible design combinations are reduced from 1.708e16 (4210) to 5.976e9 in Phase 1. For example, the bounds of the first design variable were changed many times after iteration 420 until its upper and lower limits became sections 28 and 42 in the set D, respectively.

At the end of the Phase 1, the total structural weights for HHC and HHCD are 5959.483 lb and 5788.563 lb, respectively. Since there is no violation of constraints, the merit function value is same as the structural weight. Phase 1 terminates at iteration 1051 for HHC and HHCD while the maximum number of iterations allowed for this phase is 4200. This implies that the proposed new stopping criterion terminates this phase due to no improvement in the current best design.

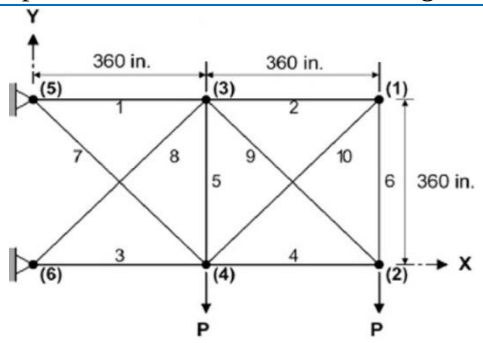

Fig. 1 Schematic of 10-bar planar truss.

The goal of the Phase 1 is to provide Phase 2 with better designs that may be closer to the best solution so that ECBO requires fewer iterations. Study of designs in HM shows that HHCD was able to improve not only the best design but also all the designs in HM. This explains the reason that HHCD needs a fewer number of structural analyses. At the end of Phase 1, the best 40 designs in HM are passed to the CB matrix. Therefore, Phase 2 starts with improved designs instead of random designs. In this example, Phase 1 iterations of 1051 are equivalent to just 26 iterations of ECBO with a population size of 40. At iteration 1126 (1051 for Phase 1 and 75 for Phase 2), the algorithm obtains the best structural weight of 5490.738 lb with no constraint violation. Phase 2 needs 76 iterations to find the best design that, generally, is less than what ECBO would need (see Table 2). HHCD shows similar behavior in all other numerical examples.

Table 1 Domain reduction technique illustration for planar 10-bar truss structure.

Design variables (in^2)		Best initial design		Best design at end of Phase 1		Best design at end of Phase 2		Design variables bounds at end of Phase 1[b] HHCD	
		HHC	HHCD	HHC	HHCD	HHC	HHCD	Lower bound	Upper bound
1	A1	15.50 (33[a])	16.90 (35)	33.50 (42)	30.00 (41)	33.50 (42)	33.50 (42)	28	42
2	A2	1.62 (1)	13.9 (31)	2.38 (5)	2.38 (5)	1.62 (1)	1.62 (1)	1	7
3	A3	33.50 (42)	13.5 (30)	26.50 (40)	30.00 (41)	22.90 (39)	22.90 (39)	27	42
4	A4	3.87 (17)	14.20 (32)	13.50 (30)	14.20 (32)	14.20 (32)	14.20 (32)	24	34
5	A5	3.55 (14)	7.97 (28)	3.13 (11)	3.09 (10)	1.62 (1)	1.62 (1)	2	12
6	A6	4.18 (19)	5.12 (25)	3.63 (15)	2.38 (5)	1.62 (1)	1.62 (1)	1	7
7	A7	3.84 (16)	3.84 (16)	14.20 (32)	13.50 (30)	7.97 (28)	7.97 (28)	28	32
8	A8	22.00 (38)	22.90 (39)	22..90 (39)	22.90 (39)	22.90 (39)	22.90 (39)	21	41
9	A9	4.18 (19)	3.47 (13)	19.90 (37)	16.90 (35)	22.00 (38)	22.00 (38)	30	37
10	A10	22.00 (38)	16.00 (34)	1.62 (1)	2.38 (5)	1.62 (1)	1.62 (1)	2	6
Weight (lb)		4888.346	4929.869	5959.483	5788.563	5490.738	5490.738	-	-
G (Eq.5)		1.815	1.329	0.0	0.0	0.0	0.0	-	-
F (Eq.4)		38724.564	26734.436	5959.483	5788.563	5490.738	5490.738	-	-

[a]Section number in the set D. [b]Lower and upper bounds for HHC remain fixed at 1 and 42 for all members.

Figure 2 demonstrates the convergence history of the best run of IHS, ECBO, HHC, and HHCD. It shows that IHS and Phase 1 of HHC and HHCD reach better designs faster than ECBO. HHCD convergences to the best design faster than ECBO and HHC because Phase 2 starts with better designs.

