Available online at www.sciencedirect.com
SCIENCE DIRECT®
Comput. Methods Appl. Mech. Engrg. 194 (2005) 3902–3933
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice

Kang Seok Lee [a,*,1], Zong Woo Geem [b]

[a] *Materials and Construction Research Division, Building and Fire Research Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8611, USA*
[b] *Department of Civil and Environmental Engineering, University of Maryland, College Park, MD 20742, USA*

Received 4 August 2003; received in revised form 31 August 2004; accepted 30 September 2004

Abstract

Most engineering optimization algorithms are based on numerical linear and nonlinear programming methods that require substantial gradient information and usually seek to improve the solution in the neighborhood of a starting point. These algorithms, however, reveal a limited approach to complicated real-world optimization problems. If there is more than one local optimum in the problem, the result may depend on the selection of an initial point, and the obtained optimal solution may not necessarily be the global optimum. This paper describes a new harmony search (HS) meta-heuristic algorithm-based approach for engineering optimization problems with continuous design variables. This recently developed HS algorithm is conceptualized using the musical process of searching for a perfect state of harmony. It uses a stochastic random search instead of a gradient search so that derivative information is unnecessary. Various engineering optimization problems, including mathematical function minimization and structural engineering optimization problems, are presented to demonstrate the effectiveness and robustness of the HS algorithm. The results indicate that the proposed approach is a powerful search and optimization technique that may yield better solutions to engineering problems than those obtained using current algorithms.

Keywords: Harmony search; Heuristic algorithm; Continuous design variables; Mathematical function minimization; Structural engineering optimization

* Corresponding author. Tel.: +1 301 975 2881; fax: +1 301 869 6275.
E-mail addresses: ksleenist@hotmail.com (K.S. Lee), zwgeem@yahoo.com (Z.W. Geem).
[1] Present address: Advanced Structure Research Station, Hanyang University, Seoul 133-791, Korea.

0045-7825/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2004.09.007

1. Introduction

Over the last four decades, a large number of algorithms have been developed to solve various engineering optimization problems. Most of these algorithms are based on numerical linear and nonlinear programming methods that require substantial gradient information and usually seek to improve the solution in the neighborhood of a starting point. These numerical optimization algorithms provide a useful strategy to obtain the global optimum in simple and ideal models. Many real-world engineering optimization problems, however, are very complex in nature and quite difficult to solve using these algorithms. If there is more than one local optimum in the problem, the result may depend on the selection of an initial point, and the obtained optimal solution may not necessarily be the global optimum. Furthermore, the gradient search may become difficult and unstable when the objective function and constraints have multiple or sharp peaks.

The computational drawbacks of existing numerical methods have forced researchers to rely on meta-heuristic algorithms based on simulations to solve engineering optimization problems. The common factor in meta-heuristic algorithms is that they combine rules and randomness to imitate natural phenomena. These phenomena include the biological evolutionary process (e.g., the evolutionary algorithm proposed by Fogel et al. [1], De Jong [2], and Koza [3] and the genetic algorithm (GA) proposed by Holland [4] and Goldberg [5]), animal behavior (e.g., tabu search proposed by Glover [6]), and the physical annealing process (e.g., simulated annealing proposed by Kirkpatrick et al. [7]). In the last decade, these meta-heuristic algorithms, especially GA-based methods have been studied by many researchers to solve various engineering optimization problems. The GA was originally proposed by Holland [4] and further developed by Goldberg [5] and by others. It is a global search algorithm that is based on concepts from natural genetics and the Darwinian survival-of-the-fittest code. Meta-heuristic algorithm-based engineering optimization methods, including GA-based methods, have occasionally overcome several deficiencies of conventional numerical methods. To solve difficult and complicated real-world optimization problems, however, new heuristic and more powerful algorithms based on analogies with natural or artificial phenomena must be explored.

Recently, Geem et al. [8] developed a new harmony search (HS) meta-heuristic algorithm that was conceptualized using the musical process of searching for a perfect state of harmony. The harmony in music is analogous to the optimization solution vector, and the musician's improvisations are analogous to local and global search schemes in optimization techniques. The HS algorithm does not require initial values for the decision variables. Furthermore, instead of a gradient search, the HS algorithm uses a stochastic random search that is based on the harmony memory considering rate and the pitch adjusting rate (defined in harmony search meta-heuristic algorithm section) so that derivative information is unnecessary. Compared to earlier meta-heuristic optimization algorithms, the HS algorithm imposes fewer mathematical requirements and can be easily adopted for various types of engineering optimization problems.

In this study, we describe a brief overview of existing meta-heuristic algorithms and a new HS meta-heuristic algorithm-based approach for engineering optimization problems with continuous design variables. Various standard benchmark engineering optimization examples including function minimization problems and structural optimization problems from the literature are also presented to demonstrate the effectiveness and robustness of the HS meta-heuristic algorithm method.

2. A brief overview of existing meta-heuristic algorithms

Since the 1970s, many meta-heuristic algorithms that combine rules and randomness imitating natural phenomena have been devised to overcome the computational drawbacks of existing numerical algorithms (i.e., complex derivatives, sensitivity to initial values, and the large amount of enumeration memory

required) when solving difficult and complex engineering optimization problems. These algorithms include simulated annealing, tabu search, and evolutionary computation methods.

Simulated annealing is a generalization of the Monte Carlo method for examining the equations of state and frozen states of n-body systems [9]. The concept is based on the manner in which liquids freeze or metals recrystallize in an annealing process. In an annealing process, a melt, initially at a high temperature and disordered, is cooled slowly so that the system at any time is approximately in thermodynamic equilibrium. As cooling proceeds, the system becomes more ordered and approaches a "frozen" ground state at $T = 0$. Hence, the process can be thought of as an adiabatic approach to the lowest energy state. If the initial temperature of the system is too low or cooling is performed too quickly, the system may become quenched, forming defects or freezing in meta-stable states (i.e., states that are trapped in a local minimum energy state). The connection between this algorithm and mathematical minimization was first noted by Pincus [10], but Kirkpatrick et al. [7] formed the basis of an optimization technique for combinatorial (and other) problems.

A tabu search, which was originally suggested by Glover [6], is basically a gradient-descent search with memory. The memory retains a number of previously visited states along with a number of states that might be considered undesirable. This information is stored in a tabu list. The definition of a state, the area around it, and the length of the tabu list are critical design parameters. In addition to these tabu parameters, two extra parameters are often used: aspiration and diversification. Aspiration is used when all the neighboring states of the current state are also included in the tabu list. In that case, the tabu obstacle is overridden by selecting a new state. Diversification adds randomness to this otherwise deterministic search. If the tabu search is not converging, the search is reset randomly.

Evolutionary computational methods, such as GA, evolution strategies, evolutionary programming, genetic programming, particle swarm, and probabilistic model building genetic algorithm (PMBGA) or estimation of distribution algorithm (EDA) are based on the principle of evolution (survival-of-the-fittest) and imitate some natural phenomena (genetic inheritance).

The GA is a search algorithm based on natural selection and the mechanisms of population genetics. The theory was proposed by Holland [4] and further developed by Goldberg [5] and others. A simple GA is comprised of three operators: reproduction, crossover, and mutation. Reproduction is the process of survival-of-the-fittest selection. Crossover is the partial swapping between two parent strings to produce two offspring strings. Mutation is the occasional random inversion of bit values that generates non-recursive offspring. The main characteristic of the GA is the simultaneous evaluation of many solutions. This differs from numerical algorithms or other heuristic algorithms, such as simulated annealing or the tabu search, which evaluate only one solution at each iteration. This feature can be an advantage, enabling a wide search and potentially avoiding convergence to a non-global optimum.

Evolution strategies, developed to solve parameter optimization problems [11], employ real-coded variables and, in its original form, the algorithm relied on mutation as the search operator with a population size of one. The algorithm has since evolved to share many features with the GA. The main similarity between these two types of algorithms is that they both maintain populations of potential solutions and use a selection mechanism to choose the best individuals from the population. Evolution strategies operate directly on floating point vectors, while the classical GA operates on binary strings. The GA relies mainly on recombination to explore the search space, while evolution strategies use mutation as the dominant operator. Evolution strategies are an abstraction of evolution at the individual behavior level, stressing the behavioral link between an individual and its offspring, while the GA maintains the genetic link.

Evolutionary programming, which was originally developed by Fogel et al. [1], describes the evolution of finite state machines to solve prediction tasks. The state transition tables in these machines are modified by uniform random mutations on the corresponding alphabet. The algorithm utilizes selection and mutation as the main operators, and the selection process is a stochastic tournament.

Genetic programming, which is an extension of the GA, was developed by Koza [3]. He suggested that the desired program should evolve during the evolution process. Genetic programming is very similar to the GA, but differs mainly in the representation of the solution. Genetic programming uses LISP or Scheme computer languages to create computer programs that represent the solution, while the GA create strings of numbers that represent the solution.

Particle swarm is an evolutionary computational method developed by Kennedy and Eberhart [12] and inspired by the social behavior of birds flocking or fish schooling. Similar to the GA, the particle swarm algorithm is a population-based optimization method. The system is initialized with a population of random solutions and searches for the optima by updating generations. Unlike the GA, however, the particle swarm algorithm has no evolution operators, such as crossover and mutation.

On the other hand, the GA tries to find better solution by selection and recombination of promising solution based on algorithm operators such as crossover and mutation, as previously stated. It works well in wide verities of problem domains. However, sometimes simple selection and crossover operators are not effective enough to get optimum solution as they might not effectively preserve important patterns, known as building blocks or partial solutions, in chromosomes. It often happens in the problem domains where the building blocks are loosely distributed. The search techniques to preserve building blocks lead to a new class of algorithm called probabilistic model building genetic algorithm (PMBGA) [13], also known as estimation of distribution algorithm (EDA) [14]. The PMBGA is a new area of evolutionary computational methods. The general procedure of PMBGA is similar to that of the GA, but the classical recombination operators, crossover and mutation, are replaced by probability estimation followed by probability sampling. The initial population of PMBGA is first generated randomly. In each iteration, promising solutions are selected from the current population of candidate solutions and the true probability distribution of these selected solutions is estimated. New candidate solutions are then generated by sampling the estimated probability distribution. The new solutions are then incorporated into the original population, replacing some of the old ones or all of them. The process is repeated until the termination criteria are met.

