Automation in Construction 69 (2016) 21–33

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

A Hybrid Harmony Search algorithm for discrete sizing optimization of truss structure

Min-Yuan Cheng [a], Doddy Prayogo [a,b,*], Yu-Wei Wu [a], Martin Marcellinus Lukito [a]

[a] Dept. of Civil and Construction Engineering, National Taiwan University of Science and Technology, #43, Sec. 4, Keelung Rd., Taipei 106, Taiwan, ROC
[b] Dept. of Civil Engineering, Petra Christian University, Jalan Siwalankerto 121–131, Surabaya 60236, Indonesia

ARTICLE INFO

Article history:
Received 17 August 2015
Received in revised form 20 April 2016
Accepted 22 May 2016
Available online 2 June 2016

Keywords:
Harmony Search algorithm
Global-best PSO
Neighbourhood search
Hybrid Harmony Search algorithm
Discrete sizing optimization of truss structures

ABSTRACT

This paper presents a new variant of the Harmony Search (HS) algorithm. This Hybrid Harmony Search (HHS) algorithm follows a new approach to improvisation: while retaining HS algorithm Harmony Memory and pitch adjustment functions, it replaces the HS algorithm randomization function with Global-best Particle Swarm Optimization (PSO) search and neighbourhood search. HHS algorithm performance is tested on six discrete truss structure optimization problems under multiple loading conditions. Optimization results demonstrate the excellent performance of the HHS algorithm in terms of both optimum solution and the convergence behaviour in comparison with various alternative optimization methods.

1. Introduction

Structural optimization has gained much attention because of its direct applicability to the design of structures [16]. Number of design variables, size of search area, and number of design constraints are factors that influence the time needed by designers to find optimized designs. Designers and owners desire optimized structures in order to reduce building structure costs [20]. Optimized structures should minimize the cost of a structure while meeting code-specified behaviour and performance requirements. Optimization allows to yield better designs at the lowest cost in terms of time and money.

Most recent studies on optimal structure designs have adopted continuous variables [7,8,21,23,33]. However, the availability of standard member sizes and precision limitations inherent in the modern steel manufacturing sector suggests to select cross-sectional areas from an available list of discrete values. Discrete optimization problems are far more difficult to solve than continuous problems [25,35]. Traditionally, researchers have used mathematical methods that employ rounding off techniques based on continuous solutions to solve discrete optimization problems. However, these methods may become infeasible or generate increasingly suboptimal solutions with larger numbers of variables [26]. This drawback has led researchers to rely on simulation-based metaheuristic algorithms to solve engineering optimization problems. Metaheuristic algorithms combine rules and randomness to imitate natural phenomena and try to find the optimum design using 'trial and error' in a reasonable amount of computing time [40]. The capability to balance intensification and diversification during a search determines the efficiency of a specific metaheuristic algorithm. Intensification (exploitation) aims to identify the best solution and select during the process a succession of best candidates/solutions. Diversification (exploration) ensures, usually by randomization, that the algorithm explores the search space efficiently. To address global search needs, modern metaheuristic algorithms have evolved to incorporate 3 main purposes: solving problems faster, solving larger problems, and enhancing algorithm robustness [5,13]. Modern metaheuristic algorithms include: Genetic Algorithms (GA) [18], Particle Swarm Optimization (PSO) [22], Differential Evolution (DE) [34], Artificial Bee Colony (ABC) [19], Bees Algorithm (BA) [29], Firefly Algorithm [39], Cuckoo Search (CS) [41], and Symbiotic Organisms Search (SOS) [4], among others.

Rather drawing its inspiration from biological or physical processes, the HS algorithm originally proposed in [14] is inspired by an artistic-creative process. The HS algorithm conceptualizes the behaviour of musicians searching for harmony and then continuing to refine their tune to achieve an increasingly better state of harmony. Musical harmony is analogous to an optimization solution vector and a musician's improvisations are analogous to local and global search schemes in optimization techniques. Due to its ease of application and simplicity, the HS algorithm has garnered growing attention and been successfully employed to a wide variety of practical structural optimization problem, such as truss structures [24], steel sway frames [9], grillage systems [32],

* Corresponding author at: Dept. of Civil Engineering, Petra Christian University, Jalan Siwalankerto 121–131, Surabaya 60236, Indonesia.
E-mail addresses: myc@mail.ntust.edu.tw (M.-Y. Cheng), doddyprayogo@ymail.com (D. Prayogo), d9305503@mail.ntust.edu.tw (Y.-W. Wu), martin.marcello@hotmail.com (M.M. Lukito).

http://dx.doi.org/10.1016/j.autcon.2016.05.023
0926-5805/

cellular beams [11], and web-expanded beams [10]. In comparison to earlier metaheuristic algorithms, the HS algorithm imposes fewer mathematical requirements and is easily adopted to solve various engineering optimization problems. In addition, this algorithm does not require initial values for decision variables, thus it may escape the local optima. The HS algorithm generates a new vector after considering all existing vectors based on the Harmony Memory Considering Rate (HMCR) and Pitch Adjusting Rate (PAR) rather than considering only two (parents) as in the Genetic Algorithm. Furthermore, instead of using gradient search, the HS algorithm uses stochastic random search based on HMCR and PAR, which obviates the need for derivative information [15]. These features increase HS algorithm flexibility and produce better solutions. Although several variants of the HS algorithm have been proposed, their effectiveness in dealing with diverse problems remains unsatisfactory [38].

While the HS algorithm is good at identifying the high performance regions of a solution space in a reasonable amount of time, this algorithm performs local searches for numerical applications poorly [28]. To improve the local search ability of the HS algorithm, this paper proposes a new algorithm called the Hybrid Harmony Search (HHS) algorithm. The HHS algorithm integrates the memory consideration and pitch adjustment process of the HS algorithm with Global-best PSO and neighbourhood search. Six classical truss design problems with sizing variables are solved in this study in order to demonstrate the efficiency of the HHS algorithm. It is shown that the present algorithm is very competitive with other optimization methods documented in literature.

The remainder of this paper is organized as follows: Section 2 presents the formulation of the discrete sizing optimization problem; Section 3 briefly reviews the HS and IHS algorithm; Section 4 describes the HHS algorithm in detail; Section 5 describes the test problems and discusses the optimization results; and Section 6 presents conclusions and recommendations for future research.

2. Discrete structural optimization problems

Since many problems in engineering have multiple solutions selecting. Discrete sizing optimization attempts to find the optimal cross-section of system elements in order to minimize structural weight. However, the minimum design must also satisfy inequality constraints that limit design variable sizes and structural responses [25].

The discrete structural optimization problem for a truss structure may be formulated as:

Find $A = [A_1, A_2, \ldots, A_{ng}]$,
$A_i \in D_i, D_i = [d_{i,1}, d_{i,2}, \ldots d_{i,r(i)}]$

To minimize

$$W(A) = \sum_{i=1}^{nm} \gamma_i \times A_i \times L_i \quad (1)$$

Subject to $\sigma_{min} \le \sigma_i \le \sigma_{max} \quad i = 1, 2, \ldots, n$
$\delta_{min} \le \delta_i \le \delta_{max} \quad i = 1, 2, \ldots, m$

where: A represents the set of design variables; Di is an allowable set of discrete values for design variable A_i; ng is the number of design variables or member groups; r(i) is the number of available discrete values for the ith design variable; W(A) is the weight of the structure; n is the number of component members in the structure; m is the number of nodes; γ_i is the material density of member i; L_i is the length of member i; δ_i is the nodal displacement/deflection at node i; σ_i is the stress developed in the i-th element; and min and max represent the lower and upper bounds, respectively.

The optimum design of truss structures must satisfy optimization constraints stated in Eq. (1). This paper uses a constraint handling procedure developed by Deb [6] to handle the problem-specific constraints. This procedure consists of the following three rules:

- Rule 1: Any feasible solution is preferred to any infeasible solution.
- Rule 2: Between two feasible solutions, the one having the better objective function value is preferred.
- Rule 3: Between two infeasible solutions, the one having the smaller constraint violation is preferred.

The first and third rules orient the search toward feasible regions. The second rule orients the search toward the feasible region with good solutions.

3. Harmony Search algorithm

3.1. Harmony Search algorithm

Harmony Search (HS) algorithm is a metaheuristic algorithm that imitates the natural music improvisation process that musicians use to achieve a perfect state of harmony such as that achieved during jazz improvisation. The HS algorithm holds several important advantages over other competing algorithms and has been applied successfully to a wide variety of optimization problems. Key advantages include ability to handle both discrete and continuous variables, conceptual simplicity, ease of implementation, and few parameters requiring adjustment.

The HS algorithm uses an optimization process to attain a global solution defined by an objective function similar to the way musicians attain aesthetic harmony as defined by an aesthetic standard. Each musician corresponds to one decision variable; the pitch range of musical instruments corresponds to the value range of the decision variable; musical harmony at a certain time corresponds to the solution vector at a certain iteration; and audience aesthetics correspond to the objective function.

In musical improvisation, each players plays at any pitch within the possible range, creating one harmony vector. If all pitches are in good harmony, the experience is stored in the memory of each player and the possibility of creating good harmony increases in the subsequent timeframe. In engineering optimization, each decision variable initially chooses any value within the possible range to create a solution vector. If all variable values create a good solution, the design is stored in the memory of each variable and the possibility of creating a good solution increases in the subsequent timeframe.

When a musician improves the musical harmony, he or she has three possible options: (1) playing any known tune exactly from memory, (2) playing a tune similar to a known tune, (3) composing a new tune at random. These three options correspond to the three main HS algorithm concepts of: memory consideration, pitch adjustment, and randomization. In general, the HS algorithm procedure consists of 5 steps:

Step 1. Initialize the problem and algorithm initial parameters.

The optimization problem is defined as Minimize f(x) subject to $LB_i \le Xi \le UB_i$, in which i = 1, 2,...., N. LB_i and UB_i are the lower and upper bounds for the decision variables. This step also specifies the HS algorithm parameters, including Harmony Memory Size (HMS), Harmony Memory Consideration Rate (HMCR), Pitch Adjusting Rate (PAR), bandwidth (bw), and number of improvisations (NI) or stopping criterion. The NI equals the total number of function evaluations. A good set of parameters will improve the ability of the algorithm to search for the global optimum with a high convergence rate.

Step 2. Initialize the Harmony Memory (HM).