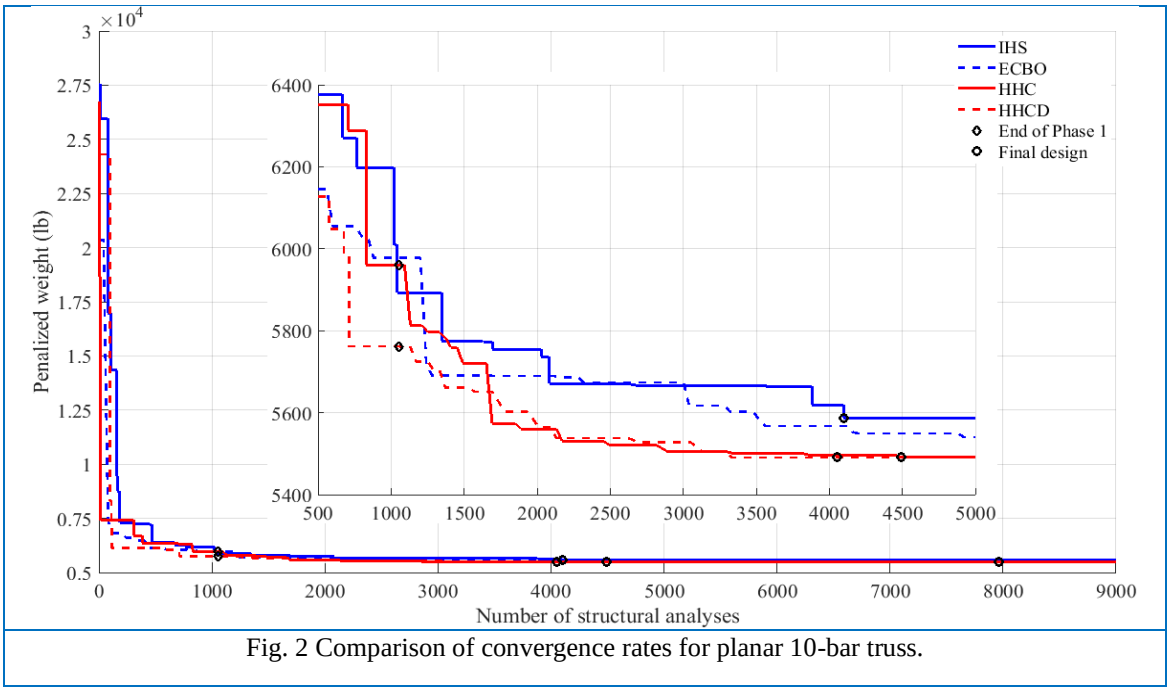

Fig. 2 Comparison of convergence rates for planar 10-bar truss.

When IHS stops improving design at iteration 4101. Note that every iteration in ECBO and Phase 2 of

Table 2 Comparison of optimal designs for 10-bar truss problem.

Design variable (in^2)		GA [17]	HPSO [16]	SA [18]	BB-BC [19]	This work			
						IHS[d]	ECBO[e]	HHC	HHCD
1	A1	33.50	30.00	33.50	33.50	30.00	33.50	33.50	33.50
2	A2	1.62	1.62	1.62	1.62	2.62	1.62	1.62	1.62
3	A3	22.00	22.90	22.90	22.90	22.90	22.90	22.90	22.90
4	A4	15.50	13.50	14.20	14.20	14.20	14.20	14.20	14.20
5	A5	1.62	1.62	1.62	1.62	1.80	1.62	1.62	1.62
6	A6	1.62	1.62	1.62	1.62	1.80	1.62	1.62	1.62
7	A7	14.20	7.97	7.97	7.97	11.50	7.97	7.97	7.97
8	A8	19.90	26.50	22.90	22.90	22.00	22.90	22.90	22.90
9	A9	19.90	22.00	22.00	22.00	22.00	22.00	22.00	22.00
10	A10	2.62	1.80	1.62	1.62	2.38	1.62	1.62	1.62
Best weight (lb)		5613.580	5531.984	5490.738	5490.738	5586.289	5490.738	5490.738	5490.738
NSA[a]		800	50000	10500	8694	4176	7960	4566	4126
G (Eq. 5)		3.77×10^{-4}	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Weight (lb)	Average	N/A	N/A	N/A	5494.17[c]	5680.406	5519.357	5499.116	5490.873
	SD[b]	N/A	N/A	N/A	12.42	40.582	53.183	30.732	0.943
NSA	Average	N/A	N/A	N/A	N/A	6999	19378	9821	8979
	SD	N/A	N/A	N/A	N/A	2728	6215	5038	3890

[a]NSA is number of structural analyses. [b]SD is the standard deviation of 50 independent runs. [c]The average of 100 runs.
[d]The maximum number of iterations is 50000. [e]The maximum number of iterations is 1000.

HHC and HHCD requires 40 structural analyses.

Table 2 summarizes results available in the literature with four different algorithms, and results from the present study. It also shows the mean values and standard deviations of the best structural weight from 50 independent runs for IHS, ECBO, HHC, and HHCD. The results show that GA, HPSO, and IHS did not obtain the best design. HHCD was able to find the best design after 4126 structural analyses. This is the same weight as obtained by SA, BB-BC, ECBO, and HHC; however, HHCD needs fewer structural analyses to obtain the best solution.

Figure 3 shows the best merit function value for each of the 50 runs for IHS, ECBO, HHC, and HHCD. It is seen that IHS was not able to obtain the final design in any run; HHC was able to reach the final design 42 times; and ECBO reached the final design 26 times. It is seen that HHC performs better than ECBO as well as IHS.

Figure 3 shows that HHCD was able to find the best solution 49 times. The average and the standard deviation of 50 runs (Table 2) and Fig. 3 demonstrate that HHCD is very effective and robust algorithm. Its average for the structural weight is closest to the best solution and its standard deviation is the smallest. This is important because it shows that HHCD does not require multiple runs to find the best solution. The average and standard deviation of number of structural analyses shows that HHCD is efficient (Table 2). IHS has the lowest NSA average but the quality of the solution is not good.