The PMBGA in binary string presentation can be classified into three classes depending on the complexity of models they use; univariate, bivariate, and multivariate models [13]. In univariate models, variables are treated independently. Algorithms belonging to this category work very well for linear problems where they achieve linear or sub-quadratic performance, depending on the type of a problem, but they fail on problems with strong interactions among variables [15]. These algorithms include population-based incremental learning (PBIL), univariate marginal distribution algorithm (UMDA), and compact genetic algorithm (cGA).

The PBIL was introduced by Baluja [16]. The solutions are represented by binary strings of fixed length, but the population of solutions is replaced by the so-called probability vector $(p_0, p_1, \ldots, p_{n-1})$, where p_i refers to the probability of obtaining a value of 1 in the ith gene. This vector is initially set to assign each value at the same position with the same probability $p_i = 0.5$. At each generation a number of solutions is generated using this vector. Then the few best solutions are selected and the probability vector is shifted towards the selected solutions by using Hebbian learning rule:

$$p_i = (1 - \alpha)p_i + \alpha \frac{1}{N} m \quad (X_i = 1), \tag{1}$$

where N is the size of selected population, $\alpha \in (0, 1)$ is a parameter of the algorithm and $m(X_i = 1)$ denotes number of individuals in the selected part of population having $X_i = 1$.

The UMDA, one of the early work in the field of PMBGA, was first introduced by Muehlenbein [17]. In contrast to PBIL, the population of solutions is kept and processed. In each iteration, the frequency functions on each position for the selected set of promising solutions are computed and these are then used to generate new solutions. The new solutions replace the old ones and the process is repeated until the termination criteria are met. From the implementation point of view the UMDA matches the pseudo-code of

typical EDA algorithm, except that the dependence graph of constructed probabilistic model contains no edges [15]. The probabilistic model is as simple as possible:

$$p(X) = \prod_{i=0}^{n-1} p(X_i), \tag{2}$$

where each univariate marginal distribution $p(X_i)$ is estimated from marginal frequencies:

$$p(X_i = 1) = \frac{m(X_i = 1)}{N}, \tag{3}$$

where N is the size of selected population and $m(X_i = 1)$ denotes number of individuals in the selected part of population having $X_i = 1$.

In the cGA proposed by Harik et al. [18], the population is replaced by a single probability vector like in PBIL. However, unlike the PBIL, it modifies the probability vector so that there is direct correspondence between the population that is represented by the probability vector and the probability vector itself. Two individuals are generated from this vector of probabilities, the competition between them is carried out and the winner is used for vector update. But instead of shifting the vector proportionally to the distance from either 0 or 1, each component of the vectors is updated by shifting its value by the contribution of a single individual to the total frequency assuming a particular population size. By using this update rule, theory of simple GA can directly be used in order to estimate the parameters and behavior of the cGA.

Bivariate models consider pair wise dependencies among variables in chromosome, i.e., consider the building blocks of order two. Similarly the probability model becomes more complex than one of univariate model and takes a form of probabilistic network between variables. This class of algorithm performs better in problems with pair wise interaction among variables, but fails in the problems with multiple variable interactions. Mutual information maximization for input clustering (MIMIC) (De Bonet et al. [19]), combining optimizers with mutual information trees (COMIT) (Baluja and Davies [20]), and bivariate marginal distribution algorithm (BMDA) (Pelikan and Muhlenbein [21]) use a bivariate model of probability distribution. Graphical representation of probabilistic networks used by these algorithms is shown in Fig. 1.

Any algorithms considering interdependency between variable of order more than two can be placed in multivariate models. The probability network representing interdependency of variables obviously becomes more complex and the computation time to construct such network hugely increases making it almost impossible to search through all possible models. Due to its simplicity, most of algorithms in multivariate models uses greedy heuristic to search a good model, however, greedy heuristics does not always guarantees accuracy. Some other complex search algorithms have also been successfully used for this purpose and lots of current researches in PMBGA are focused on finding good heuristic. Extended compact genetic algorithm (ECGA) (Harik [22]), factorized distribution algorithm (FDA) (Muhlenbein and Mahnig [23]), Bayesian optimization algorithm (BOA) (Pelikan et al. [24]), learning factorized distribution algorithm (LFDA) (Muhlenbein and Mahnig [25]), and estimation of Bayesian network algorithm (EBNA)

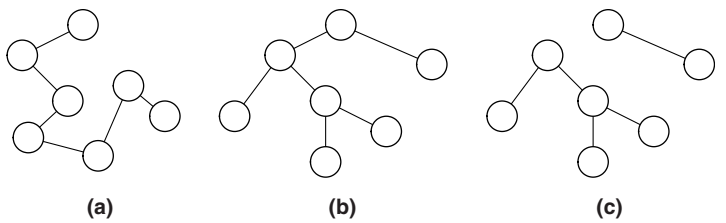


Fig. 1. Graphical representation of bivariate models. (a) Chain model (MMIC), (b) tree model (COMIT) and (c) forest model (BMDA).

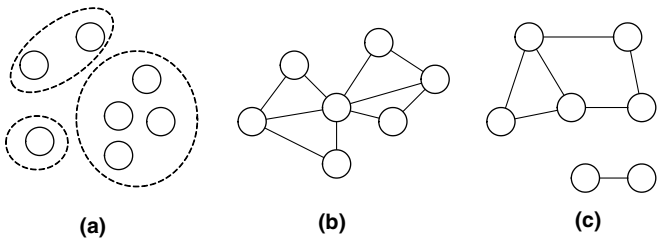


Fig. 2. Graphical representation of multivariate models. (a) Marginal product model (ECGA), (b) triangular model (FDA) and (c) Bayesian network (BOA, EBOA).

(Etxeberria and Larranaga [26]) use multivariate models of probability distribution. Fig. 2 shows a graphical representation of different probabilistic networks used by these algorithms.

3. Harmony search meta-heuristic algorithm

Current meta-heuristic algorithms imitate natural phenomena, i.e., physical annealing in simulated annealing, human memory in tabu search, and evolution in evolutionary algorithms. A new HS meta-heuristic algorithm was conceptualized using the musical process of searching for a perfect state of harmony. Musical performances seek to find pleasing harmony (a perfect state) as determined by an aesthetic standard, just as the optimization process seeks to find a global solution (a perfect state) as determined by an objective function. The pitch of each musical instrument determines the aesthetic quality, just as the objective function value is determined by the set of values assigned to each decision variable. The new HS meta-heuristic algorithm was derived based on natural musical performance processes that occur when a musician searches for a better state of harmony, such as during jazz improvisation [8].

Fig. 3 shows the details of the analogy between music improvisation and engineering optimization. In music improvisation, each player sounds any pitch within the possible range, together making one harmony vector. If all the pitches make a good harmony, that experience is stored in each player's memory, and the possibility to make a good harmony is increased next time. Similarly in engineering optimization, each decision variable initially chooses any value within the possible range, together making one solution vector. If all the values of decision variables make a good solution, that experience is stored in each variable's memory, and the possibility to make a good solution is also increased next time.

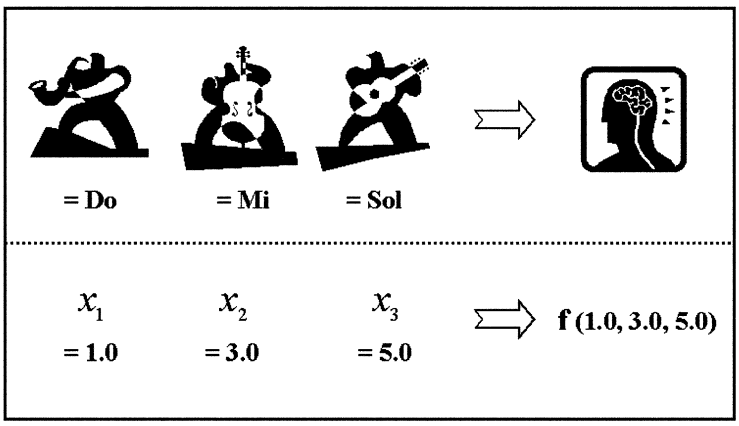


Fig. 3. Analogy between music improvisation and engineering optimization.

Although the estimation of a harmony is aesthetic and subjective, on the other hand, there are several theorists who have provided the standard of harmony estimation: Greek philosopher and mathematician Pythagoras (582–497BC) worked out the frequency ratios (or string length ratios with equal tension) and found that they had a particular mathematical relationship, after researching what notes sounded pleasant together. The octave was found to be a 1:2 ratio and what we today call a fifth to be a 2:3 ratio; French composer Leonin (1135–1201) is the first known significant composer of polyphonic "organum" which involved a simple doubling of the chant at an interval of a fifth or fourth above or below; and French composer Jean-Philippe Rameau (1683–1764) established the classical harmony theories in the book "Treatise on Harmony", which still form the basis of the modern study of tonal harmony.

In engineering optimization, the estimation of a solution is carried out in putting values of decision variables to objective function or fitness function, evaluating the function value with respect to several aspects such as cost, efficiency, and/or error.

Fig. 4 shows the structure of the harmony memory (HM) that is the core part of the HS. Consider a jazz trio composed of saxophone, double bass, and guitar. There exist certain amount of preferable pitches in each musician's memory: saxophonist, {Do, Mi, Sol}; double bassist, {Si, Sol, Re}; and guitarist, {La, Fa, Do}. If saxophonist randomly plays {Sol} out of its memory {Do, Mi, Sol}, double bassist {Si} out of {Si, Sol, Re}, and guitarist {Do} out of {La, Fa, Do}, that harmony (Sol, Si, Do) makes another harmony (musically C-7 chord). And if this new harmony is better than existing worst harmony in the HM, the new harmony is included in the HM and the worst harmony is excluded from the HM. This procedure is repeated until fantastic harmony is found.