The second step is Harmony Memory initialization. The HS algorithm has memory storage, called Harmony Memory (HM), in which

all solution vectors (decision variable sets) are stored. In this step, the HM matrix is filled with as many randomly generated solution vectors as the HMS. The initial HM is generated using a uniform distribution in the range [LBi, UBi], where 1 < i < N, with Eq. (2).

$$x_i^j = LB_i + \text{rand}() \times (UB_i - LB_i) \quad , \quad j = 1, 2, \ldots, HMS; \quad \text{where rand} \sim U(0,1) \tag{2}$$

where LB_i and UB_i are the lower and upper bounds of the ith decision variable, respectively, and rand is a uniformly distributed random number generated for each value of j. Fig. 1 illustrates the HM form. Candidate solution vectors in HM are then analysed, after which objective function values are calculated ($f(x_{i,:})$, i = 1, 2,…., HMS) and sorted by objective function f(x) value.

Step 3. Improvise a new harmony from the HM.

Generating new harmony is called improvisation. The new harmony vector x′ = (x′1, x′2,…., x′N) is generated using three rules: memory consideration, pitch adjustment, and random selection. The procedure works as follows [14]:

```
for each i ϵ [1,N] do
      if U(0,1) < HMCR then
            X'i = X^j_i (j = 1,2,...,HMS)   % memory consideration
            if U(0,1) < PAR then
                  X'i = X'i + r x bw   % pitch adjustment
                  if X'i < LBi
                        X'i = LBi
                  elseif X'i > UBi
                        X'i = UBi
                  end
            end
      else
            X'i = LBi + rand() x (UBi - LBi) % random selection
      end
end
```

where r and rand() are uniformly generated random numbers in the region of (0,1) and bw is an arbitrary distance bandwidth.

Step 4. Update Harmony Memory.

This step updates the HM. If the objective function value of the new improvised harmony vector is better than the worst harmony stored in HM, this vector replaces the worst harmony in the HM, after which HM is resorted according to objective function value.

Step 5. Terminate the process.

The HS algorithm is terminated when the stopping criterion (maximum number of improvisations [NI]) is satisfied. Otherwise, steps 3 and 4 are repeated.

3.2. The improved Harmony Search (IHS) algorithm

Prior studies have worked to improve HS algorithm performance and eliminate the limitations associated with fixed values for PAR and bw. In [28], it was developed a new HS algorithm variant called improved Harmony Search (IHS). Each step in this algorithm is the same as the original HS Algorithm with the exception of Step 3, which dynamically updates PAR and bw parameter using (3) and (4), respectively.

$$PAR(t) = PAR_{min} + \frac{PAR_{max} - PAR_{min}}{NI} \times t \tag{3}$$

where, PAR(t) is the pitch adjustment rate in generation t and PAR_{min} and PAR_{max} are the minimum and maximum pitch adjustment rates, respectively.

$$bw(t) = bw_{max} \exp(c \times t)$$
$$c = \frac{\text{Ln}\left(\frac{bw_{min}}{bw_{max}}\right)}{NI} \tag{4}$$

where, bw(t) is the distance bandwidth in generation t, bw_{min} is the minimum bandwidth, and bw_{max} is the maximum bandwidth.

4. The Hybrid Harmony Search (HHS) algorithm

Intensification and diversification are two critical components in modern metaheuristic algorithms. Intensification (exploitation) intensifies a local search in the neighbourhood of an optimal or near-optimal solution. Diversification (exploration) involves the global search to ensure that an algorithm explores the search space efficiently and effectively. Excessive diversification causes solutions to jump around within a potentially optimal solution and increase the convergence time to the optimum. On the other hand, excessive intensification may trap the algorithm in local optima because only a fraction of local space may be visited [40]. Therefore, a good algorithm requires a proper balance between these two components to ensure fast and efficient convergence, prevent trapping the algorithm in a local optima, and guarantee solution quality.

Although the HS algorithm has demonstrated its ability to find near global regions within a reasonable time, it is comparatively inefficient in performing local searches [12]. This paper thus proposed the HHS algorithm as a new and superior HS algorithm variant to improve HS algorithm local search capabilities and balance associated intensification and diversification components. The HHS algorithm introduces a new improvisation scheme. In addition to the concepts of Harmony Memory and Pitch Adjustment, HHS algorithm employs two new features to ensure that the new harmony imitates the global best harmony of the HM. These two features are Global-best PSO search and neighbourhood search. Furthermore, HHS algorithm omits the randomization function of the original HS algorithm.

$$HM = \left[\begin{array}{cccccc|c} x_{1,1} & x_{1,2} & \cdots & \cdots & x_{1,N-1} & x_{1,N} & f(x_{1,:}) \\ x_{2,1} & x_{2,2} & \cdots & \cdots & x_{2,N-1} & x_{2,N} & f(x_{2,:}) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{HMS-1,1} & x_{HMS-1,2} & \cdots & \cdots & x_{HMS-1,N-1} & x_{HMS-1,N} & f(x_{HMS-1,:}) \\ x_{HMS,1} & x_{HMS,2} & \cdots & \cdots & x_{HMS,N-1} & x_{HMS,N} & f(x_{HMS,:}) \end{array}\right]$$

Fig. 1. Structure of the Harmony Memory matrix.

Global-best PSO was inspired by Particle Swarm Optimization (PSO). PSO is an optimization procedure based on social organizations found in nature such as bird flocks and fish schools [22]. In PSO, a population of particles starts to move in a search space by following current optimum particles and changing their positions to find the best particle. In other words, PSO directs its particles to move toward good areas in a search space in response to information distributed through the swarm [3, 27]. Neighbourhood search, the second feature added to the HHS algorithm, provides powerful capabilities to search in the neighbourhood of the selected particle. These searches guide particles to search in more promising (optimum or near-optimum) areas. Appropriately integrating Global-best PSO and neighbourhood search increases HHS algorithm efficiency. Fig. 2 shows the HHS algorithm flowchart.

The following five subsections present the 5 steps of the HHS algorithm.

4.1. Initialize the problem and algorithm parameters

This step specifies algorithm parameter, including Harmony Memory Size (HMS), Harmony Memory Consideration Rate minimum ($HMCR_{min}$), Harmony Memory Consideration Rate maximum ($HMCR_{max}$), Pitch Adjustment Rate minimum (PAR_{min}), Pitch Adjustment Rate maximum (PAR_{max}), bandwidth minimum (bw_{min}), bandwidth maximum (bw_{max}), Global-Best Rate (GBR), and number of improvisations (NI) or stopping criterion. GBR, a new parameter introduced into the HHS algorithm, determines the probabilities of improvising new harmony using Global-best PSO search and neighbourhood search, respectively. Subsection 4.3 provides a detailed explanation.

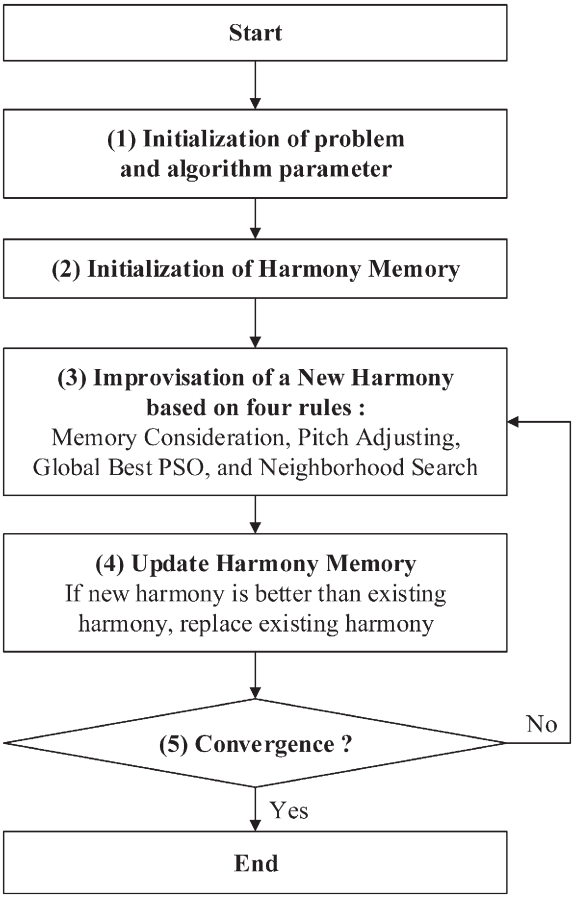

Fig. 2. Flowchart of the HHS algorithm.

4.2. Initialize the Harmony Memory (HM)

As in the HS Algorithm, in this step, the Harmony Memory matrix is filled with the same number of randomly generated solution vectors as in the HMS. To create the initial memory, random values between the lower and upper bounds of the decision variable are assigned to each decision parameter of each memory vector, as follows.

$$x_i^j = LB_i + \text{rand}() \times (UB_i\text{-}LB_i) \quad , \quad j = 1, 2, \ldots, HMS; \quad \text{where rand} \sim U(0,1) \tag{5}$$

where LB_i and UB_i are lower and upper bounds of the ith decision variable, respectively, and rand is a uniformly distributed random number generated for each value of j. After initialization, the objective function values are evaluated and then the HM are sorted by objective function (fx) value.

4.3. New harmony improvisation

This step generates a new harmony vector $x' = (x'1, x'2, \ldots, x'N)$ based on four rules: memory consideration, pitch adjustment, Global-best PSO search, and neighbourhood search. As the HS algorithm, the HHS algorithm produces one new 'child' during each iteration. In other words, the number of iterations in the HHS algorithm equals the number of evaluations. Fig. 3 shows the new harmony improvisation flowchart.

4.3.1. Memory consideration

The value of a design variable may be chosen from the values stored in HM using the Harmony Memory Consideration Rate (HMCR). The HMCR, which varies between 0 and 1, is the probability of selecting one value from historical values in the specified HM range. The (1-HMCR) in the HHS algorithm is the probability of improvising new harmony based on the Global-best PSO search or neighbourhood search, as shown in the following pseudo-code.

```
For j = 1 to number of decision variables (N) do
    r1 = uniform random number between 0 and 1
    if (r1 < HMCR) (memory consideration)
       x(j) will be randomly chosen from HM
    else
       Do Global best PSO search and neighbourhood search
    end if
end for
```

4.3.2. Pitch adjustment

Each component obtained using the memory consideration is checked for pitch adjustment need by moving to a neighbour value of a selected value from the HM with a pitch adjustment rate (PAR) probability. The pseudocode for the pitch adjustment process, where r2 is a uniform random number between 0 and 1 and bw is an arbitrary distance bandwidth, is shown as follows.

```
r2 = uniform random number between 0 and 1
if (r2 < PAR) (pitch adjustment)
    r3 = uniform random number between -1 and 1
    x(j) = x(j) + r3*bw
end
```

4.3.3. Global-best PSO search and neighbourhood search

In the original HS algorithm, (1-HMCR) refers to the probability of randomly selecting one value from the range of possible values.