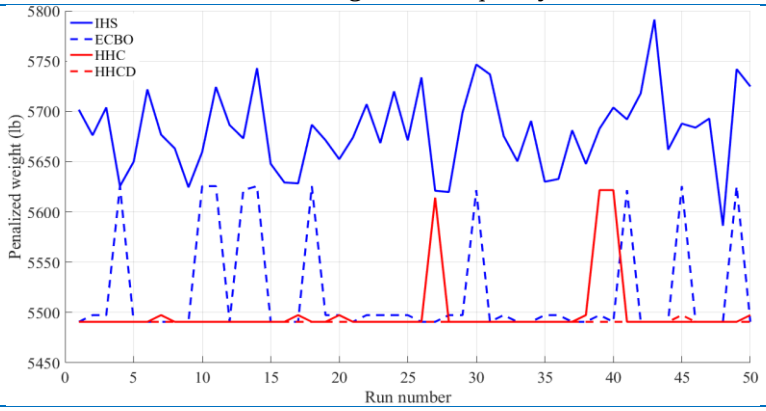

Fig. 3 Comparison of best designs from 50 runs for the 10-bar truss structure.

5.2 Spatial 25-bar truss

Figure 4 shows the configuration of the spatial 25-bar truss. This example was solved in Rajeev and Krishnamoorthy [17], Lee et al. [20], Li et al. [16], Xiang et al. [18], Camp [19], and Kaveh and Mahdavi [15]. For all members, the modulus of elasticity is 10,000 ksi and material density is 0.1 lb/in3. The allowable displacement for all nodes in both vertical and horizontal directions is ±0.35 in. All members are subjected to stress limitations of 40 ksi for both tension and compression. This spatial truss was subjected to the two loading conditions shown in Table 3. The structure includes 25 members organized into 8 groups as given in Table 4. Design variables are selected form the discrete set of 30 elements as follows:

$$D=[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2, 3.4] \ (in^2). \quad (39)$$

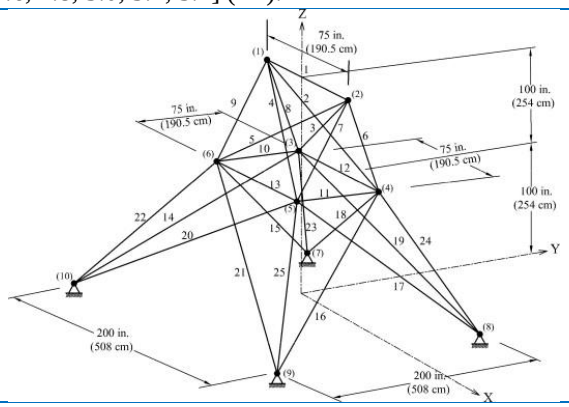

Fig. 4 Schematic of 25-bar space truss.

Table 3 Load conditions of the spatial 25-bat truss.

Case	Load condition	Nodes	Loads (kips)		
			P_x	P_y	P_z
1	1	1	1.0	-10.0	-10.0
		2	0.0	-10.0	-10.0
		3	0.5	0.0	0.0
		6	0.6	0.0	0.0
2	1	1	0.0	20.0	-5.0
		2	0.0	-20.0	-5.0
	2	1	1.0	10.0	-5.0
		2	0.0	10.0	-5.0
		3	0.5	0.0	0.0
		6	0.5	0.0	0.0

In this study case, ψ was 10 (in Eq. 4) because algorithms found some infeasible solutions with very small violation when ψ was 1. HHCD was able to obtain the best design after 2043 structural analyses (759 iterations) as shown in Figure 5 and Table 4. Figure 5 shows the convergence history of the best run of IHS, ECBO, HHC, and HHCD. It shows that HHCD and HHC convergence faster than ECBO to the best design where IHS did not obtain the best design. Table 4 shows that although HS, HPSO, SA, BB-BC and ECBO obtained the best design, both HHC and HHCD need fewer structural analyses. Also, Table 4 explains that HHCD has the lowest average and standard division of NSA and the best average and standard division of final designs from 50 runs while HHC is close second. Although, BB-BC shows slightly better stability than HHCD, it needs more structural analyses compared to HHCD to obtain the best design. GA did not reach the final design.

Figure 6 shows the penalized weight for each of the 50 runs for IHS, ECBO, HHC, and HHCD. It shows that IHS did not reach the final design in any run. HHCD reached the final design more than HHC and ECBO.