In real optimization, each musician can be replaced with each decision variable, and its preferred sound pitches can be replaced with each variable's preferred values. If each decision variable represents pipe diameter of an arc between two nodes, it has certain number of preferred diameters. And if first variable chooses {100 mm} out of {100 mm, 300 mm, 500 mm}, second {500 mm} out of {700 mm, 500 mm, 200 mm}, and third {400 mm} out of {600 mm, 400 mm, 100 mm}, those values (100 mm, 500 mm, 400 mm) make another solution vector. And if this new vector is better than existing worst vector in the HM, the new vector is included in the HM and the worst vector is excluded from the HM. This procedure is repeated until certain termination criterion is satisfied.

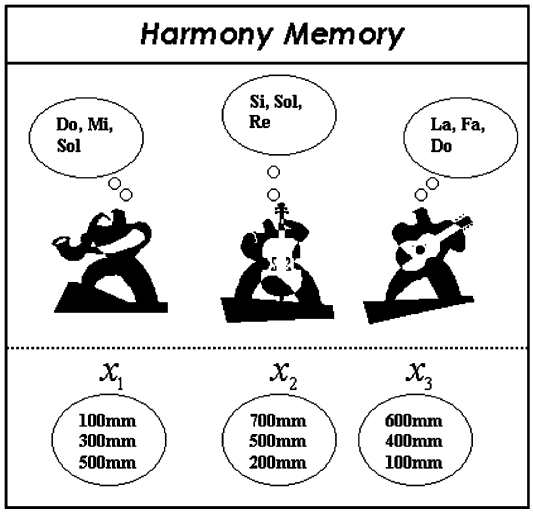


Fig. 4. Structure of harmony memory.

When a musician improvises one pitch, usually he (or she) follows any one of three rules: (1) playing any one pitch from his (or her) memory, (2) playing an adjacent pitch of one pitch from his (or her) memory, and (3) playing totally random pitch from the possible sound range. Similarly, when each decision variable chooses one value in the HS algorithm, it follows any one of three rules: (1) choosing any one value from the HS memory (defined as memory considerations), (2) choosing an adjacent value of one value from the HS memory (defined as pitch adjustments), and (3) choosing totally random value from the possible value range (defined as randomization). The three rules in HS algorithm are effectively directed using two parameters, i.e., harmony memory considering rate (HMCR) and pitch adjusting rate (PAR), as stated later.

Fig. 5 represents the optimization procedure of the HS meta-heuristic algorithm, which consists of Steps 1–5, as follows:

Step 1. Initialize the optimization problem and algorithm parameters.
Step 2. Initialize the harmony memory (HM).
Step 3. Improvise a new harmony from the HM.
Step 4. Update the HM.
Step 5. Repeat Steps 3 and 4 until the termination criterion is satisfied.

(1) *Step 1. Initialize the optimization problem and algorithm parameters*. First, the optimization problem is specified as follows:

$$\text{Minimize } f(x) \text{ s.t. } x_i \in X_i, \quad i = 1, 2, \ldots, N, \tag{4}$$

where $f(x)$ is the objective function; x is the set of each design variable (x_i); X_i is the set of the possible range of values for each design variable (continuous design variables), that is, ${}_Lx_i \leqslant X_i \leqslant {}_Ux_i$; and N is the number of design variables. The HS algorithm parameters that are required to solve the optimization problem (i.e.,

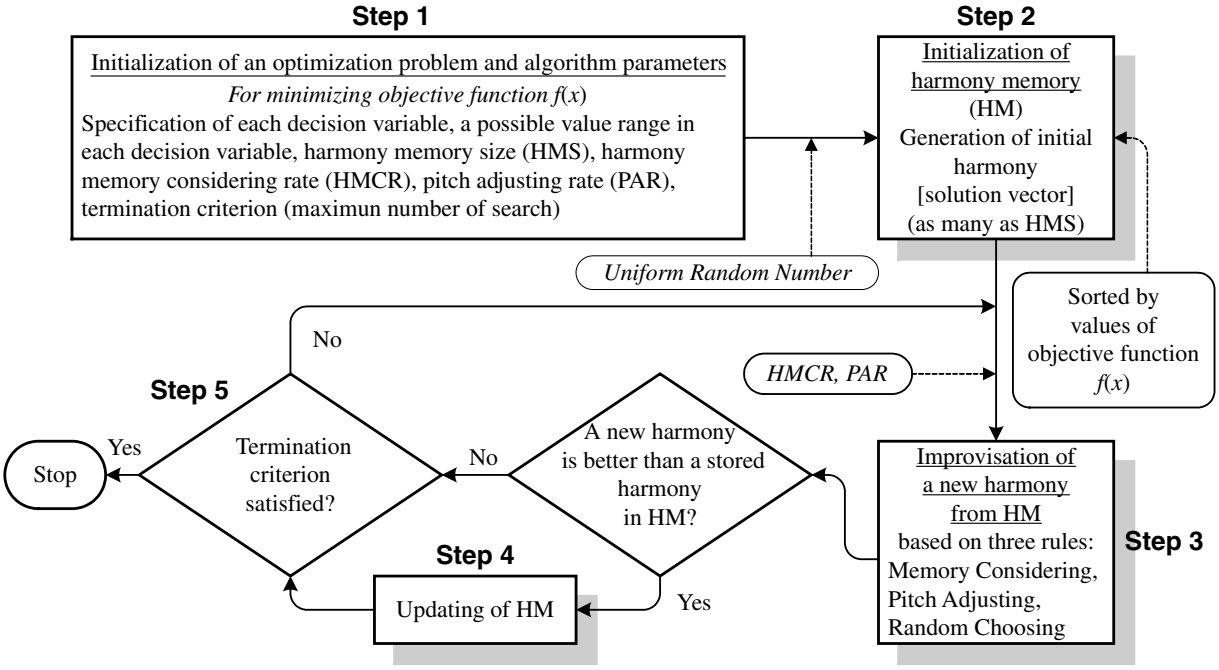


Fig. 5. Optimization procedure of the harmony search algorithm.

Eq. (4)) are also specified in this step: the harmony memory size (number of solution vectors in harmony memory, HMS), harmony memory considering rate (HMCR), pitch adjusting rate (PAR), and termination criterion (maximum number of searches). Here, HMCR and PAR are parameters that are used to improve the solution vector. Both are defined in Step 3.

(2) *Step 2. Initialize the harmony memory (HM).* In Step 2, the "harmony memory" (HM) matrix, shown in Eq. (5), is filled with randomly generated solution vectors and sorted by the values of the objective function, $f(x)$.

$$\mathrm{HM} = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^{\mathrm{HMS}} \end{bmatrix}. \tag{5}$$

(3) *Step 3. Improvise a new harmony from the HM.* In Step 3, a new harmony vector, $x' = (x'_1, x'_2, \ldots, x'_N)$ is generated from the HM based on memory considerations, pitch adjustments, and randomization, as shown in Fig. 6. For instance, the value of the first design variable (x'_1) for the new vector can be chosen from any value in the specified HM range $(x'_1 – x_1^{\mathrm{HMS}})$. Values of the other design variables (x'_1) can be chosen in the same manner. Here, it is possible to choose the new value using the HMCR parameter, which varies between 0 and 1 as follows:

$$x'_i \leftarrow \begin{cases} x'_i \in \{x_i^1, x_i^2, \ldots, x_i^{\mathrm{HMS}}\} & \text{with probability HMCR,} \\ x'_i \in X_i & \text{with probability (1-HMCR).} \end{cases} \tag{6}$$

The HMCR is the probability of choosing one value from the historic values stored in the HM, and (1-HMCR) is the probability of randomly choosing one feasible value not limited to those stored in the HM. For example, an HMCR of 0.95 indicates that the HS algorithm will choose the design variable value from historically stored values in the HM with a 95% probability, and from the entire feasible range with a 5% probability. An HMCR value of 1.0 is not recommended because of the possibility that the solution may be improved by values not stored in the HM. This is similar to the reason why the genetic algorithm uses a mutation rate in the selection process.

Every component of the new harmony vector, $x' = (x'_1, x'_2, \ldots, x'_N)$, is examined to determine whether it should be pitch-adjusted. This procedure uses the PAR parameter that sets the rate of adjustment for the pitch chosen from the HM as follows:

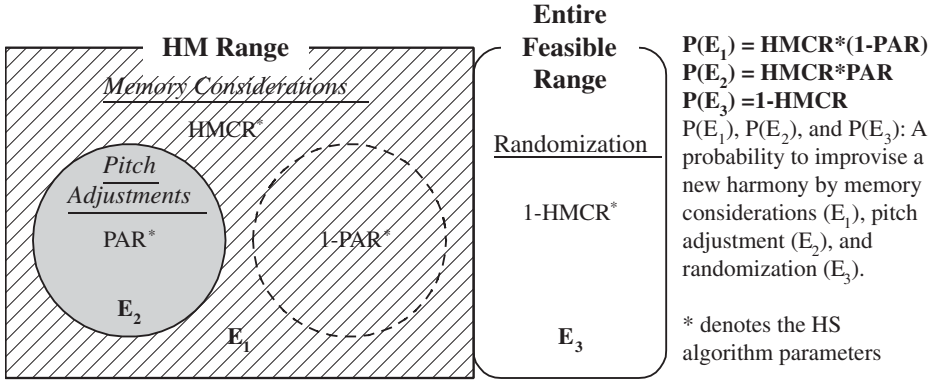


Fig. 6. New harmony improvisation concept (Step 3).

$$\text{Pitch adjusting decision for } x_i' \leftarrow \begin{cases} \text{Yes with probability PAR,} \\ \text{No with probability (1-PAR).} \end{cases} \tag{7}$$

The Pitch adjusting process is performed only after a value is chosen from the HM. The value (1-PAR) sets the rate of doing nothing. A PAR of 0.3 indicates that the algorithm will choose a neighboring value with $30\% \times \text{HMCR}$ probability. If the pitch adjustment decision for x_i' is Yes, and x_i' is assumed to be $x_i(k)$, i.e., the kth element in X_i, the pitch-adjusted value of $x_i(k)$ is:

$$x_i' \leftarrow x_i' + \alpha, \tag{8}$$

where α is the value of $bw \times u(-1,1)$, bw is an arbitrary distance bandwidth for the continuous design variable, and $u(-1,1)$ is a uniform distribution between -1 and 1. A detailed flowchart for the new harmony continuous search strategy based on the HS meta-heuristic algorithm is given in Fig. 7. The HMCR and PAR parameters introduced in the harmony search help the algorithm find globally and locally improved solutions, respectively.