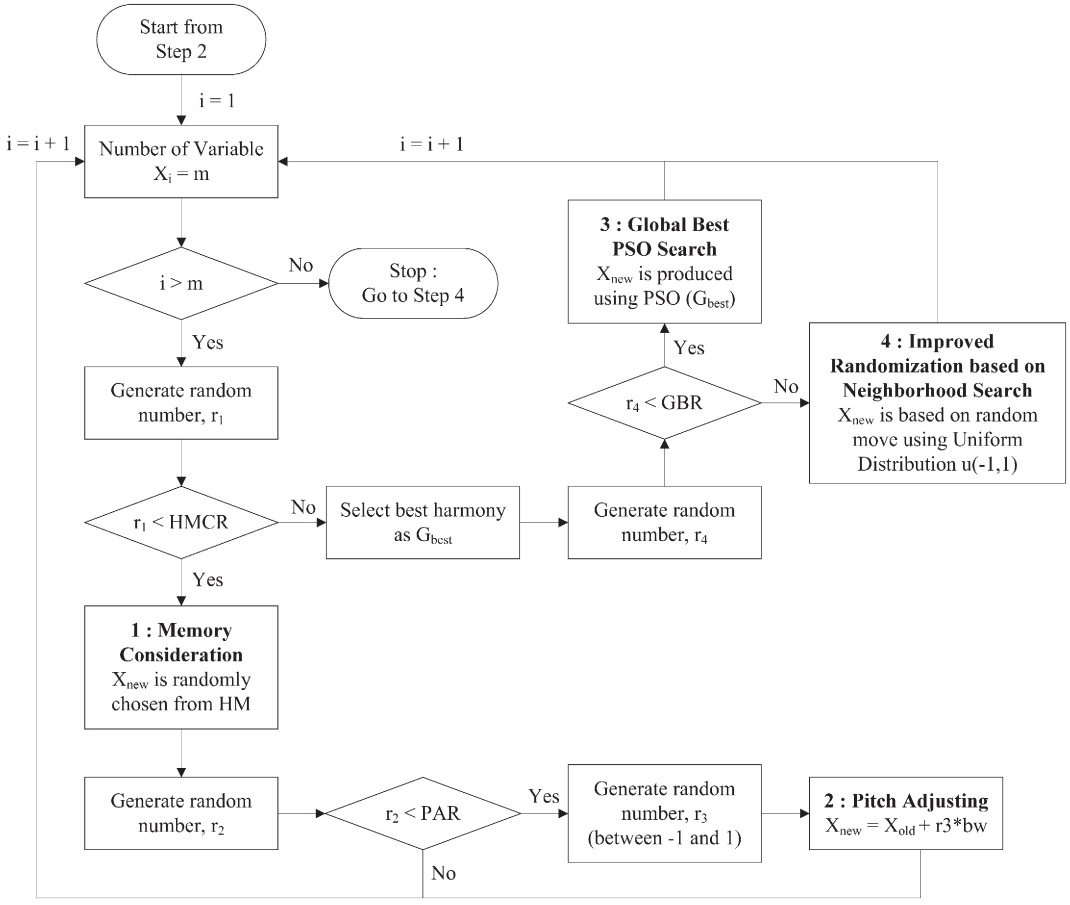

Fig. 3. Flowchart of new harmony improvisation in the HHS algorithm (Step 3).

However, in the HHS algorithm, this randomization process is omitted and replaced with G_{best}, the best harmony of the decision variable. This step uses the existing best harmony to improvise a new harmony. Thereafter, there are two options available determined by the Global-best Rate (GBR). GBR is the probability of generating one harmony by imitating the behaviour of the Global-best PSO and (1-GBR) is the probability of generating one harmony from the neighbourhood of the G_{best}. These two options increase the probability of the HHS algorithm to obtain the optimum value of the design variable.

4.3.3.1. Global best PSO search. If rand < GBR, the HHS algorithm imitates the performance of Global-best PSO. It should be noted that, as in PSO, the purpose of this step is to make the target variable move in the direction of the global best. First, we must randomly determine the variable to be moved. Next, Eq. (6) calculates a new velocity, where x_m = the variable to be moved and rand = a uniform random number between 0 and 1. Eq. (7) then calculates the new position.

$$\text{New Velocity}(j) = \text{rand}() \times (G_{best}(j) - x_m(j)) \tag{6}$$

$$x(j) = x_m(j) + \text{New Velocity}(j) \tag{7}$$

4.3.3.2. Improved randomization based on neighbourhood search. If (rand > GBR), the HHS algorithm employs a neighbourhood search to spread the new harmony through the neighbourhood of G_{best}, as shown in Eq. (8), where rand = uniform random number between 0 and 1.

$$x(j) = G_{best}(j) \times (\text{rand}() - 0.5) \times 2 + G_{best}(j) \tag{8}$$

The pseudocode for these steps can be described as follows.

```
select best harmony as Gbest
r4 = uniform random number between 0 and 1
if (r4 < GBR)(Global Best PSO Search)
    random 1 variable to be moved (xm(j))
    new velocity (j) = rand*(Gbest(j)-xm(j))
    x(j) = xm(j) + new velocity(j)
else (Random based on Neighbourhood Search)
    x(j) = Gbest(j)*(rand-0.5)*2 + Gbest(j)
end if
```

The second modification allows the HHS algorithm to update dynamically the value of HMCR using Eq. (9), with the values of $HMCR_{min}$ and $HMCR_{max}$ derived from our experiments 0.1 and 0.9, respectively. Conversely, the original HS algorithm and other HS algorithm variants apply a fixed value to the HMCR.

$$HMCR(t) = HMCR_{min} + \frac{HMCR_{max} - HMCR_{min}}{NI} \times t \tag{9}$$

where HMCR(t) = HMCR for generation t. PAR and bw parameter of the HHS algorithm are dynamically updated with Eqs. (3) and (4) in the IHS algorithm [28].

These principles enable the HHS algorithm to balance exploration and exploitation capabilities. In the early stage, the HHS algorithm focuses on global search using memory consideration and pitch adjustment, while, in later stages, the HHS algorithm focuses on local search

using Global-best PSO search and neighbourhood search. The complete pseudocode for the improvisation step of the HHS algorithm is presented as follows.

```
For j = 1 to number of decision variables (N) do
    r1 = uniform random number between 0 and 1
    if (r1 < HMCR) (memory consideration)
      x(j) will be randomly chosen from HM
      r2 = uniform random number between 0 and 1
      if (r2 < PAR) (pitch adjustment)
            r3 = uniform random number between -1 and 1
            x(j) = x(j) + r3*bw
      end if
    else
      select best harmony as Gbest
      r4 = uniform random number between 0 and 1
      if (r4 < GBR)(Global Best PSO Search)
           random 1 variable to be moved (xm(j))
           new velocity (j) = rand*(Gbest(j)-xm(j))
           x(j) = xm(j) + new velocity(j)
      else (Random based on Neighbourhood Search)
           x(j) = Gbest(j)*(rand-0.5)*2 + Gbest(j)
      end if
      constraint handling
    end if
end for
```

4.4. Update Harmony Memory

If the newly improvised harmony vector is better than the worst harmony vector stored in HM, the new harmony vector is included and the worst harmony is excluded from the HM. Subsequently, the HM is sorted by objective function value.

4.5. Terminate the optimization process

The HHS algorithm is terminated when the termination criterion (maximum number of improvisations [NI]) is satisfied. Otherwise, steps 3 and 4 are repeated.

5. Test problems and optimization results

In order to investigate efficiency of the HHS algorithm, we solved the weight minimization problems of three planar and two spatial bar truss structures under multiple loading conditions using discrete variables. The algorithms were coded in Matlab and structures were analysed using the finite element (direct stiffness) method. Optimization results were compared to the results obtained by other optimization methods in order to demonstrate HHS algorithm efficiency. HHS internal parameters were set as follows: Harmony Memory Size (HMS) was set to 10; HMCR increased linearly from 0.1 to 0.9; PAR_{min} was set to 0.4; PAR_{max} was set to 0.9; bw_{min} and bw_{max} were set to 0.0001 and 1, respectively;

Table 1
Comparison of HHS optimization results with the literature for the 10-bar truss problem (Case 1).

Variables (in^2)	GA [30]	HPSO [26]	SA [37]	BB–BC [2]	HHS
A_1	33.50	30.00	33.50	33.50	33.50
A_2	1.62	1.62	1.62	1.62	1.62
A_3	22.00	22.90	22.90	22.90	22.90
A_4	15.50	13.50	14.20	14.20	14.20
A_5	1.62	1.62	1.62	1.62	1.62
A_6	1.62	1.62	1.62	1.62	1.62
A_7	14.20	7.97	7.97	7.97	7.97
A_8	19.90	26.50	22.90	22.90	22.90
A_9	19.90	22.00	22.00	22.00	22.00
A_{10}	2.62	1.80	1.62	1.62	1.62
Best (lb)	5613.84	5531.98	5490.74	5490.74	5490.74
Average (lb)	N/A	N/A	N/A	5494.17	5493.489
Stdev (lb)	N/A	N/A	N/A	12.420	10.463
No. of analyses	800	50,000	10,500	8694	5000
Constraint violation	0.03%	None	None	None	None

Table 2
Comparison of HHS optimization results with the literature for the 10-bar truss problem (Case 2).

Variables (in^2)	Ringertz [31]	HPSO [26]	HHS
A_1	30.50	31.50	30.50
A_2	0.10	0.10	0.10
A_3	23.00	24.50	24.00
A_4	15.50	15.50	14.00
A_5	0.10	0.10	0.10
A_6	0.50	0.50	0.50
A_7	7.50	7.50	7.50
A_8	21.0	20.50	21.50
A_9	21.5	20.50	21.50
A_{10}	0.10	0.10	0.10
Best (lb)	5059.9	5073.51	5067.33
Average (lb)	N/A	N/A	5068.36
Stdev (lb)	N/A	N/A	2.343
No. of analyses	N/A	50,000	5000
Constraint violations	0.04%	None	None

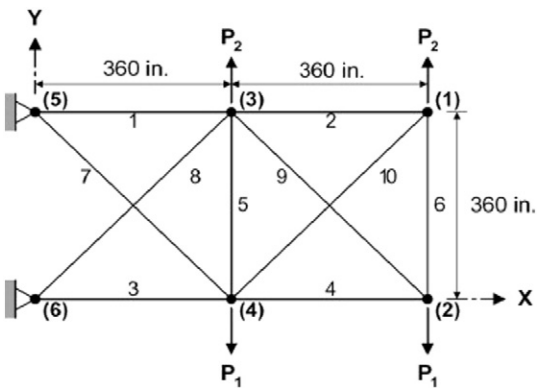

Fig. 4. Schematic of the planar 10-bar truss structure.