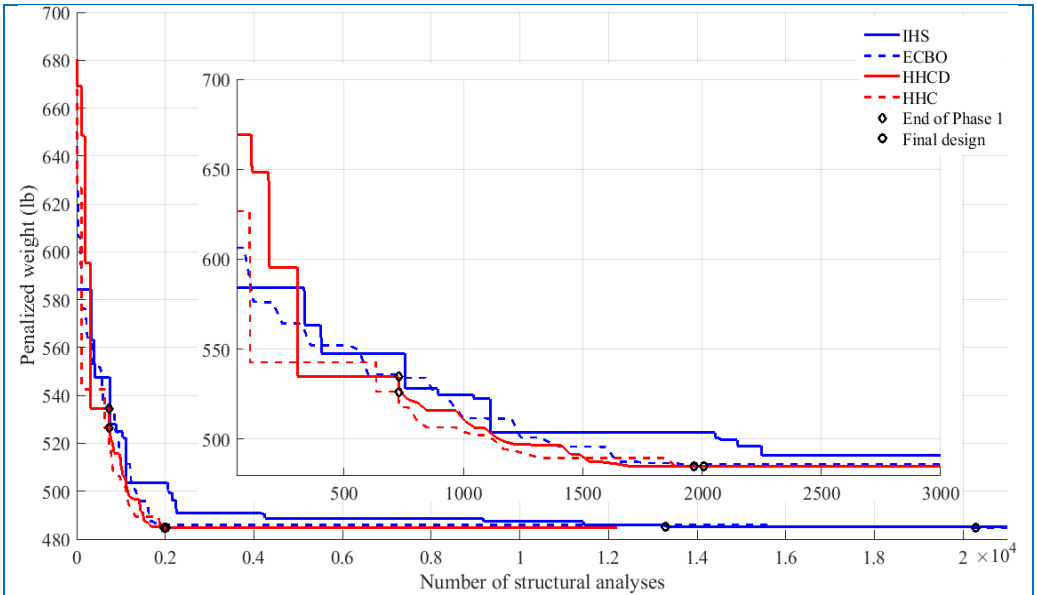

Fig. 5 Comparison of convergence rates for 25-bar spatial truss.

Table 4 Performance comparison for the 25-bar spatial truss.

Design variable (in2)		GA [17]	HS [18]	HPSO [16]	SA [18]	BB-BC [19]	ECBO [15]	This work			
								IHS[4]	ECBO[f]	HHC	HHCD
1	A1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	A2-A5	1.8	0.3	0.3	0.3	0.3	0.3	0.5	0.3	0.3	0.3
3	A6-A9	2.3	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4
4	A10-A11	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
5	A12-A13	0.1	2.1	2.1	2.1	2.1	2.1	1.9	2.1	2.1	2.1
6	A14-A17	0.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
7	A18-A21	1.8	0.5	0.5	0.5	0.5	0.5	0.4	0.5	0.5	0.5
8	A22-A25	3.0	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4
Best weight (lb)		546.013	484.854	484.854	484.854	484.854	484.854	485.054	484.854	484.854	484.854
NSA		800	13523	25000	7900	9090	61200[a]	13368	20280	2083	2043
G (Eq. 5)		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Weight (lb)	Average	N/A	N/A	N/A	486.354[b]	485.10[c]	485.89[d]	490.547	485.575	485.480	485.252
	SD	N/A	N/A	N/A	N/A	0.44	N/A	4.986	1.244	0.850	0.505
NSA's	Average	N/A	N/A	N/A	N/A	N/A	N/A	23791	17694	8288	7045
	SD	N/A	N/A	N/A	N/A	N/A	N/A	12039	10876	4194	3233

[a]Population size is 30. [b]The average of 12 runs. [c]The average of 100 runs. [d]The average of 20 runs.
[e]The maximum number of iterations is 50000. [f]The maximum number of iterations is 1000.

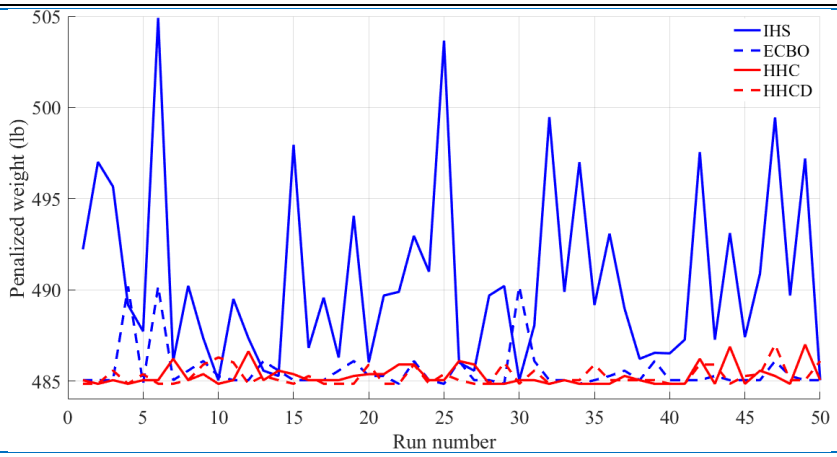

Fig. 6 Comparison of convergence rates for 25-bar spatial truss.