(4) *Step 4. Update the HM.* In Step 4, if the new harmony vector is better than the worst harmony in the HM in terms of the objective function value, the new harmony is included in the HM and the existing worst harmony is excluded from the HM. The HM is then sorted by the objective function value.

(5) *Step 5. Repeat Steps 3 and 4 until the termination criterion is satisfied.* In Step 5, the computations are terminated when the termination criterion is satisfied. If not, Steps 3 and 4 are repeated.

To further understand the HS meta-heuristic algorithm, consider the following mathematical function minimization problem:

$$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4. \tag{9}$$

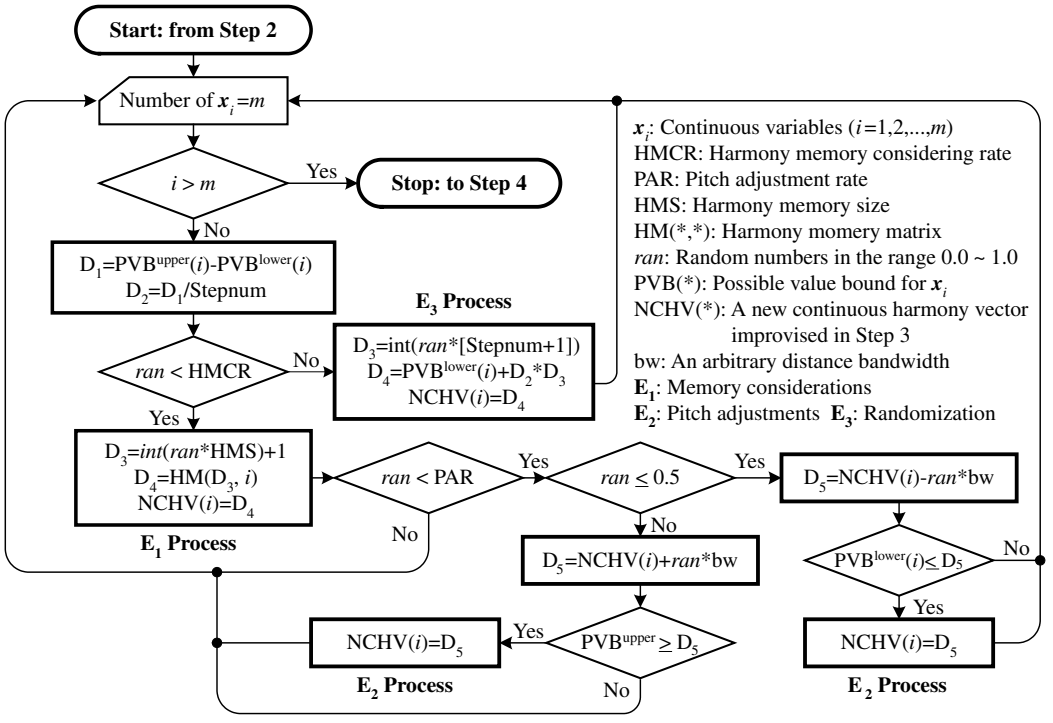


Fig. 7. A new harmony improvisation flowchart for continuous variables (Step 3).

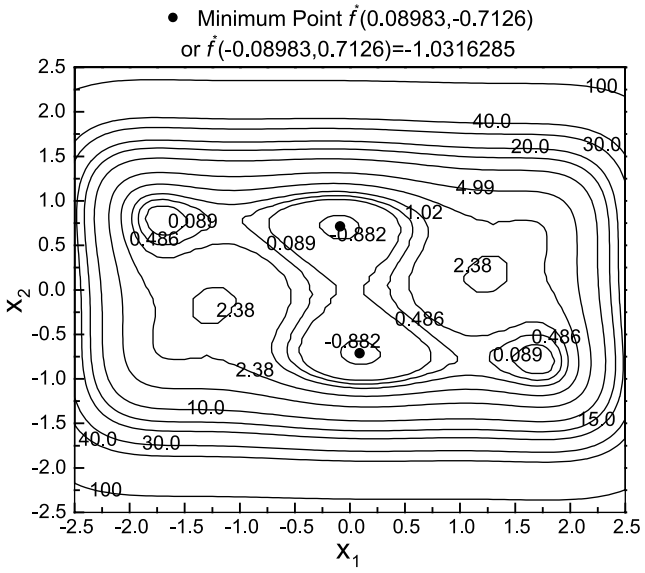


Fig. 8. Six-Hump Camelback function.

The objective function is the Six-Hump Camelback function, as shown in Fig. 8, which is one of the standard test functions in optimization problems [27]. Due to the six local optima that are present in the function, two of which are global, the result from gradient-based algorithms may depend on the selection of an initial point. Thus, the obtained optimal solution may not necessarily be the global optima that are located at either $x^* = (-0.08984, 0.71266)$ or $x^* = (0.08984, -0.71266)$, each with a corresponding function value equal to $f^*(x) = -1.0316285$. The HS meta-heuristic algorithm described in this study finds the solution vector using a different method. When applying the HS algorithm to the Six-Hump Camelback function, possible value bounds between -10.0 and 10.0 were used for the two design variables, x_1 and x_2, shown in Eq. (9). The total number of solution vectors, i.e., the HMS, was 10, and the HMCR and the PAR were 0.85 and 0.45, respectively (Step 1).

As shown in Table 1, the HM was initially structured with randomly generated solution vectors within the bounds prescribed for this example (i.e., -10.0 to 10.0); these were sorted according to the values of the objective function, Eq. (9) (Step 2). Next, a new harmony vector $x_i' = (3.183, 8.666)$ was improvised from possible values based on three rules: memory considerations with a 46.75% probability ($0.85 \times 0.55 = 0.4675$), pitch adjustments with a 38.25% probability ($0.85 \times 0.45 = 0.3825$), and randomization with a 15% probability ($1 - 0.85 = 0.15$), as described above in Step 3. As the objective function value of the new harmony (3.183, 8.666) was 22454.67, the new harmony was included in the HM and the worst harmony ($-0.95, 3.333$) was excluded from the HM, as shown in Table 1 (see subsequent HM) (Step 4).

The probability of finding the minimum vector, $x^* = (0.08984, -0.71266)$, which is one of the global optima, increased with the number of searches (see HM after 50, 100, and 1000 searches in Table 1). Finally, after 4870 searches, the HS algorithm improvised a near-optimal harmony vector, $x = (0.08984, -0.71269)$, that had a function value of -1.0316285, as shown in Table 1 and Fig. 9. The HS algorithm also found the other optimal point ($-0.08984, 0.71266$) using a different random number seed.

The HS algorithm incorporates the structure of current meta-heuristic optimization algorithms. It preserves the history of past vectors (HM) similar to the tabu search, and is able to vary the adaptation rate (HMCR) from the beginning to the end of the computations, which resembles simulated annealing. It also

Table 1
Optimal results for the Six-Hump Camelback function using the HS algorithm

Rank	x_1	x_2	$f(x)$	x_1	x_2	$f(x)$	x_1	x_2	$f(x)$
	Initial HM			Subsequent HM			HM after 50 searches		
1	3.183	−0.400	169.95	3.183	−0.400	169.95	0.80558	−0.400	0.94270
2	−6.600	5.083	26274.83	3.183[a]	8.666[a]	22454.67[a]	0.80558	2.301	94.65
3	6.667	7.433	37334.24	−6.600	5.083	26274.83	0.80558	2.322	98.47
4	6.767	8.317	46694.70	6.667	7.433	37334.24	0.81676	2.419	117.35
5	−7.583	5.567	60352.77	6.767	8.317	46694.70	−0.88333	2.561	145.66
6	7.767	4.700	67662.40	−7.583	5.567	60352.77	−0.88333	2.589	152.54
7	8.250	2.750	95865.20	7.767	4.700	67662.40	3.074	−1.833	157.57
8	−8.300	8.533	120137.09	8.250	2.750	95865.20	3.183	−0.400	169.95
9	−9.017	−8.050	182180.00	−8.300	8.533	120137.09	3.183	−1.755	191.78
10	−9.500	3.333	228704.72	−9.017	−8.050	182180.00	3.183	2.308	271.38
	HM after 100 searches			HM after 1000 searches			HM after 4870 searches		
1	0.31672	0.40000	−0.2838402	0.09000	−0.71143	−1.0316159	0.08984[b]	−0.71269[b]	−1.0316285[b]
2	0.23333	0.32581	−0.2439473	0.09028	−0.71143	−1.0316149	0.09000	−0.71269	−1.0316284
3	0.26504	0.32581	−0.1951466	0.08863	−0.71143	−1.0316119	0.09000	−0.71277	−1.0316283
4	0.23333	0.28628	−0.1561579	0.09081	−0.71143	−1.0316114	0.09013	−0.71269	−1.0316281
5	0.35011	0.30594	0.0128968	0.09000	−0.71446	−1.0316020	0.08951	−0.71269	−1.0316280
6	0.26504	0.22232	0.0238899	0.09028	−0.71446	−1.0316019	0.08951	−0.71277	−1.0316279
7	0.35011	0.28628	0.0581726	0.09081	−0.71446	−1.0316000	0.08951	−0.71279	−1.0316278
8	0.31883	0.25029	0.0705802	0.09000	−0.71062	−1.0315942	0.09028	−0.71269	−1.0316277
9	0.35011	0.23078	0.1768801	0.08863	−0.71446	−1.0315939	0.08980	−0.71300	−1.0316275
10	0.54693	0.28628	0.5600001	0.09028	−0.71062	−1.0315928	0.09000	−0.71300	−1.0316274

[a] Denotes a new harmony improvised in first search and the corresponding function value.
[b] Denotes the best solution vector (0.08984, −0.71269) and the corresponding optimum point (−1.0316285) found using the HS algorithm.