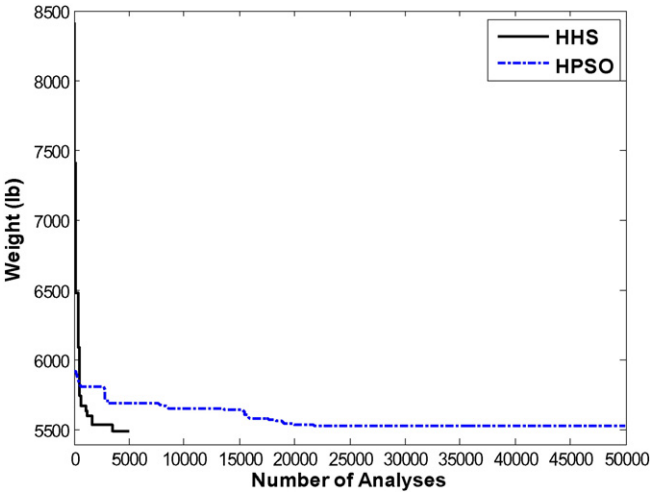

Fig. 5. Convergence curves for the planar 10-bar truss structure (Case 1).

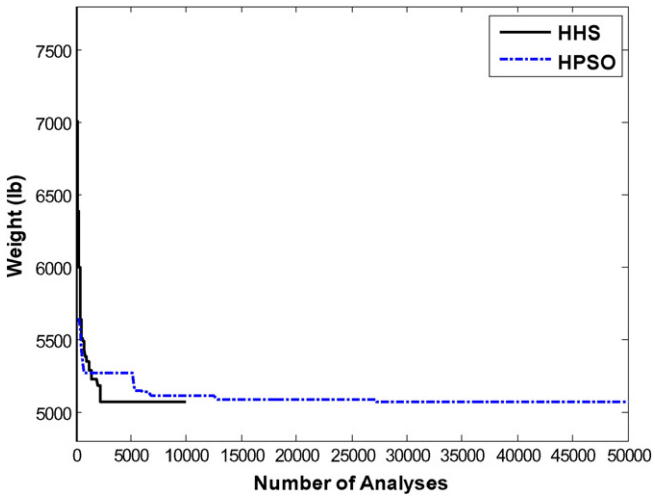

Fig. 6. Convergence curves for the planar 10-bar truss structure (Case 2).

Table 3
Comparison of HHS optimization results with the literature for the 15-bar truss problem.

Variables (mm^2)	Improved-GA [42]	HPSO [26]	HHS
A_1	308.6	113.2	113.2
A_2	174.9	113.2	113.2
A_3	338.2	113.2	113.2
A_4	143.2	113.2	113.2
A_5	736.7	736.7	736.7
A_6	185.9	113.2	113.2
A_7	265.9	113.2	113.2
A_8	507.6	736.7	736.7
A_9	143.2	113.2	113.2
A_{10}	507.6	113.2	113.2
A_{11}	279.1	113.2	113.2
A_{12}	174.9	113.2	113.2
A_{13}	297.1	113.2	113.2
A_{14}	235.9	334.3	334.3
A_{15}	265.9	334.3	334.3
Best (lb)	142.117	105.735	105.735
Average (lb)	N/A	N/A	106.157
Stdev (lb)	N/A	N/A	1.095
No. of analyses	N/A	25,000	5000
Constraint violations	None	None	None

and GBR was set to 0.5. The limit number of analyses was set to 5000 for each design problem. Thirty independent runs were performed for each design problem starting from different initial populations. It should be noted that one iteration in HPSO [26] entailed 50 structural analyses: therefore 1000 iterations in the convergence curve correspond to 50,000 structural analyses.

5.1. Planar 10-bar truss structure

The 10-bar truss structure shown in Fig. 4 is one of the most popular test problems in structural optimization, previously solved in [30,31,2,37,26]. Fig. 4 shows the geometry and support conditions for this two dimensional, cantilevered truss with loading condition. The material density is 0.1 lb/in^3 and the modulus of elasticity is 10,000 ksi. The allowable displacement for all nodes in both vertical and horizontal directions equals ±2.0 in. Members are subjected to stress limitations of ±25 ksi for both tension and compression. Both loads, $P_1 = 10^5$ lb and $P_2 = 0$, are considered. There are 10 design variables and two cases to study in this problem.

For the first case, we selected discrete variables from the set: $D =$ [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).

For the second case, we selected discrete variables from the set: $D =$ [0.1, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.0, 9.5, 10.0, 10.5, 11.0, 11.5, 12.0, 12.5, 13.0, 13.5, 14.0, 14.5, 15.0, 15.5, 16.0, 16.5, 17.0, 17.5, 18.0, 18.5, 19.0, 19.5, 20.0, 20.5, 21.0, 21.5, 22.0, 22.5, 23.0, 23.5, 24.0, 24.5, 25.0, 25.5, 26.0, 26.5, 27.0, 27.5, 28.0, 28.5, 29.0, 29.5, 30.0, 30.5, 31.0, 31.5] (in^2).

Tables 1 and 2 compare HHS optimization results with literature. In Case 1, HHS found an optimum weight of 5490.74 lb after 3533 structural analyses. Optimum weights found by BB–BC, SA, HPSO, and GA were 5490.74, 5490.74, 5531.98 lb, and 5613.84 lb, respectively. Although. HHS algorithm obtained the same weight with SA and BB–BC, it required less structural analyses than the other reported algorithms. Meanwhile, HPSO required over 20,000 structural analyses to obtain the optimum weight of 5531.98 lb whereas the HHS algorithm obtained the same using only 2835 structural analyses.

In Case 2, the HHS algorithm obtained the second-best design after [31]. The HHS algorithm result was 0.147% heavier than the best design. The HHS algorithm obtained the optimum weight of 5067.33 lb after 2291 structural analyses. HPSO obtained the optimum weight of 5073.51 lb after more than 25,000 structural analyses, while the HHS algorithm obtained the same weight in 2057 structural analyses. Figs. 5

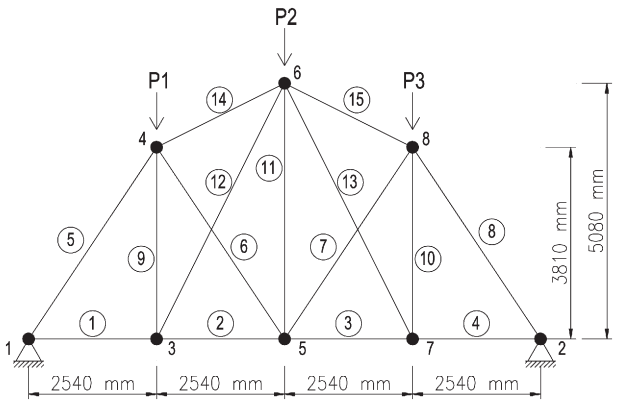

Fig. 7. Schematic of the planar 15-bar truss structure.

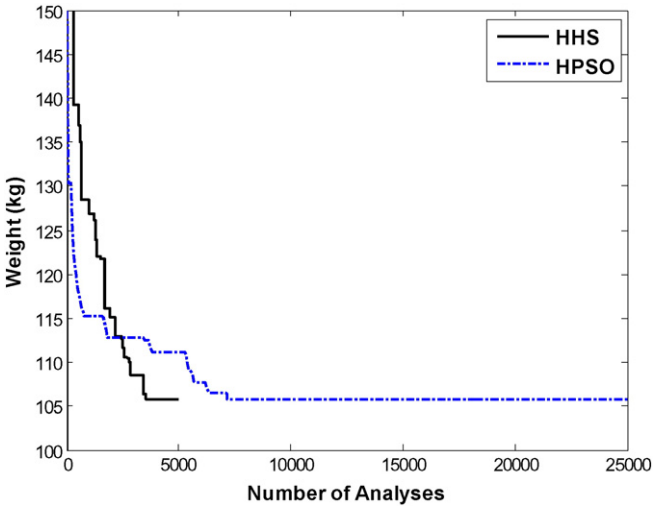

Fig. 8. Convergence curves for the planar 15-bar truss structure.

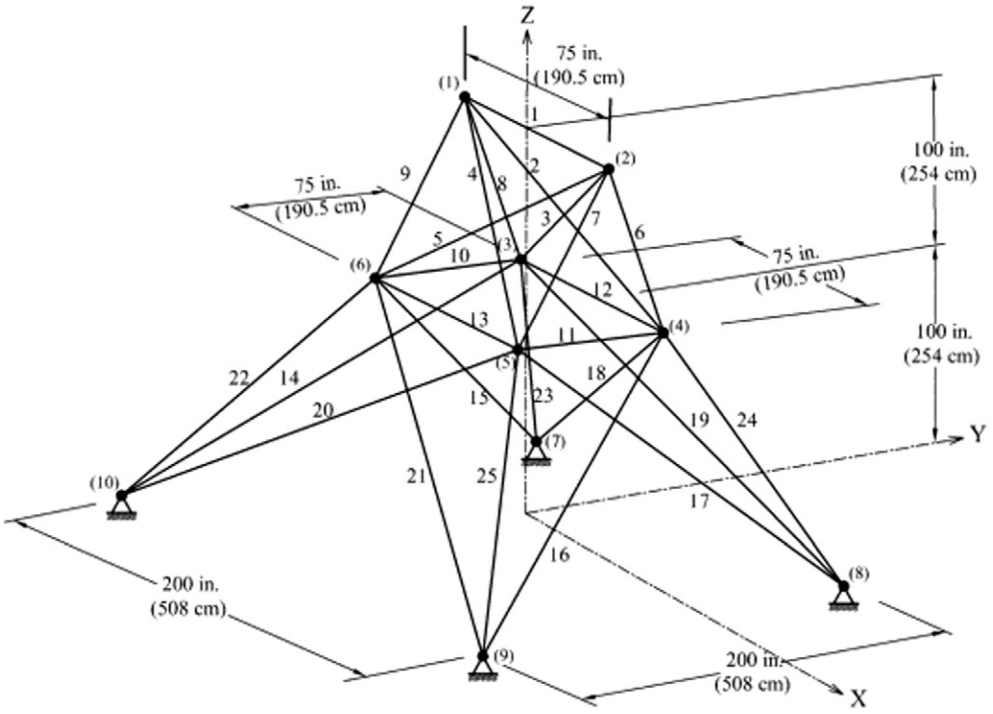

Fig. 9. Schematic of the spatial 25-bar truss structure.