5.3 Spatial 72-bar truss structure

Figure 7 shows the configuration of the spatial 72-bar truss. This example was solved in Li et al. [16], Kaveh and Khayat [11], and Kaveh and Mahdavi [15]. For all members, the modulus of elasticity is 10000 ksi and material density is 0.1 lb/in3. The allowable displacement for all nodes in both vertical and horizontal directions equals ±0.25 in. All members are subjected to stress limitations of 25 ksi for both tension and compression. This spatial truss was subjected to the two loading conditions, as shown in Table 5. The structure includes 72 members organized into 16 groups (Table 6). Design variables are selected from the discrete set of 64 elements as follows:

D=[0.111, 0.141, 0.196, 0.250, 0.307, 0.391, 0.442, 0.563, 0.602, 0.766, 0.785, 0.994, 1.000, 1.228, 1.266, 1.457, 1.563, 1.620, 1.800, 1.990, 2.130, 2.380, 2.620, 2.630, 2.880, 2.930, 3.090, 3.13, 3.380, 3.470, 3.550, 3.630, 3.840, 3.870, 3.880, 4.180, 4.220, 4.490, 4.590, 4.800, 4.970, 5.120, 5.740, 7.220, 7.970, 8.530, 9.300, 10.850, 11.500, 13.500, 13.900, 14.200, 15.500, 16.000, 16.900, 18.800, 19.900, 22.000, 22.900, 24.500, 26.500, 28.000, 30.000, 33.500] (in^2). (40)

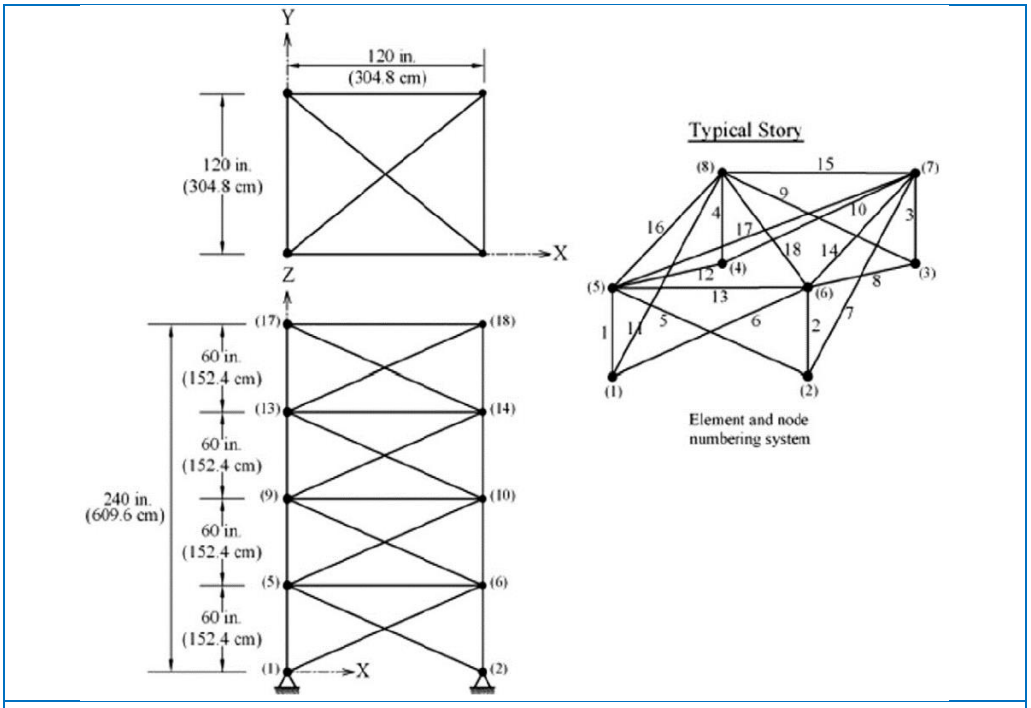

Fig. 7 Schematic of 72-bar space truss.

Table 5 Load conditions of the spatial 72-bat truss.

Case	Nodes	Loads (kips)		
		P_x	P_y	P_z
1	17	5.0	5.0	-5.0
2	17	0.0	0.0	-5.0
	18	0.0	0.0	-5.0
	19	0.0	0.0	-5.0
	20	0.0	0.0	-5.0

Table 6 provides a comparison between some best designs reported in the literature along with those obtained in this study. HHCD obtained best design after 20836 structural analyses (3017 iteration). The best structural weight of 389.334 lb was obtained by IRO, ECBO, and HHC after 17925, 95400 (35100 in this study), 26476 structural analyses, respectively. It is seen that both HHC and HHCD performed better than IHS and ECBO. Note that in ECBO (this work), HHC and HHCD, design variables 2 and 6 values are 0.442 and 0.563, respectively, whereas in IRO and ECBO (Kaveh and Mahdavi, 2015), they are 0.563 and 0.442. In this study case, HHCD needs more analyses than IRO; however, HHCD has smaller average value of 50 independent runs (see Table 6).