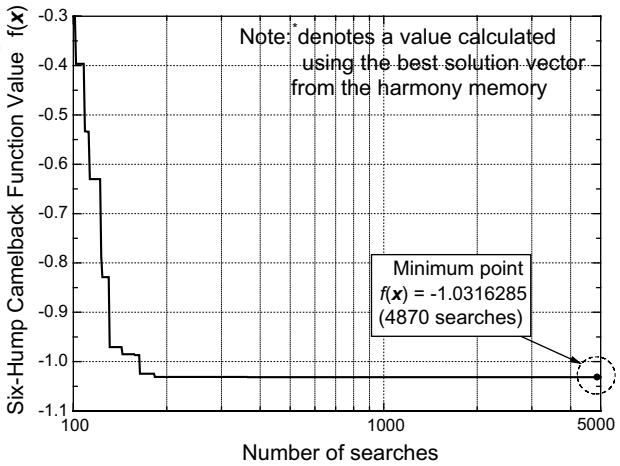


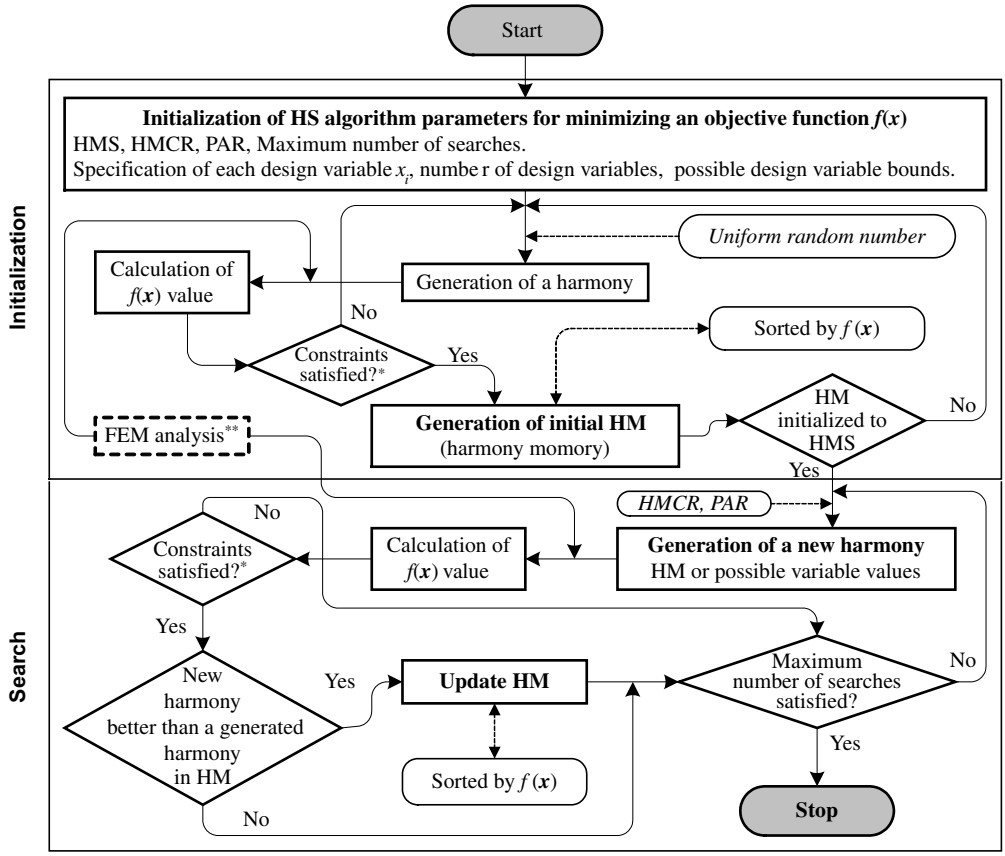
Fig. 9. Minimum value convergence history for the Six-Hump Camelback function.

considers several vectors simultaneously in a manner similar to the GA. However, the major difference between the GA and the HS algorithm is that the latter generates a new vector from all the existing vectors (all

harmonies in the HM), while the GA generate a new vector from only two of the existing vectors (parents). Similar to univariate models of the PMBGA, the HS meta-heuristic algorithm can independently consider each component variable in a vector when it generates a new vector. Unlike the PMBGA, however, the HS algorithm does not require probability models to estimate a distribution of promising design variable values (individuals) selected from the population. In addition, the HS meta-heuristic algorithm selects a promising design variable value from the HM as well as all possible value ranges; the PMBGA selects promising individuals solely from the population.

4. HS algorithm-based approach for engineering optimization

According to the HS meta-heuristic algorithm process, solutions to engineering optimization problems with continuous design variables can be obtained by: (a) defining value bounds for the design variables, (b) generating the initial harmony memory (HM), (c) improvising a new harmony, (d) evaluating the objective function subject to the constraint functions, and (e) updating the initialized HM. Here, we will focus on the HS algorithm mechanisms used to arrive at the optimum values.



Note: * Constraints are not required for unconstrained function optimization problems and the hydrologic model parameter calibration.
** The FEM analysis is only required for structural size and configuration optimization problems.

Fig. 10. HS algorithm-based engineering optimization design procedure.

Fig. 10 shows the detailed procedure of the HS meta-heuristic algorithm-based method used to determine the optimal design values in engineering optimization problems. The detailed procedure can be divided into the following two steps.

(1) *Step 1: Initialization.* HS algorithm parameters, such as HMS, HMCR, PAR, maximum number of searches, number of design variables, and design variable bounds, are initialized. Harmonies (i.e., solution vectors) are then randomly generated within the possible variable bounds until the number of harmonies equals the size of the HM. In this step, the initial HM is generated based on the objective function values subjected to the constraint functions and sorted by the objective function values.

(2) *Step 2: Search.* A new harmony is improvised from the initially generated HM or possible variable values using the HMCR and PAR parameters (see Fig. 6). These parameters are introduced to allow the solution to escape from local optima and to improve the global optimum prediction of the HS algorithm. The objective function $f(x)$ is calculated using the new harmony, and its suitability is evaluated using the constraint functions. If the constraint functions are satisfied and if the new harmony is better than the previous worst harmony, the new harmony is included in the HM and the previous worst harmony is excluded from the HM. The HM is then sorted by the objective function values. The computations terminate when the search number criterion is satisfied. If not, this step is repeated.

5. Engineering optimization examples

The computational procedures described above have been implemented in a FORTRAN computer program that can be applied to engineering optimization problems with continuous design variables. In this study, various standard benchmark engineering optimization examples including function minimization problems and structural optimization problems from the literature will be presented to demonstrate the efficiency and robustness of the proposed HS meta-heuristic algorithm method. These examples include six unconstrained and six constrained function minimization problems, five structural optimization examples, including pressure vessel design, welded beam design, truss sizing optimization, truss configuration optimization, and hydrologic model parameter calibration. For all examples presented in this paper, the HS algorithm parameters were set as follows: harmony memory size (HMS) = 20, harmony memory consideration rate (HMCR) = 0.90, and pitch adjusting rate (PAR) = 0.35.

5.1. Unconstrained function minimization examples

5.1.1. Rosenbrock function

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

The Rosenbrock function is probably the best known test case [28]. Due to a long narrow and curved valley present in the function, as shown in Fig. 11, gradient-based algorithms may require a large number of iterations before the minimum solution $x^* = (1.0, 1.0)$ with a function value equal to $f^*(x) = 0.0$ is found. The HS algorithm was applied to the Rosenbrock function problem using bounds between -10.0 and 10.0 for the two design variables, x_1 and x_2, given by Eq. (10). The HS algorithm-based method found 20 different near-optimal solution vectors (i.e., the values of the two design variables) after 50,000 searches that took approximately 2 min on a Pentium 600 MHz PC. The best solution vector was $x = (0.1000000000\text{E}{+}01, 0.1000002384\text{E}{+}01)$ with a corresponding function value equal to $f(x) = 5.6843418860\text{E}{-}10$, as shown in Table 2. The HS result is very close to the optimal solution, i.e., $f^*(x) = 0.0$.

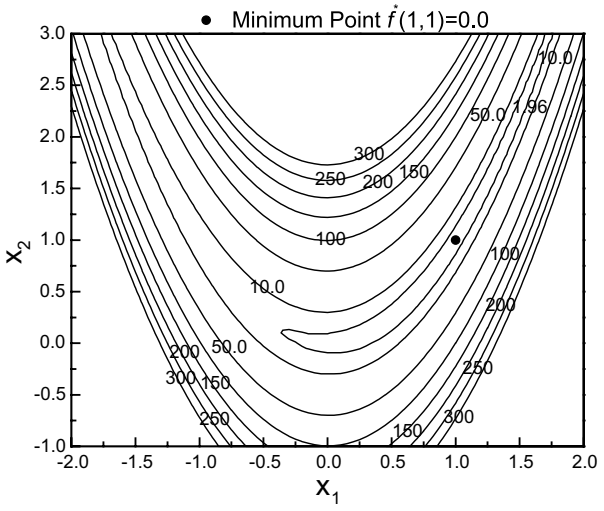


Fig. 11. Rosenbrock function.