Table 4
Loading condition for Case 1 of the 25-bar truss problem.

	Load cases	Nodes	Loads		
			P_x (kips)	P_y (kips)	P_z (kips)
Case 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

and 6 confirm the good convergence behaviour of HHS in the 10-bar design problem.

5.2. Planar 15-bar truss structure

Fig. 7 shows the geometry and the loading condition of the planar 15-bar truss structure, previously studied in [42,26]. The material density is 7800 kg/m^3 and the modulus of elasticity is 200 GPa. The maximum allowable stress for all members is 120 MPa (the same for tension and compression) and maximum displacements for all free nodes in the X and Y directions must not exceed 10 mm. This test problem includes 15 discrete design variables that can be selected from the following discrete set:

Table 5
Loading conditions for Case 2 of the 25-bar truss problem.

	Load cases	Nodes	Loads		
			P_x (kips)	P_y (kips)	P_z (kips)
Case 2	2	1	0.0	20.0	−5.0
		2	0.0	−20.0	−5.0
	3	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

Table 6
Comparison of HHS optimization results with the literature for the 25-bar truss problem (Case 1).

Variables (mm^2)	GA [30]	HS [25]	HPSO [26]	SA [37]	BB–BC [2]	HHS
A_1	0.1	0.1	0.1	0.1	0.1	0.1
A_2–A_5	1.8	0.3	0.3	0.3	0.3	0.3
A_6–A_9	2.3	3.4	3.4	3.4	3.4	3.4
A_{10}–A_{11}	0.2	0.1	0.1	0.1	0.1	0.1
A_{12}–A_{13}	0.1	2.1	2.1	2.1	2.1	2.1
A_{14}–A_{17}	0.8	1.0	1.0	1.0	1.0	1.0
A_{18}–A_{21}	1.8	0.5	0.5	0.5	0.5	0.5
A_{22}–A_{25}	3.0	3.4	3.4	3.4	3.4	3.4
Best (lb)	546.01	484.85	484.85	484.85	484.85	484.85
Average (lb)	N/A	N/A	N/A	N/A	485.10	484.946
Stdev (lb)	N/A	N/A	N/A	N/A	0.44	0.365
No. of analyses	800	13,523	25,000	7900	9090	5000
Constraint violation	None	None	None	None	None	None

Table 7
Comparison of HHS optimization results with the literature for the 25-bar truss problem (Case 2).

Variables (mm^2)	Ringertz [31]	HS [25]	HPSO [26]	DHPSACO [21]	HHS
A_1	0.01	0.01	0.01	0.01	0.01
A_2–A_5	1.6	2.0	2.0	1.6	2.0
A_6–A_9	3.6	3.6	3.6	3.2	3.6
A_{10}–A_{11}	0.01	0.01	0.01	0.01	0.01
A_{12}–A_{13}	0.01	0.01	0.01	0.01	0.01
A_{14}–A_{17}	0.8	0.8	0.8	0.8	0.8
A_{18}–A_{21}	2.0	1.6	1.6	2.0	1.6
A_{22}–A_{25}	2.4	2.4	2.4	2.4	2.4
Best (lb)	568.69	560.59	560.59	551.61	560.59
Average (lb)	N/A	N/A	N/A	N/A	560.785
Stdev (lb)	N/A	N/A	N/A	N/A	0.743
No. of analyses	N/A	7435	25,000	25,000	5000
Constraint violation	None	None	None	0.10%	None

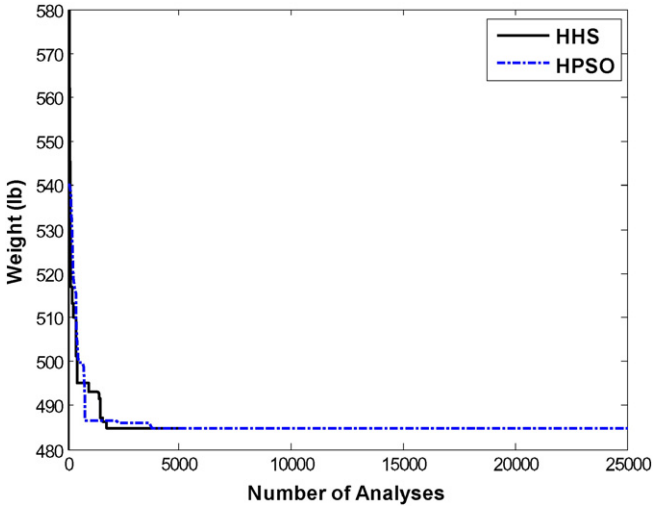

Fig. 10. Convergence curves for the spatial 25-bar truss structure (Case 1).

D = [113.2, 143.2, 145.9, 174.9, 185.9, 235.9, 265.9, 297.1, 308.6, 334.3, 338.2, 497.8, 507.6, 736.7, 791.2, 1063.7] (mm^2).

The structure is subject to three independent loading conditions: Load Case 1: $P_1 = 35$ kN, $P_2 = 35$ kN, $P_3 = 35$ kN; Load Case 2: $P_1 = 35$ kN, $P_2 = 0$ kN, $P_3 = 35$ kN; and Load Case 3: $P_1 = 35$ kN, $P_2 = 35$ kN, $P_3 = 0$ kN.

Table 3 shows the optimal result of the HHS algorithm with 15 size variables and compares these results with the literature. HPSO obtained a minimum weight of 105.735 lb in more than 7500 structural analyses, while the HHS algorithm obtained the same weight in only 3563 structural analyses. Fig. 8 confirms the better convergence rate of HHS that requires less structural analyses than the other reported algorithms.

5.3. Spatial 25-bar truss structure

The 25-bar transmission tower spatial truss shown in Fig. 9 has been studied by several researchers including [30,26,2,37,25]. The material of the structure has mass density of 0.1 lb/in^3 and elastic modulus of 10 Msi. The stress limits in tension/compression is ±40,000 psi and maximum nodal displacement is ±0.35 in. The structure includes 25 members categorized into the following 8 groups: (1) A_1, (2) A_2–A_5, (3) A_6–A_9, (4) A_{10}–A_{11}, (5) A_{12}–A_{13}, (6) A_{14}–A_{17}, (7) A_{18}–A_{21} and (8) A_{22}–A_{25}.

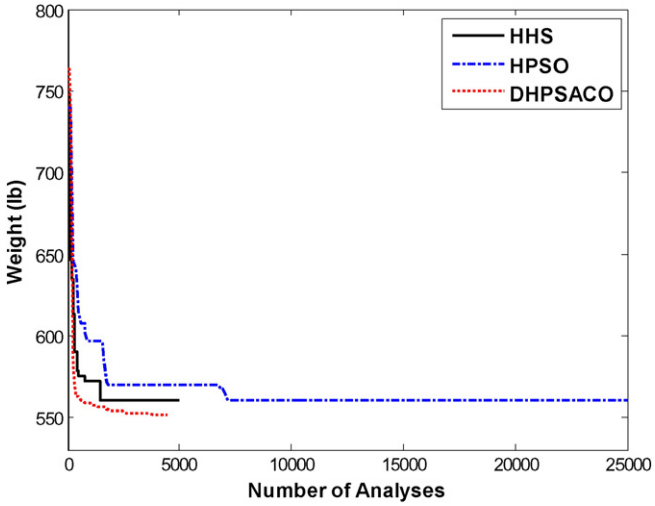

Fig. 11. Convergence curves for the spatial 25-bar truss structure (Case 2).

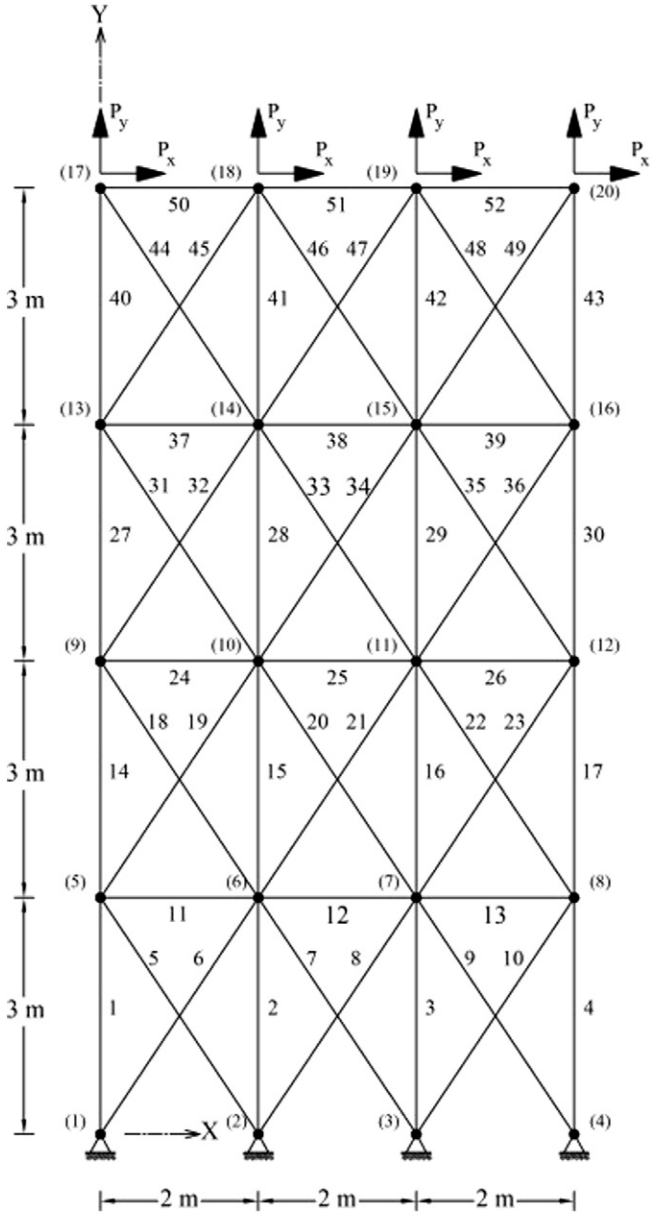

Fig. 12. Schematic of the planar 52-bar truss structure.