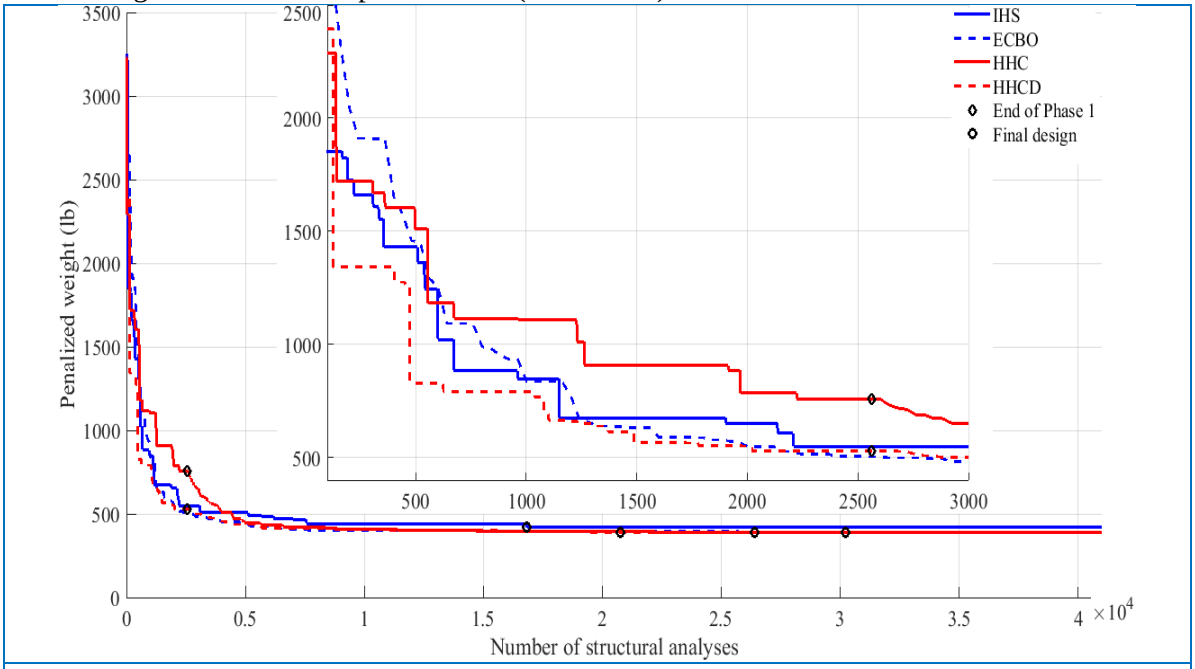

Fig. 8 Comparison of convergence rates for 72-bar spatial truss.

Figure 8 shows the convergence history of IHS, ECBO, HHC, and HHCD. It shows at HHCD convergences faster than IHS, ECBO, and HHC. Figure 9 and Table 6 demonstrate that HHCD is more stable than IRO, CBO, ECBO, IHS, and HHC in terms of the quality of final designs. For a similar quality of design other algorithms needed more simulations (NSA averages in Table 6).

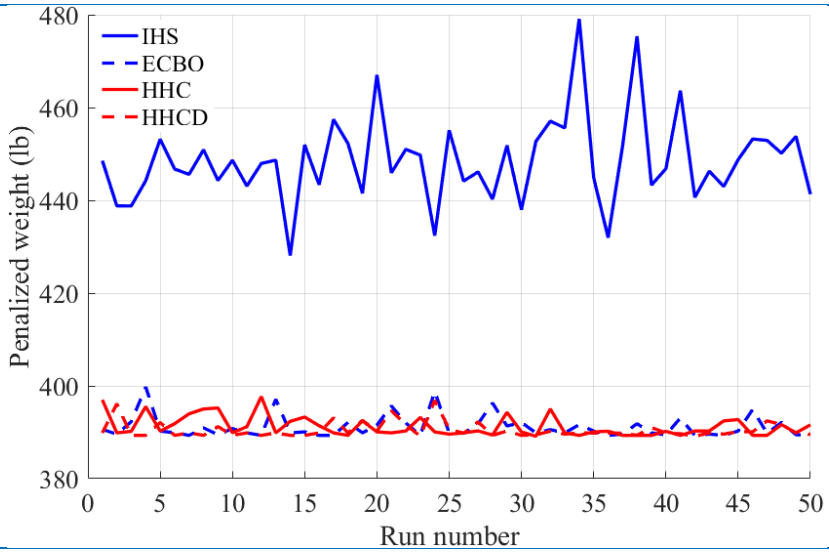

Fig. 9 Comparison of convergence rates for 72-bar spatial truss.

Table 6 Performance comparison for the 72-bar spatial truss.

Design variable (in2)		HPSO [16]	IRO [11]	CBO [15]	ECBO [15]	This work			
						IHS[c]	ECBO[d]	HHC	HHCD
1	A1-A4	4.970	1.990	2.130	1.990	2.62	1.990	1.990	1.990
2	A5-A12	1.228	0.563	0.563	0.563	0.442	0.442	0.442	0.442
3	A13-A16	0.111	0.111	0.111	0.111	0.141	0.111	0.111	0.111
4	A17-A18	0.111	0.111	0.111	0.111	0.141	0.111	0.111	0.111
5	A19-A22	2.880	1.228	1.228	1.228	1.457	1.228	1.228	1.228
6	A23-A30	1.457	0.442	0.442	0.442	0.563	0.563	0.563	0.563
7	A31-A34	0.141	0.111	0.141	0.111	0.141	0.111	0.111	0.111
8	A35-A36	0.111	0.111	0.111	0.111	0.196	0.111	0.111	0.111
9	A37-A40	1.563	0.563	0.442	0.563	0.442	0.563	0.563	0.563
10	A41-A48	1.228	0.563	0.563	0.563	0.602	0.563	0.563	0.563
11	A49-A52	0.111	0.111	0.111	0.111	0.141	0.111	0.111	0.111
12	A53-A54	0.196	0.111	0.111	0.111	0.141	0.111	0.111	0.111
13	A55-A58	0.391	0.196	0.196	0.196	0.25	0.196	0.196	0.196
14	A59-A66	1.457	0.563	0.563	0.563	0.563	0.563	0.563	0.563
15	A67-A70	0.766	0.391	0.391	0.391	0.442	0.391	0.391	0.391
16	A71-A72	1.563	0.563	0.563	0.563	0.391	0.563	0.563	0.563
Best weight (lb)		933.094	389.334	391.230	389.334	418.380	389.334	389.334	389.334
NSA		50000	17925	160200	95400[a]	16918	30240	26476	20836
G (Eq. 5)		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Weight (lb)	Average	N/A	408.17	456.69	391.59[b]	448.554	391.173	391.242	390.632
	SD	N/A	N/A	N/A	N/A	9.412	2.073	2.105	1.679
NSA	Average	N/A	N/A	N/A	N/A	14135	37531	32208	27442
	SD	N/A	N/A	N/A	N/A	6037	8858	7994	6884