Table 2
Optimal results of the unconstrained function minimization examples obtained using the HS algorithm

Examples	Functions	Optimal solutions	Best HS solutions
1	Rosenbrock function (Eq. (10))	$f^*(x) = 0$, $x_1^* = 1.0$, $x_2^* = 1.0$	$f(x) = 5.6843418860\text{E}{-10}$, $x_1 = 0.1000000000\text{E}{+01}$, $x_2 = 0.1000002384\text{E}{+01}$
2	Goldstein and Price function-I (Eq. (11))	$f^*(x) = 3.0$, $x_1^* = 1.0$, $x_2^* = -1.0$	$f(x) = 0.3000000000\text{E}{+01}$, $x_1 = -0.0000087289$, $x_2 = -1.0000001192$
3	Goldstein and Price function-II (Eq. (12))	$f^*(x) = 1.0$, $x_1^* = 3.0$, $x_2^* = 4.0$	$f(x) = 0.1000000000\text{E}{+01}$, $x_1 = 2.9998245239$, $x_2 = 4.0001201630$
4	Easton and Fenton function (Eq. (13))	$f^*(x) = 1.74$, $x_1^* = 1.7435$, $x_2^* = 2.0297$	$f(x) = 1.74415$, $x_1 = 1.74338$, $x_2 = 2.03050$
5	Wood function (Eq. (14))	$f^*(x) = 0.0$, $x_1^* = 1.0$, $x_2^* = 1.0$, $x_3^* = 1.0$, $x_4^* = 1.0$	$f(x) = 4.8515\text{E}{-09}$, $x_1 = 1.0000349$, $x_2 = 1.0000702$, $x_3 = 0.9999628$, $x_4 = 0.9999260$
6	Powell quartic function (Eq. (15))	$f^*(x) = 0.0$, $x_1^* = 0.0$, $x_2^* = 0.0$, $x_3^* = 0.0$, $x_4^* = 0.0$	$f(x) = 0.1254032468\text{E}{-11}$, $x_1 = -0.0008828875$, $x_2 = 0.0000882906$, $x_3 = -0.0004372409$, $x_4 = -0.0004372455$

5.1.2. Goldstein and Price function I (with four local minima)

$$f(x) = \{1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\} \times \{30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\}. \quad (11)$$

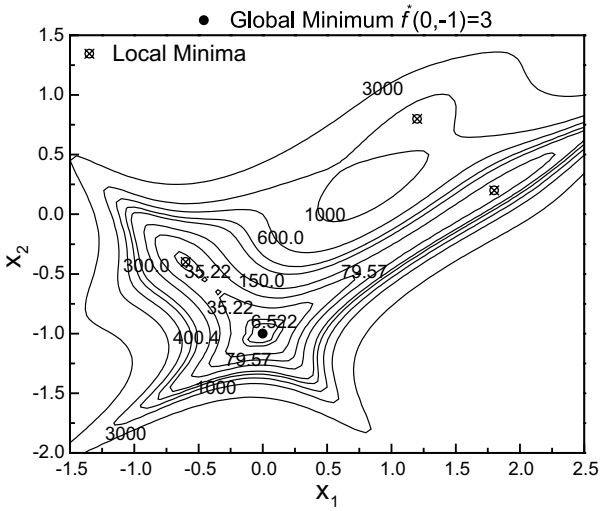


Fig. 12. Goldstein and Price function I (with four local minima).

This function is an eighth-order polynomial in two variables that is well behaved near each minimum value. As shown in Fig. 12, however, the function has four local minima, one of which is global, as follows: $f(1.2, 0.8) = 840.0$, $f(1.8, 0.2) = 84.0$, $f(-0.6, -0.4) = 30.0$, and $f^*(0, -1.0) = 3.0$ (global minimum) [29]. In this example, the bounds for two design variables (x_1 and x_2) were set between -5.0 and 5.0. After 40,000 searches, the best solution vector found by the HS algorithm was $x =$ (-0.0000087289, -1.0000001192) with a corresponding function value of $f(x) = 0.3000000000\text{E+}01$, as presented in Table 2. These values are also very close to the global minimum point.

5.1.3. Goldstein and Price function II (with many local minima)

$$f(x) = \exp\left\{\frac{1}{2}(x_1^2 + x_2^2 - 25)^2\right\} + \sin^4(4x_1 - 3x_2) + \frac{1}{2}(2x_1 + x_2 - 10)^2. \tag{12}$$

This function has many local minima [29]. The minimum solution of the function is located at point $x^* = (3, 4)$ with an objective function value equal to $f^*(x) = 1.0$. When applying the HS algorithm to the function given by Eq. (12), the two design variables x_1 and x_2 were initially structured with random values bounded between -5.0 and 5.0. The algorithm found 20 different solution vectors after approximately 45,000 iterations, and arrived at a best solution when $x = (2.9998245239, 4.0001201630)$, as shown in Table 2. The corresponding objective function value was $f(x) = 0.1000000000\text{E+}01$, which was also very close to the global optimal value.

5.1.4. Eason and Fenton's gear train inertia function

$$f(x) = \left\{12 + x_1^2 + \frac{1 + x_2^2}{x_1^2} + \frac{x_1^2 x_2^2 + 100}{(x_1 x_2)^4}\right\}\left(\frac{1}{10}\right). \tag{13}$$

The function shown in Eq. (13) consists of a minimization problem for the inertia of a gear train. It is one function from the set used by Eason and Fenton [30]. The minimum of the function is located at $x^* = (1.7435, 2.0297)$ with a corresponding objective function value of $f^*(x) = 1.74$, as illustrated in

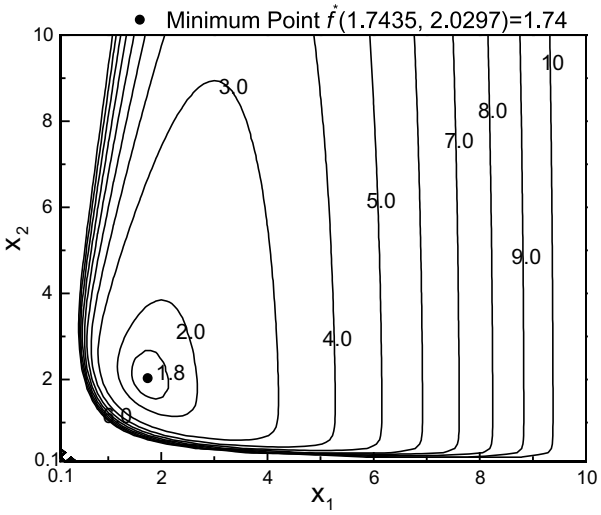


Fig. 13. Eason and Fenton gear inertia problem.

Fig. 13. The HS algorithm was applied to the gear train inertia function problem using bounds between 0.0 and 10.0 for the two design variables x_1 and x_2 given by Eq. (13). After a few seconds (approximately 800 searches), a minimum solution vector of $x = (1.74338, 2.03050)$ with a corresponding function value of $f(x) = 1.74415$ was obtained. This solution vector was well optimized towards the optimal point.

5.1.5. Wood function

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1). \quad (14)$$

This function is a fourth-degree polynomial that is a particularly good test of convergence criteria and simulates a feature of many physical problems quite well [31]. The minimum solution of the function is obtained at $x^* = (1, 1, 1, 1)$; the corresponding objective function value is $f^*(x) = 0.0$. When applying the HS algorithm to the Wood function, the four design variables, x_1 through x_4, were initially structured with random values bounded between -5.0 and 5.0. The algorithm found 20 different solution vectors after approximately 70,000 searches; the best objective function value of $f(x) = 4.8515\text{E}{-09}$ was obtained at $x = (1.0000349, 1.0000702, 0.9999628, 0.9999260)$, as shown in Table 2. These values are quite close to the global optimal point.

5.1.6. Powell quartic function

$$f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4. \quad (15)$$

As the second derivative of the Powell quartic function, shown in Eq. (15), becomes singular at the minimum point, it is quite difficult to obtain the minimum solution (i.e., $f^*(0, 0, 0, 0) = 0.0$) using gradient-based algorithms [32]. The HS algorithm was applied using bounds between -5.0 and 5.0 for the four design variables (x_1–x_4). After approximately 100,000 searches, a minimum solution vector of $x = (-0.0008828875, 0.0000882906, -0.0004372409, -0.0004372455)$ with a corresponding function value equal to $f(x) = 0.1254032468\text{E}{-11}$ was obtained (see Table 2). This HS solution vector was also well optimized towards the optimal point.

5.2. Constrained function minimization examples

5.2.1. Constrained function I

$$\begin{aligned} &f(x) = (x_1 - 2)^2 + (x_2 - 1)^2 \\ &\text{subject to,} \\ &g_1(x) = x_1 - 2x_2 + 1 = 0, \\ &g_2(x) = -x_1^2/4 - x_2^2 + 1 \geqslant 0, \\ &-10 \leqslant x_1 \leqslant 10, \quad -10 \leqslant x_2 \leqslant 10. \end{aligned} \tag{16}$$

This problem, originally introduced by Braken and McCormick [33], is a relatively simple constrained minimization problem. The optimum solution is obtained at $x^* = (0.82288, 0.91144)$ with an objective function value equal to $f^*(x) = 1.3935$. Homaifar et al. [34] solved this problem using GA and compared the result of GA with the optimum solution. Fogel [35] also compared the result of evolutionary programming with that of the GA. After approximately 40,000 searches, the HS algorithm-based method found 20 different solution vectors. Table 3 shows the best solution vector from the HS algorithm and also provides the results obtained using the GA (Homaifar et al. [34]) and the evolutionary programming (Fogel [35]). The best vector found using the HS meta-heuristic algorithm was $x = (0.8343, 0.9121)$, and the corresponding objective function value was 1.3770. The HS result is very close to the optimal solution and outperforms other two results not only in the aspect of the objective function values but also in the aspect of the constraint accuracy.