Two optimization cases were implemented, as follows.

Case 1. Discrete values of cross-sectional areas were selected from the following set: 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, 2.6, 2.8, 3.0, 3.2, 3.4] (in^2). Loads are shown in Table 4.

Case 2. Discrete values of cross-sectional areas were selected from the following set: D = [0.01, 0.4, 0.8, 1.2, 1.6, 2.0, 2.4, 2.8, 3.2, 3.6, 4.0, 4.4, 4.8, 5.2, 5.6, 6.0] (in^2). Loads are shown in Table 5.

Tables 6 and 7 compare the results obtained by the HHS algorithm with the other optimization methods reported in the literature. In Case 1, HPSO obtained a minimum weight of 484.85 lb in over 25,000 structural analyses. The HHS algorithm obtained the same weight in 1739 structural analyses. Table 7 shows that the HHS algorithm also obtained the second-best design, which weighed 1.7% more than

Table 8
Available cross-sectional areas from the AISC design code.

No.	in^2	mm^2	No.	in^2	mm^2
1	0.111	71.613	33	3.840	2477.414
2	0.141	90.968	34	3.870	2496.769
3	0.196	126.451	35	3.880	2503.221
4	0.250	161.290	36	4.180	2696.769
5	0.307	198.064	37	4.220	2722.575
6	0.391	252.258	38	4.490	2896.768
7	0.442	285.161	39	4.590	2961.284
8	0.563	363.225	40	4.800	3096.768
9	0.602	388.386	41	4.970	3206.445
10	0.766	494.193	42	5.120	3303.219
11	0.785	506.451	43	5.740	3703.218
12	0.994	641.289	44	7.220	4658.055
13	1.000	645.160	45	7.970	5141.925
14	1.228	792.256	46	8.530	5503.215
15	1.266	816.773	47	9.300	5999.988
16	1.457	939.998	48	10.850	6999.986
17	1.563	1008.385	49	11.500	7419.340
18	1.620	1045.159	50	13.500	8709.660
19	1.800	1161.288	51	13.900	8967.724
20	1.990	1283.868	52	14.200	9161.272
21	2.130	1374.191	53	15.500	9999.980
22	2.380	1535.481	54	16.000	10,322.560
23	2.620	1690.319	55	16.900	10,903.204
24	2.630	1696.771	56	18.800	12,129.008
25	2.880	1858.061	57	19.900	12,838.684
26	2.930	1890.319	58	22.000	14,193.520
27	3.090	1993.544	59	22.900	14,774.164
28	3.130	2019.351	60	24.500	15,806.420
29	3.380	2180.641	61	26.500	17,096.740
30	3.470	2238.705	62	28.000	18,064.480
31	3.550	2290.318	63	30.000	19,354.800
32	3.630	2341.931	64	33.500	21,612.860

DHPSACO, the best design. However, the best design using DHPSACO is considered to be infeasible because it has slightly violated the constraint with a violation percentage of 0.10%. Figs. 10 and 11 show that the HHS algorithm has better convergence behaviour than other reported methods.

5.4. Planar 52-bar truss structure

Fig. 12 shows t he geometry and loading condition of a planar truss structure consisting of 52 bars and twenty nodes. This problem was previously studied in [36,25,26]. The members of this structure are divided into 12 groups: (1) A_1–A_4, (2) A_5–A_{10}, (3) A_{11}–A_{13}, (4) A_{14}–A_{17}, (5) A_{18}–A_{23}, (6) A_{24}–A_{26}, (7) A_{27}–A_{30}, (8) A_{31}–A_{36}, (9) A_{37}–A_{39}, (10) A_{40}–A_{43}, (11) A_{44}–A_{49}, and (12) A_{50}–A_{52}. The material density is 7860.0 kg/m^3 and the modulus of elasticity is 207 GPa. The maximum allowable stress for all members in terms of both tension and compression is 180 MPa. Loads $P_x = 100$ kN and $P_y = 200$ kN are both considered. Discrete values of cross-sectional areas can be selected from Table 8.

Table 9
Comparison of HHS optimization results with the literature for the 52-bar truss problem.

Variables (mm^2)	HS [25]	HPSO [26]	DHPSACO [21]	HHS
A_1–A_4	4658.055	4658.055	4658.055	4658.055
A_5–A_{10}	1161.288	1161.288	1161.288	1161.288
A_{11}–A_{13}	506.451	363.225	494.193	494.193
A_{14}–A_{17}	3303.219	3303.219	3303.219	3303.219
A_{18}–A_{23}	940.000	940.000	1008.385	940.000
A_{24}–A_{26}	494.193	494.193	285.161	494.193
A_{27}–A_{30}	2290.318	2238.705	2290.318	2238.705
A_{31}–A_{36}	1008.385	1008.385	1008.385	1008.385
A_{37}–A_{39}	2290.318	388.386	388.386	494.193
A_{40}–A_{43}	1535.481	1283.868	1283.868	1283.868
A_{44}–A_{49}	1045.159	1161.288	1161.288	1161.288
A_{50}–A_{52}	506.451	792.256	506.451	494.193
Weight (kg)	1906.760	1905.495	1904.830	1902.605
Average (kg)	N/A	N/A	N/A	1904.587
Stdev (kg)	N/A	N/A	N/A	1.309
No. of analyses	19,104	150,000	5300	5000
Constraint violation	2.66%	None	0.27%	None

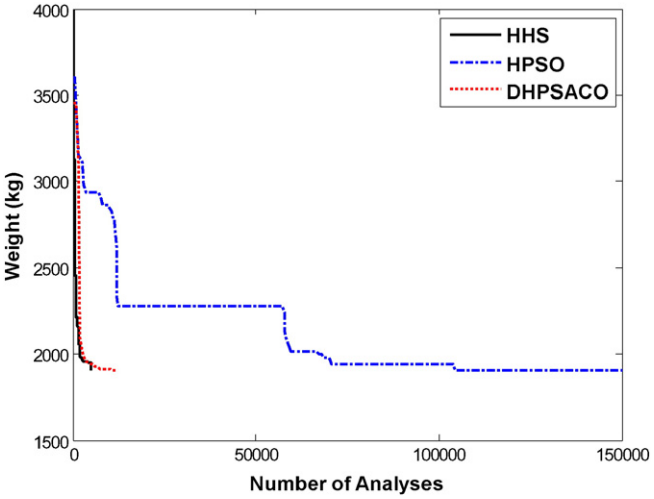

Fig. 13. Convergence curves for the planar 52-bar truss structure.

Table 9 shows the results obtained by the HHS algorithm and other optimization methods. The HHS algorithm converged to the best design overall corresponding to the structural weight of 1902.605 kg. The optimization process was completed in 4523 structural analyses. DHPSACO obtained the second best weight of 1904.83 kg after 5300 structural analyses, roughly twice as many as the 2639 structural analyses required by HHS. It is worth noting that the design using DHPSACO has violated the constraint with a violation percentage of 0.27%. Fig. 13 clearly shows the good convergence behaviour of HHS in the 52-bar design problem.

5.5. Spatial 72-bar truss structure

The 72-bar, 4-level tower, shown in Fig. 14, was optimized in [26,36]. Material density and modulus of elasticity were the same as those of the 10 and 25-bar truss structures. The stress limit in tension/compression was ±25,000 psi. The top nodes could not displace by more than ±0.25 in in both X and Y-directions. The structure is subjected to the two load cases described in Table 10.

The 72 structural members of this spatial truss are categorized into 16 groups using symmetry considerations: (1) A_1–A_4, (2) A_5–A_{12}, (3) A_{13}–A_{16}, (4) A_{17}–A_{18}, (5) A_{19}–A_{22}, (6) A_{23}–A_{30} (7) A_{31}–A_{34}, (8) A_{35}–A_{36}, (9) A_{37}–A_{40}, (10) A_{41}–A_{48}, (11) A_{49}–A_{52}, (12) A_{53}–A_{54}, (13) A_{55}–A_{58}, (14) A_{59}–A_{66} (15) A_{67}–A_{70}, (16) A_{71}–A_{72}. Discrete values of cross-sectional areas were selected from the following set:

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, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2] (in^2).

Table 11 compares the results obtained by the HHS algorithm and other optimization methods. Fig. 15 compares the corresponding convergence rates. In this case, the HHS algorithm obtained the same best weight as DHPSACO (385.54 lb). HPSO, HS, and GA obtained best weights of 388.94 lb, 387.94 lb, and 400.66 lb, respectively. The HHS algorithm obtained the best solution in 3294 structural analyses. The HPSO and DHPSACO algorithms obtained the best solution after 50,000 and 5330 analyses, respectively.

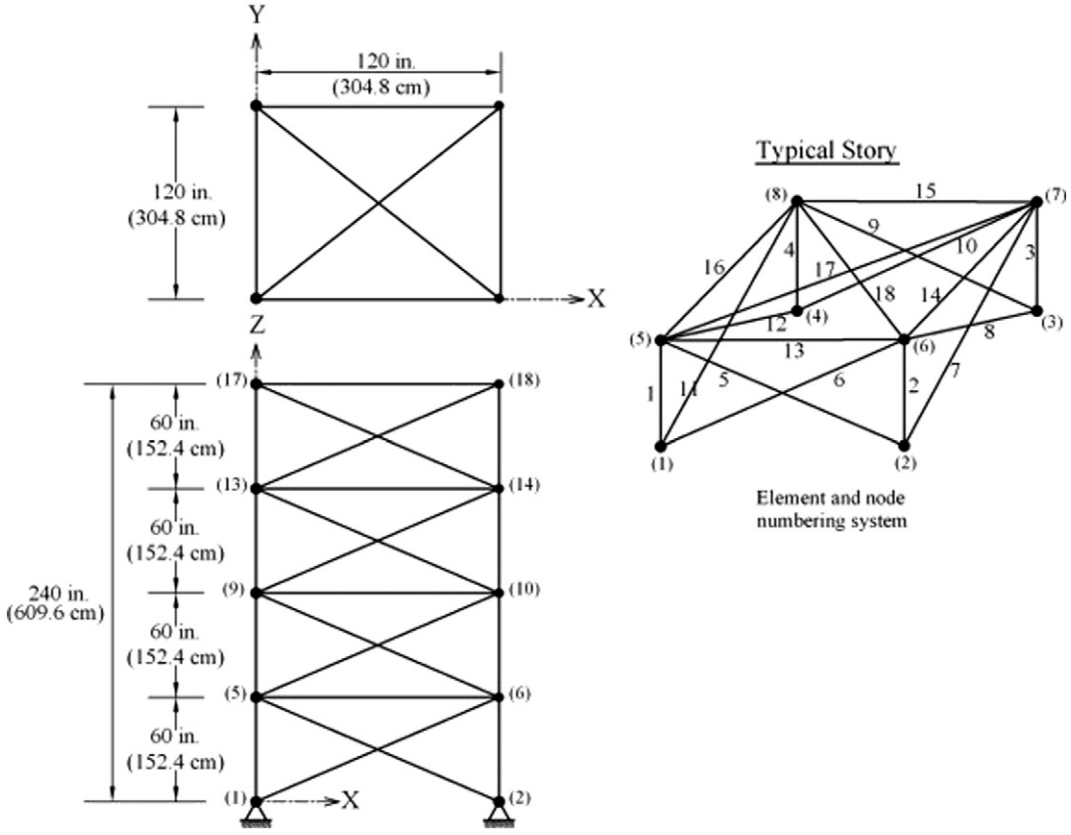

Fig. 14. Schematic of the spatial 72-bar truss structure.