[a]Population size is 30. [b]The average of 20 runs. [c]The maximum number of iterations is 50000.
[d]The maximum number of iterations is 1000.

6 Discussion and Conclusions

A new two-phase metaheuristic optimization algorithm was presented in this study. Phase 1 used Improved Harmony Search (IHS) with a new domain reduction technique that used statistical analysis of some of the better designs in the current population. Phase 2 used the Enhanced Colliding Bodies Optimization (ECBO) where the initial population consisted of some of the better designs from Phase 1. With this better initial population, ECBO obtained the best design more efficiently. Also, in Phase 1, an improved stopping criterion was proposed that terminated the phase when there was no or small improvement in the best design after many iterations.

Detailed results for three standard truss test structures were presented and discussed. Table 7 summarize comparative data obtained with IHS, ECBO, HHC and HHCD for the three design examples. It shows, in term of the quality of the solution, HHCD obtained the best designs with the lowest averages and standard deviations from 50 independent runs. HHC is close second behind HHCD. Figure 10 is a bar chart representation of the number of structural analyses needed to reach final designs of the three numerical examples with the four methods. The best structural weight values are also shown with bar charts. It shows that IHS does not reach the best design for any of the examples. ECBO needs the largest number of analyses to obtain the final designs. However, HHC and HHCD need a smaller number of simulations to reach the final designs. Tables 7 and Fig. 10 show that HHCD is quite stable and more efficient among all metaheuristic algorithms that are discussed in this study. For the 3-D truss problems (the last two examples), HHCD shows an outstanding performance in terms of the number of structural analyses needed to obtain the best design. This is an attractive feature of the proposed metaheuristic algorithm with domain adjustment.

Based on the comparison with other metaheuristic optimization algorithms for the numerical examples, the following conclusions are drawn:

1- The 50 independent runs for each example showed that the proposed HHC algorithm was quite reliable in obtaining the best designs for each run. Also, HHCD had the lowest averages and standard deviations for the final cost function values. This implies that fewer runs are needed to obtain the best design compared to many other stochastic algorithms.
2- The proposed domain adjustment approach worked very well with IHS.
3- The proposed hybrid algorithm with domain adjustment was able to find the best design with fewer structural analyses by a substantial amount in some cases. This efficiency is critically important for solving more complex applications, such as nonlinear structural analysis problems, dynamic response optimization problems, and multidisciplinary optimization problems.