5.2.2. Constrained function II

$$\begin{aligned} &f(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \\ &\text{subject to,} \\ &g_1(x) = 4.84 - (x_1 - 0.05)^2 - (x_2 - 2.5)^2 \geqslant 0, \\ &g_2(x) = x_1^2 + (x_2 - 2.5)^2 - 4.84 \geqslant 0, \\ &0 \leqslant x_1 \leqslant 6, \quad 0 \leqslant x_2 \leqslant 6. \end{aligned} \tag{17}$$

This function is a minimization problem with two design variables and two inequality constraints [36]. The unconstrained objective function $f(x)$ has a minimum solution at $(3, 2)$ with a corresponding function value equal to zero. Due to the presence of the constraints given by Eq. (17) (i.e., g_1 and g_2), however, the unconstrained solution is not feasible; the constrained minimum solution is located at $x^* = (2.246826, 2.381865)$ with an objective function value equal to $f^*(x) = 13.59085$, as shown in Fig. 14. The HS algorithm was applied to constrained function II, and the initial HM was randomly generated from possible variables bounded between 0.0 and 6.0, as shown in Fig. 14. After approximately 15,000 searches, the best solution

Table 3
Optimal results of the constrained function I

Methods	Optimal design variables (x)		Constraints		Objective function value $f(x)$
	x_1	x_2	g_1	g_2	
Homaifar et al. [34]	0.8080	0.8854	3.7×10^{-2}	5.2×10^{-2}	1.4339
Fogel [35]	0.8350	0.9125	1.0×10^{-2}	-7.0×10^{-2}	1.3772
Present study	0.8343	0.9121	5.0×10^{-3}	5.4×10^{-3}	1.3770
Optimal solution	0.82288	0.91144	7.05×10^{-9}	1.73×10^{-8}	1.3935

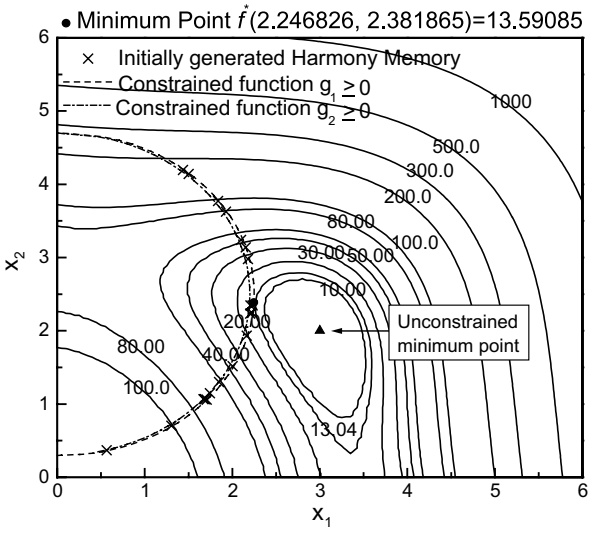


Fig. 14. Constrained function-II.

vector was $x = (2.246840, 2.382136)$ with a corresponding function value of $f(x) = 13.590845$. The HS best solution was compared to the previous solutions reported by Deb [36] in Table 4. Deb solved this function problem using the hybrid GA-based method with tournament selection (TS-R method) and with Powell and Skolnick's constraint handling method (PS method), respectively, and obtained the best solution of 13.59085 using TS-R method, which showed an excellent agreement with the optimal solution. The best solution using the HS meta-heuristic algorithm is also very close to the optimal solution.

5.2.3. Constrained function III

$$
\begin{aligned}
&f(x) = 5.357847x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792141\\
&\text{subject to,}\\
&g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.002205x_3x_5 \geqslant 0,\\
&g_2(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.002205x_3x_5 \leqslant 92,\\
&g_3(x) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \geqslant 90,\\
&g_4(x) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \leqslant 110,\\
&g_5(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \geqslant 20,\\
&g_6(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leqslant 25,\\
&78 \leqslant x_1 \leqslant 102, \quad 33 \leqslant x_2 \leqslant 45, \quad 27 \leqslant x_i \leqslant 45 \quad (i = 3, 4, 5).
\end{aligned}
\tag{18}
$$

This function has five design variables and six inequality constraints, and has been solved by many researchers [34,36–40]. The best known optimum solution to the function is $x^* = (78.0, 33.0, 29.995, 45.0, 36.776)$, and the corresponding objective function value is $f^*(x) = -30665.5$ [38]. The constraints g_2 and g_5 are active at this solution. Table 5 lists the optimal values of the function problem obtained by the HS algorithm-based method, and compares them with earlier results reported by Homaifar et al. [34], Coello [39], Deb [36] and Shi and Eberhart [40]. Homaifar et al. and Coello obtained a best solution function value of $f(x) = -30005.7$ and $f(x) = -31020.859$, respectively, using the GA-based methods. Shi and Eberhart [40] solved the problem using a modified particle swarm optimization algorithm,

Table 4
Optimal results of the constrained function-II

Methods	Optimal design variables (x)		Objective function value $f(x)$
	x_1	x_2	
Deb [36]			
GA with PS ($R = 0.01$)	Unavailable	Unavailable	13.58958
GA with PS ($R = 1$)			13.59108
GA with TS-R			13.59085
Present study	2.246840	2.382136	13.590845
Optimal solution	2.246826	2.381865	13.59085

Table 5
Optimal results of the constrained function-III

Methods	Optimal design variables (x)					Objective function value $f(x)$
	x_1	x_2	x_3	x_4	x_5	
Homaifar et al. [34]	80.39	35.07	32.05	40.33	33.34	−30005.700
Coello [39]	78.0495	33.007	27.081	45.00	44.94	−31020.859
Shi and Eberhart [40]	78.0	33.0	27.07099	45.0	44.969	−31025.561
Deb [36]	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	−30665.500
Present study	78.0	33.0	29.995	45.0	36.776	−30665.500
Optimal solution	78.0	33.0	29.995	45.0	36.775	−30665.500

and obtained a best solution of −31025.561. Deb [36], who recently solved this problem using an efficient constraint handling method for the GA, reported an objective function value of $f(x) = -30665.5$, which corresponds to the optimum solution reported in the literature [38]. The HS algorithm-based method found a best solution vector of $x = (78.0, 33.0, 29.995, 45.0, 36.776)$ with a function value equal to $f(x) = -30665.5$ after approximately 65,000 searches, as shown in Table 5. The best solution obtained using the HS algorithm was the same as the optimal solution.

5.2.4. Constrained function IV

$$
\begin{aligned}
& f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \\
& \text{subject to,} \\
& g_1(x) = 127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geqslant 0, \\
& g_2(x) = 282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geqslant 0, \\
& g_3(x) = 196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geqslant 0, \\
& g_4(x) = -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geqslant 0, \\
& -10 \leqslant x_i \leqslant 10 \quad (i = 1\text{–}7).
\end{aligned} \tag{19}
$$

This function problem has seven design variables and four nonlinear constraint functions [36,41]. The optimal solution is obtained at $x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$ with a corresponding function value equal to $f^*(x) = 680.6300573$. The constraints g_1 and g_4 are active at this solution. Table 6 presents the best solution of the constraint function IV obtained using the HS algorithm-based method, and compares the HS result with previous best solutions reported by Michalewicz [41] and Deb [36]. Michalewicz [41] reported a best objective function value of

Table 6
Optimal results of the constrained function-IV

Optimal design variables (x) and objective function value ($f(x)$)	Michalewicz [41]	Deb [36]	Present study	Optimal solution
x_1	Unavailable	Unavailable	2.32345617	2.330499
x_2			1.951242	1.951372
x_3			−0.448467	−0.4775414
x_4			4.3619199	4.365726
x_5			−0.630075	−0.6244870
x_6			1.03866	1.038131
x_7			1.605384	1.594227
$f(x)$	680.642	680.63446	680.6413574	680.6300573

$f(x) = 680.642$, which required a total of 350,070 function evaluations. Deb [36] obtained an optimized function value of $f(x) = 680.63446$, which is the best solution obtained using GA-based methods. In the HS algorithm-based method, a best solution vector of $x = (2.32345617, 1.951242, -0.448467, 4.3619199, -0.630075, 1.032866, 1.605384)$ with an objective function value equal to $f(x) = 680.6413574$ was obtained after approximately 160,000 searches. The HS solution is comparable to the previous result obtained by Deb [36].

5.2.5. Constrained function V

$$
\begin{aligned}
& f(x) = x_1 + x_2 + x_3 \\
& \text{subject to,} \\
& g_1(x) = 1 - 0.0025(x_4 + x_6) \geqslant 0, \\
& g_2(x) = 1 - 0.0025(x_5 + x_7 - x_4) \geqslant 0, \\
& g_3(x) = 1 - 0.01(x_8 - x_5) \geqslant 0, \\
& g_4(x) = x_1 x_6 - 833.33252 x_4 - 100 x_1 + 83333.333 \geqslant 0, \\
& g_5(x) = x_2 x_7 - 1250 x_5 - x_2 x_4 + 1250 x_4 \geqslant 0, \\
& g_6(x) = x_3 x_8 - x_3 x_5 + 2500 x_5 - 1250000 \geqslant 0, \\
& 100 \leqslant x_1 \leqslant 10000, \quad 1000 \leqslant x_2, \quad x_3 \leqslant 10000, \quad 10 \leqslant x_i \leqslant 1000 \quad (i = 4\text{–}8).
\end{aligned} \tag{20}
$$

This constrained function has eight variables and six inequality constraints, and has been solved previously by Deb [36] and Michalewicz [41]. The optimal solution of the function is $x^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)$, and the corresponding function value is $f^*(x) = 7049.330923$. All six constraints are active at this solution. Table 7 lists the best solution of the constraint function V obtained by the HS algorithm-based method, and compares the HS result with previous best solutions reported by Michalewicz [41] and Deb [36]. Michalewicz tried to optimize this function using seven different GA-based methods. Three of them found solutions that were somewhat close to the optimal solution. The best solution obtained by Michalewicz had an objective function value equal to $f(x) = 7377.976$, which is approximately 4.66% worse than the optimal solution. Deb [36], who also solved this function using an efficient constraint handling method for the GA based on a penalty function approach that did not require any penalty parameters, obtained a best objective value equal to $f(x) = 7060.221$ using a maximum of 320,080 function evaluations. The HS algorithm found 20 different solution vectors from the HM after 150,000 searches. The best solution vector, as shown in Table 7, was $x = (500.0038, 1359.3110, 5197.9595, 174.7263, 292.0817, 224.7054, 282.6446, 392.0817)$ with a function value equal to $f(x) = 7057.274414$, which is very close to the true optimal solution (i.e., $f(x)^* = 7049.33923$).