5.6. Planar 200-bar truss structure

An 11-level planar truss structure consisting of 200 bars, shown in Fig. 16, was optimized in [1,17]. The material density is 0.283 lb/in^3 whereas the modulus of elasticity is 30,000 psi. The stress limit was ±10,000 psi. The structure is subjected to the two load cases: (1) One kip is applied in positive × direction at nodes 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, and 71; (2) 10 kips is applied in negative Y direction at nodes 1, 2, … , 6, 8, 10, 12, 14, 15, … , 20, 22, 24, 25, … , 73, 74, and 75; (3) Cases 1 and 2 are combined together.

The 200 structural members of this spatial truss are categorized into 29 groups described in Table 12. Discrete values of cross-sectional areas were selected from the following set:

D = [0.1, 0.347, 0.44, 0.539, 0.954, 1.081, 1.174, 1.333, 1.488, 1.764, 2.142, 2.697, 2.8, 3.131, 3.565, 3.813, 4.805, 5.952, 6.572, 7.192, 8.525, 9.3, 10.85, 13.33, 14.29, 17.17, 19.18, 23.68, 28.08, 33.7] (in^2);

Table 13 compares the results obtained by the HHS algorithm and other optimization methods. Fig. 17 compares the corresponding convergence rates. In this case, the HHS algorithm obtained the same best weight as ADS (27,190.49 lb). ESASS obtained best weight of 28,075.49 lb. The HHS algorithm obtained the best solution in 7500 structural analyses. The ADS and ESASS algorithms obtained the best solution after 5000, 11,156, and 51,360 analyses, respectively.

Table 10
Loading conditions for the 72-bar truss problem.

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

6. Conclusions

This paper proposes a new variant of the HS algorithm called the Hybrid Harmony Search (HHS) algorithm. In addition to Harmony Memory and pitch adjustment concepts, the HHS algorithm integrates two new features based on Global-best PSO search and neighbourhood search and omits the randomization used in the original HS algorithm. In

Table 11
Comparison of HHS optimization results with the literature for the 72-bar truss problem.

Variables (mm^2)	HS [25]	HPSO [26]	DHPSACO [21]	HHS
A_1–A_4	1.9	2.1	1.9	1.9
A_5–A_{12}	0.5	0.6	0.5	0.5
A_{13}–A_{16}	0.1	0.1	0.1	0.1
A_{17}–A_{18}	0.1	0.1	0.1	0.1
A_{19}–A_{22}	1.4	1.4	1.3	1.4
A_{23}–A_{30}	0.6	0.5	0.5	0.5
A_{31}–A_{34}	0.1	0.1	0.1	0.1
A_{35}–A_{36}	0.1	0.1	0.1	0.1
A_{37}–A_{40}	0.6	0.5	0.6	0.5
A_{41}–A_{48}	0.5	0.5	0.5	0.5
A_{49}–A_{52}	0.1	0.1	0.1	0.1
A_{53}–A_{54}	0.1	0.1	0.1	0.1
A_{55}–A_{58}	0.2	0.2	0.2	0.2
A_{59}–A_{66}	0.5	0.5	0.6	0.6
A_{67}–A_{70}	0.4	0.3	0.4	0.4
A_{71}–A_{72}	0.6	0.7	0.6	0.6
Best (lb)	387.94	388.94	385.54	385.54
Average (lb)	N/A	N/A	N/A	386.040
Stdev (lb)	N/A	N/A	N/A	1.155
No. of analyses	16,044	50,000	5330	5000
Constraint violations	None	None	None	None

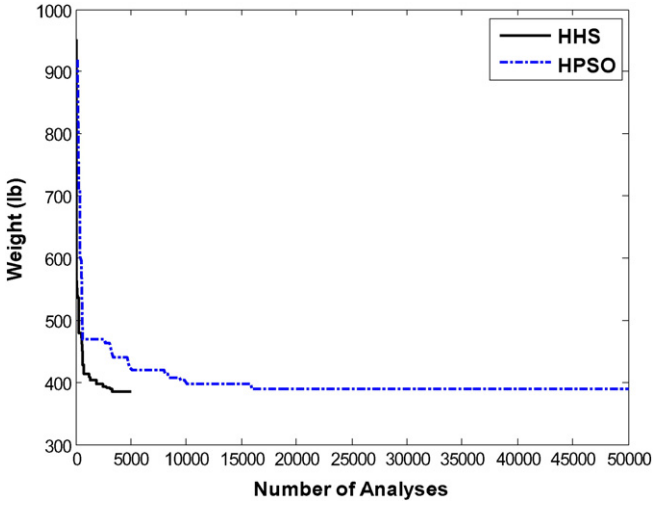

Fig. 15. Convergence curves for the spatial 72-bar truss structure.

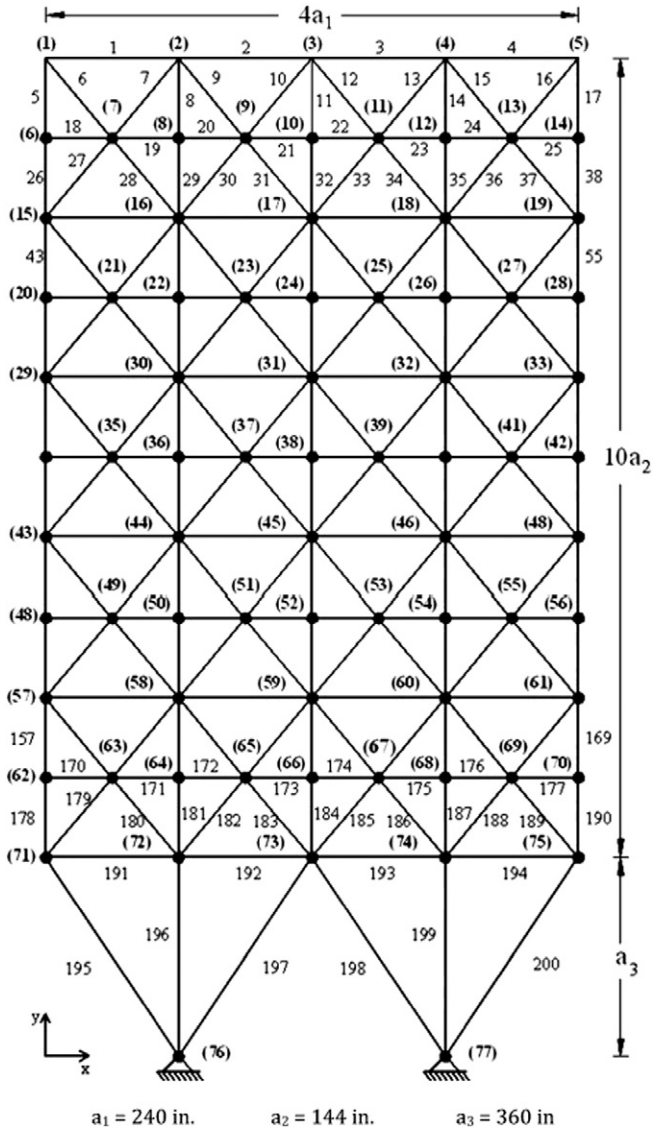

Fig. 16. Schematic of the planar 200-bar truss structure.

Table 12
Design variables in planar 200-bar truss problem.

Groups/Variables	No. of truss members
A_1	1, 2, 3, 4
A_2	5, 8, 11, 14, 17
A_3	19, 20, 21, 22, 23, 24
A_4	18, 25, 56, 63, 94, 101, 132, 139, 170, 177
A_5	26, 29, 32, 35, 38
A_6	6, 7, 9, 10, 12, 13, 15, 16, 27, 28, 30, 31, 33, 34, 36, 37
A_7	39, 40, 41, 42
A_8	43, 46, 49, 52, 55
A_9	57, 58, 59, 60, 61, 62
A_{10}	64, 67, 70, 73, 76
A_{11}	44, 45, 47, 48, 50, 51, 53, 54, 65, 66, 68, 69, 71, 72, 74, 75
A_{12}	77, 78, 79, 80
A_{13}	81, 84, 87, 90, 93
A_{14}	95, 96, 97, 98, 99, 100
A_{15}	102, 105, 108, 111, 114
A_{16}	82, 83, 85, 86, 88, 89, 91 92, 103, 104, 106, 107, 109, 110, 112, 113
A_{17}	115, 116, 117, 118
A_{18}	119, 122, 125, 128, 131
A_{19}	133, 134, 135, 136, 137, 138
A_{20}	140, 143, 146, 149, 152
A_{21}	120, 121, 123, 124, 129, 127, 129, 130, 141, 142, 144, 145, 147, 148, 150, 151
A_{22}	153, 154, 155, 156
A_{23}	157, 160, 163, 166, 169
A_{24}	171, 172, 173, 174, 175, 176
A_{25}	178, 181, 184, 187, 190
A_{26}	158, 159, 161, 162, 164, 165, 167, 168, 179, 180, 182, 183, 185, 186, 188, 189
A_{27}	191, 192, 193, 194
A_{28}	195, 197, 198, 200
A_{29}	196, 199

Table 13
Comparison of HHS optimization results with the literature for the 200-bar truss problem.