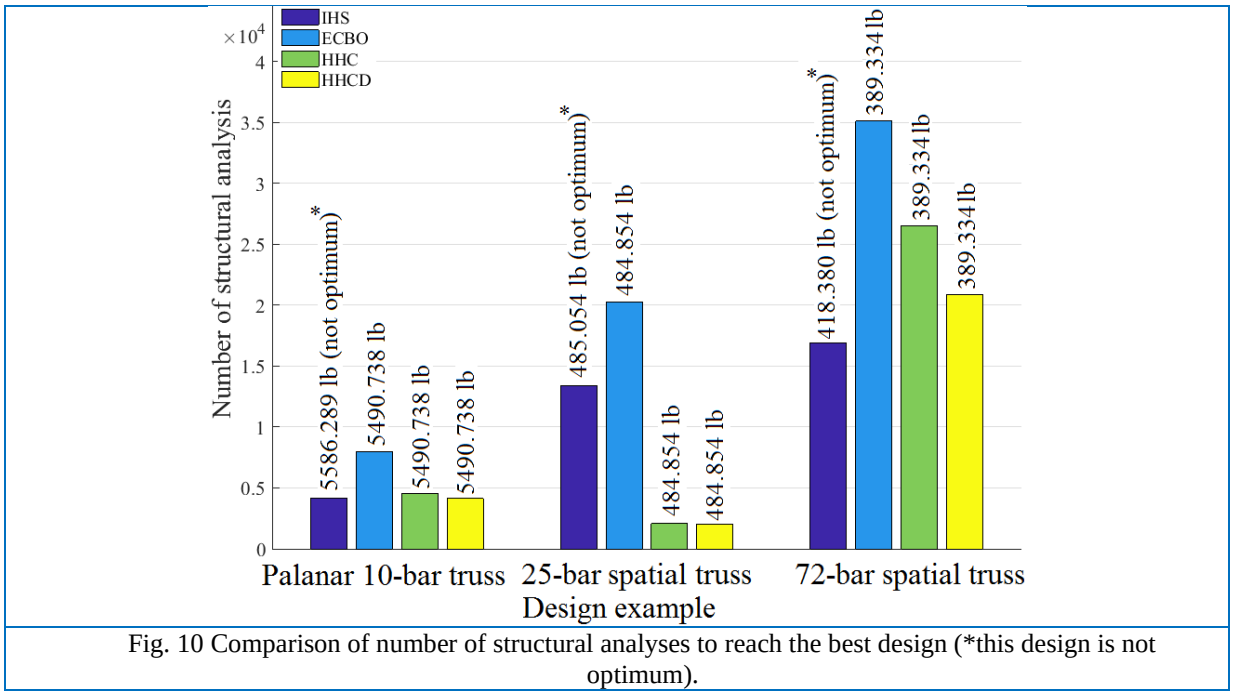

Fig. 10 Comparison of number of structural analyses to reach the best design (*this design is not optimum).

Table 7 Comparative data for design examples.

Design Example			Optimization algorithm			
			IHS	ECBO	HHC	HHCD
Planar 10-bar truss	Best weight (lb)		5586.289	5490.738	5490.738	5490.738
	Weight (lb)	Average	5680.406	5519.357	5499.116	5490.873
		SD	40.582	53.183	30.732	0.943
	NSA	Average	6999	19378	9821	8979
		SD	2728	6215	5038	3890
25-bar spatial truss	Best weight (lb)		485.054	484.854	484.854	484.854
	Weight (lb)	Average	490.547	485.575	485.480	485.252
		SD	4.986	1.244	0.850	0.505
	NSA	Average	23791	17694	8288	7045
		SD	12039	10876	4194	3233
72-bar spatial truss	Best weight (lb)		418.380	389.334	389.334	389.334
	Weight (lb)	Average	448.554	391.173	391.242	390.632
		SD	9.412	2.073	2.105	1.679
	NSA	Average	14135	37531	32208	27442
		SD	6037	8858	7994	6884

Acknowledgments: The researcher would like to thank everyone who has helped us and convey their sincere gratitude.

Author Contributions: The authors contributed to all parts of the current study.

Funding: This study received no external funding .

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] Arora, J. S. (2017). Introduction to Optimum Design (4th Ed.). Elsevier Inc.
[2] Kaveh, A. (2017). Advances in Metaheuristic Algorithms for Optimal Design of Structures (2nd Ed.). Springer.
[3] Mahdavi, M., Fesanghary, M., Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. Applied Mathematics and Computation. 188(2), 1567–1579.
[4] Goldberg, D. E., Holland, J. H. (1988). Genetic algorithms and machine learning. Machine learning, 3(2), 95–99.
[5] Kirkpatrick, S., Gelatt, C., Vecchi, M. (1983). Optimization by simulated annealing. Science, 220 (4598): 671–80.
[6] Kennedy, J., Eberhart, R. (2001). Swarm intelligence. San Francisco: Morgan Kaufmann Publishers.
[7] Dorigo, M., Maniezzo, V., Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26(1):29–41.
[8] Geem, Z. W., Kim, J. H., Loganathan, G. V. (2001). A New Heuristic Optimization Algorithm: Harmony Search. SAGE Journals. 76(2): 60-68.
[9] Erol, O., Eksin, I. (2006). A new optimization method: Big Bang-Big Crunch. Advances in Engineering Software. 37(2), 106–111.
[10] Kaveh, A., Mahdavi, V. R. (2014). Colliding bodies optimization: a novel meta-heuristic method. Computers and Structures. 139: 18-27.
[11] Kaveh, A., Khayat, A. M. (2012). A novel meta-heuristic method: ray optimization. Computer and Structures. 112–113:283–294.
[12] Degertekin, S. O. (2008). Optimum design of steel frames using harmony search algorithm. Structural and Multidisciplinary Optimization, 36: 393–401.
[13] Geem, Z. W. (2009). Harmony Search Algorithms for Structural Design Optimization. Springer.
[14] Sun, W., Chang, X. (2015). An improved harmony search algorithm for power distribution network planning. Journal of Electrical and Computer Engineering. Vol. 2015, Article ID 753712.
[15] Kaveh A, Mahdavi R (2015). Colliding Bodies Optimization: Extension and Application. Springer.
[16] Li, L. J., Huang, Z. B., Liu, F. (2009). A heuristic particle swarm optimization method for truss structures with discrete variables. Computer and Structures. 87(7–8): 435–443.

[17] Rajeev, S., Krishnamoorthy, C. (1992). Discrete optimization of structures using genetic algorithms. Journal of Structural Engineering. 118(5): 1233–1250.
[18] Xiang, B. W., Chen, R. Q., Zhang, T. (2009). Optimization of trusses using simulated annealing for discrete variables, image analysis and signal processing. International Conference on Image analysis and signal processing. 2009, pp. 410.
[19] Camp, C. V. (2007). Design of space trusses using big bang–big crunch optimization. Journal of Structural Engineering. 133(7): 999–1008.
[20] Lee, K. S., Geem, Z. W., Lee, S. H., Bae, K. W. (2005). The harmony search heuristic algorithm for discrete structural optimization. Eng. Optimization. 37(7): 663–684.