Variables (in^2)	ESASS [1]	ADS [17]	HHS
A_1	0.1	0.1	0.1
A_2	0.954	0.954	0.954
A_3	0.1	0.347	0.1
A_4	0.1	0.1	0.1
A_5	2.142	2.142	2.142
A_6	0.347	0.347	0.347
A_7	0.1	0.1	0.1
A_8	3.131	3.131	3.131
A_9	0.1	0.1	0.1
A_{10}	4.805	4.805	4.805
A_{11}	0.347	0.44	0.44
A_{12}	0.1	0.1	0.347
A_{13}	5.952	5.952	5.952
A_{14}	0.1	0.1	0.347
A_{15}	6.572	6.572	6.572
A_{16}	0.44	0.539	0.954
A_{17}	0.539	0.1	0.347
A_{18}	7.192	8.525	8.525
A_{19}	0.44	0.539	0.1
A_{20}	8.525	9.3	9.3
A_{21}	0.954	0.954	1.081
A_{22}	1.174	0.1	0.347
A_{23}	10.85	10.85	13.33
A_{24}	0.44	0.954	0.954
A_{25}	10.85	13.33	13.33
A_{26}	1.764	1.333	1.764
A_{27}	8.525	7.192	3.813
A_{28}	13.33	10.85	8.525
A_{29}	13.33	14.29	17.17
Best (lb)	28,075.49	27,190.49	27,163.59
Average (lb)	N/A	N/A	28,159.59
Stdev (lb)	N/A	N/A	1149.91
No. of analyses	11,156	5000	5000
Constraint violations	None	None	None

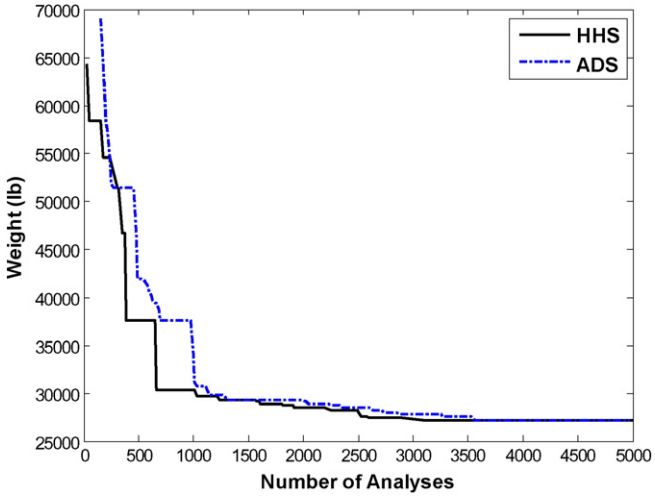

Fig. 17. Convergence curves for the spatial 200-bar truss structure.

principle, the HHS algorithm aims to optimize the balance between exploration and exploitation. Therefore, the HHS algorithm uses memory consideration and pitch adjustment to focus on global search in the early stage and Global-best PSO search and neighbourhood search to focus on local search in the later stage.

This paper tested the HHS algorithm extensively on six discrete truss structure optimization problems under multiple loading conditions. We compared the numerical results for various trusses obtained by the HHS algorithm with results obtained by other published approaches in order to verify HHS algorithm effectiveness, efficiency, and robustness.

The HHS algorithm generally performed better than the optimization methods used for comparison in terms of both optimum solution and convergence capability. In nearly all design examples, the HHS algorithm obtained a result that was comparable or better than literature and required much less structural analyses. Because HHS algorithm not only strikes an optimal balance between exploration and exploitation but also converges to the optimum solution significantly faster than other tested methods, this algorithm is an ideal method for dealing with complex engineering problems.

The HHS algorithm infused the advantages of several current techniques to improve overall effectiveness and overcome weaknesses inherent in each individual technique. Obtained results confirm the effectiveness of this hybrid algorithm in minimizing shortcoming in local searching ability while still maintaining the good capability in finding the high performance regions of searching space. Further research is required to clarify HHS efficiencies in dealing with large-scale optimization problems.

References

[1] S.K. Azad, O. Hasançebi, An elitist self-adaptive step-size search for structural design optimization, Appl. Soft Comput. 19 (2014) 226–235.
[2] C.V. Camp, Design of space trusses using big bang–big crunch optimization, J. Struct. Eng. 133 (7) (2007) 999–1008.
[3] M.-Y. Cheng, L.-C. Lien, A hybrid AI-based particle bee algorithm for facility layout optimization, Eng. Comput. 28 (1) (2012) 57–69.
[4] M.-Y. Cheng, D. Prayogo, Symbiotic organisms search: a new metaheuristic optimization algorithm, Comput. Struct. 139 (2014) 98–112.
[5] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, B.K. Panigrahi, Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization, Syst, Man Cybern. B Cybern. IEEE Trans. 41 (1) (2011) 89–106.
[6] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Eng. 186 (2–4) (2000) 311–338.
[7] T. Dede, S. Bekiroğlu, Y. Ayvaz, Weight minimization of trusses with genetic algorithm, Appl. Soft Comput. 11 (2) (2011) 2565–2575.
[8] S.O. Degertekin, Improved harmony search algorithms for sizing optimization of truss structures, Comput. Struct. 92–93 (2012) 229–241.
[9] S.O. Degertekin, Optimum design of steel frames using harmony search algorithm, Struct. Multidiscip. Optim. 36 (4) (2008) 393–401.
[10] F. Erdal, The comparative analysis of optimal designed web expanded beams via improved harmony search method, Struct. Eng. Mech. 54 (4) (2015) 665–691.
[11] F. Erdal, E. Doğan, M.P. Saka, Optimum design of cellular beams using harmony search and particle swarm optimizers, J. Constr. Steel Res. 67 (2) (2011) 237–247.
[12] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, Y. Alizadeh, Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems, Comput. Methods Appl. Mech. Eng. 197 (33–40) (2008) 3080–3091.
[13] A. Gandomi, X.-S. Yang, A. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Eng. Comput. 29 (1) (2013) 17–35.
[14] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, SIMULATION 76 (2) (2001) 60–68.
[15] K. Guney, M. Onay, Optimal synthesis of linear antenna arrays using a harmony search algorithm, Expert Syst. Appl. 38 (12) (2011) 15455–15462.
[16] W. Hare, J. Nutini, S. Tesfamariam, A survey of non-gradient optimization methods in structural engineering, Adv. Eng. Softw. 59 (2013) 19–28.
[17] O. Hasançebi, S.K. Azad, Adaptive dimensional search: a new metaheuristic algorithm for discrete truss sizing optimization, Comput. Struct. 154 (2015) 1–16.
[18] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.
[19] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization, 2005.
[20] A. Kaveh, S. Talatahari, A particle swarm ant colony optimization for truss structures with discrete variables, J. Constr. Steel Res. 65 (8–9) (2009) 1558–1568.
[21] A. Kaveh, S. Talatahari, Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures, Comput. Struct. 87 (5–6) (2009) 267–283.
[22] J. Kennedy, R. Eberhart, Particle swarm optimization, Proceedings of the IEEE International Conference on Neural Networks, vol. 4, Perth, Australia 1995, pp. 1942–1948.
[23] L. Lamberti, An efficient simulated annealing algorithm for design optimization of truss structures, Comput. Struct. 86 (19–20) (2008) 1936–1953.
[24] K.S. Lee, Z.W. Geem, A new structural optimization method based on the harmony search algorithm, Comput. Struct. 82 (9–10) (2004) 781–798.
[25] K.S. Lee, Z.W. Geem, S.-h. Lee, K.-w. Bae, The harmony search heuristic algorithm for discrete structural optimization, Eng. Optim. 37 (7) (2005) 663–684.
[26] L.J. Li, Z.B. Huang, F. Liu, A heuristic particle swarm optimization method for truss structures with discrete variables, Comput. Struct. 87 (7–8) (2009) 435–443.
[27] L.-C. Lien, M.-Y. Cheng, A hybrid swarm intelligence based particle-bee algorithm for construction site layout optimization, Expert Syst. Appl. 39 (10) (2012) 9642–9650.
[28] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Appl. Math. Comput. 188 (2) (2007) 1567–1579.
[29] D.T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, M. Zaidi, The bees algorithm, a novel tool for complex optimisation problems, Proceedings of the 2nd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2006), Elsevier, Oxford 2006, pp. 454–459.
[30] S. Rajeev, C. Krishnamoorthy, Discrete optimization of structures using genetic algorithms, J. Struct. Eng. 118 (5) (1992) 1233–1250.
[31] U.L.F.T. Ringertz, On methods for discrete structural optimization, Eng. Optim. 13 (1) (1988) 47–64.
[32] M.P. Saka, F. Erdal, Harmony search based algorithm for the optimum design of grillage systems to LRFD-AISC, Struct. Multidiscip. Optim. 38 (1) (2009) 25–41.
[33] M. Sonmez, Artificial bee colony algorithm for optimization of truss structures, Appl. Soft Comput. 11 (2) (2011) 2406–2418.
[34] R. Storn, K. Price, Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
[35] W.H. Tong, G.R. Liu, An optimization procedure for truss structures with discrete design variables and dynamic constraints, Comput. Struct. 79 (2) (2001) 155–162.
[36] S.-J. Wu, P.-T. Chow, Steady-state genetic algorithms for discrete optimization of trusses, Comput. Struct. 56 (6) (1995) 979–991.
[37] B.-W. Xiang, R.-Q. Chen, T. Zhang, Optimization of trusses using simulated annealing for discrete variables, image analysis and signal processing, 2009, IASP 2009. International Conference on 2009, pp. 410–414.
[38] P. Yadav, R. Kumar, S.K. Panda, C.S. Chang, An intelligent tuned harmony search algorithm for optimisation, Inf. Sci. 196 (0) (2012) 47–72.
[39] X.-S. Yang, Firefly algorithms for multimodal optimization, Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications, Springer-Verlag, Sapporo, Japan 2009, pp. 169–178.
[40] X.-S. Yang, Harmony search as a metaheuristic algorithm, in: Z. Geem (Ed.), Music-inspired Harmony Search Algorithm, vol. 191, Springer, Berlin Heidelberg 2009, pp. 1–14.
[41] X.-S. Yang, S. Deb, Cuckoo search via Levy flights, nature & biologically inspired computing, 2009, NaBIC 2009. World Congress on 2009, pp. 210–214.
[42] Y.-N. Zhang, J.-P. Liu, B. Liu, C.-Y. Zhu, Y. Li, Application of improved hybrid genetic algorithm to optimized design of architecture structures, J. South China Univ. Technol. 33 (3) (2003) 69–72.