Table 7
Optimal results of the constrained function-V

Optimal design variables (x) and objective function value ($f(x)$)	Michalewicz [41]	Deb [36]	Present study	Optimal solution
x_1	Unavailable	Unavailable	500.0038	579.3167
x_2			1359.3110	1359.943
x_3			5197.9595	5110.071
x_4			174.7263	182.0174
x_5			292.0817	295.5985
x_6			224.7054	217.9799
x_7			282.6446	286.4162
x_8			392.0817	395.5979
$f(x)$	7377.976	7060.221	7057.274414	7049.330923

5.2.6. Constrained function VI

$$
\begin{aligned}
f(x) &= x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 \\
&\quad + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10}F - 7)^2 + 45 \\
&\text{subject to,} \\
g_1(x) &= 105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geqslant 0, \\
g_2(x) &= -10x_1 + 8x_2 + 17x_7 - 2x_8 \geqslant 0, \\
g_3(x) &= 8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geqslant 0, \\
g_4(x) &= -3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geqslant 0, \\
g_5(x) &= -5x_x^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geqslant 0, \\
g_6(x) &= -x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \geqslant 0, \\
g_7(x) &= -0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geqslant 0, \\
g_8(x) &= 3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geqslant 0, \\
&-10 \leqslant x_1 \leqslant 10 \quad (i = 1\text{–}10).
\end{aligned}
\tag{21}
$$

Table 8
Optimal results of the constrained function-VI

Optimal design variables (x) and objective function value ($f(x)$)	Michalewicz [41]	Deb [36]	Present study	Optimal solution
x_1	Unavailable	Unavailable	2.155225	2.171996
x_2			2.407687	2.363683
x_3			8.778069	8.773926
x_4			5.102078	5.95984
x_5			0.967625	0.9906548
x_6			1.357685	1.430574
x_7			1.287760	1.321644
x_8			9.800438	9.828726
x_9			8.187803	8.280092
x_{10}			8.256297	8.375927
$f(x)$	24.690	24.37248	24.3667946	24.3062091

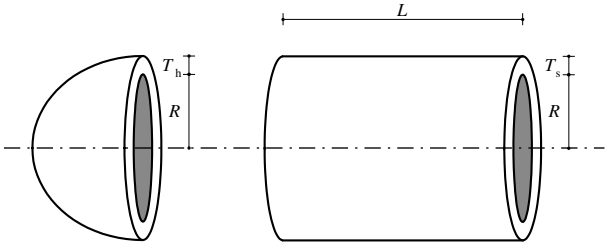
This function has 10 variables and eight constraints [36,41]. The optimal solution to this function is $x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$, $f^*(x) = 24.3062091$. The first six constraints are active at this solution. The HS algorithm-based method was applied to the constraint function VI. Table 8 lists the best solution of the function problem obtained by the HS meta-heuristic algorithm, and compares the HS result with previous best solutions reported by Michalewicz [41] and Deb [36]. Michalewicz solved this problem using different constraint handling techniques based on the GA. The best objective function value, which was achieved with a multi-level penalty function approach, was $f(x) = 24.690$. This solution required a total of 350,070 function evaluations. The problem was also solved by Deb [36] using an efficient constraint handling method for the GA. The best function value reported by Deb was $f(x) = 24.37248$; this solution required a similar number of function evaluations to the method proposed by Michalewicz. In the HS algorithm-based method, a best objective function value equal to $f(x) = 24.3667946$ was obtained after approximately 230,000 searches, as shown in Table 8. The HS solution was slightly better than the previous results obtained by Michalewicz [41] and Deb [36].

5.3. Structural engineering optimization examples

5.3.1. Pressure vessel design

The pressure vessel design, shown in Fig. 15, was previously analyzed by Sandgren [42] who first proposed this problem and Wu and Chow [43] to minimize the total cost of the material, forming and welding of a cylindrical vessel. There are four design variables: x_1 (T_s, shell thickness), x_2 (T_h, spherical head thickness), x_3 (R, radius of cylindrical shell) and x_4 (L, shell length). T_s (=x_1) and T_h (=x_2) are integer multipliers of 0.0625 in. in accordance with the available thickness of rolled steel plates, and R (=x_3) and L (=x_4) have continuous values of $40 \leqslant R \leqslant 80$ in. and $20 \leqslant L \leqslant 60$ in., respectively. The mathematical formulation of the optimization problem is as follows:

$$
\begin{aligned}
&f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^3 + 3.1611x_1^2x_4 + 19.84x_1^2x_3 \\
&\text{subject to,} \\
&g_1(x) = 0.0193x_3 - x_1 \leqslant 0, \\
&g_2(x) = 0.00954x_3 - x_2 \leqslant 0, \\
&g_3(x) = 750.0 \times 1728.0 - \pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 \leqslant 0, \\
&g_4(x) = x_4 - 240.0 \leqslant 0, \\
&g_5(x) = 1.1 - x_1 \leqslant 0, \\
&g_6(x) = 0.6 - x_2 \leqslant 0.
\end{aligned}
\tag{22}
$$



T_s: Shell thickness, T_h: Spherical head thickness
R: Radius of cylindrical shell, L: Shell length,

Fig. 15. Schematic of pressure vessel.

Table 9
Optimal results for pressure vessel design

Optimal design variable (in.) and typical constraint functions	Sandgren [42]	Wu and Chow [43]	Present study
T_s (=x_1)	1.125	1.125	1.125
T_h (=x_2)	0.625	0.625	0.625
R (=x_3)	48.97	58.1978	58.2789
L (=x_4)	106.72	44.2930	43.7549
$g_1(x)$	−0.1799	−0.00178	−0.00022
$g_2(x)$	−0.1578	−0.06979	−0.06902
$g_3(x)$	97.760	−974.3	−3.71629
$g_4(x)$	−133.28	−195.707	−196.245
Cost ($)	7980.894	7207.494	7198.433

The HS algorithm method was applied to the pressure vessel optimization problem and the optimal results were compared to earlier solutions reported by Sandgren [42] and Wu and Chow [43], as shown in Table 9. Sandgren achieved the optimal values of $7980.894 using the branch and bound method. However, the solution does not satisfy the third constraint $g_3(x)$ that should be less than or equal to zero. Wu and Chow obtained the minimum cost of $7207.494 using the GA-based approach. The HS algorithm-based method achieves a design with a best solution vector of $(1.125, 0.625, 58.2789, 43.7549)$ and a minimum cost of $7198.433 without violating any constraint. The results obtained using the HS algorithm were better optimized than both earlier solutions.

5.3.2. Welded beam design

The welded beam structure, shown in Fig. 16, is a practical design problem that has been often used as a benchmark for testing different optimization methods [36,44–47]. The structure consists of beam A and the weld required to hold the beam to member B. The objective is to find a feasible set of dimensions h, l, t, and b to carry a certain load (P) while maintaining the minimum total fabrication cost. The mathematical formulation of the objective function $f(x)$, which is the total fabricating cost mainly comprised of the set-up, welding labor, and material costs, is as follows:

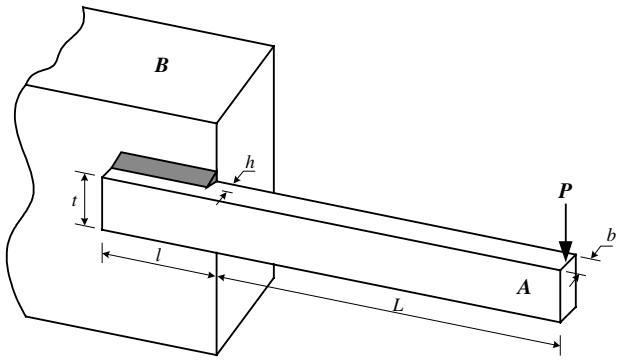


Fig. 16. Welded beam structure.

$$
\begin{aligned}
&f(x) = 1.10471h^2l + 0.04811tb(L + l)\\
&\text{subject to,}\\
&g_1(x) = \tau_d - \tau(x) \geqslant 0,\\
&g_2(x) = \sigma_d - \sigma(x) \geqslant 0,\\
&g_3(x)b - h \geqslant 0,\\
&g_4(x) = P_c(x) - P \geqslant 0,\\
&g_5(x) = 0.25 - \delta(x) \geqslant 0,\\
&0.125 \leqslant h \leqslant 5, \quad 0.1 \leqslant l, \quad t \leqslant 10, \quad 0.1 \leqslant b \leqslant 5,
\end{aligned}
\tag{23}
$$

where x are the design variables (i.e., h, l, t, and b), L is the overhang length of the beam (14 in.), g_1–g_5 are the constraint functions, τ_d is the allowable design shear stress of weld (13,600 psi), $\tau(x)$ is the weld shear stress, σ_d is the allowable design yield stress for the bar material (30,600 psi), $\sigma(x)$ is the maximum bar bending stress, $P_c(x)$ is the bar buckling load, P is the loading condition (6000 1 b), and $\delta(x)$ is the bar end deflection. The terms of $\tau(x)$, $\sigma(x)$, $P_c(x)$, and $\delta(x)$ are expressed as follows:

$$
\begin{aligned}
&\tau(x) = \sqrt{(\tau'(x))^2 + (\tau''(x))^2 + l\tau'(x)\tau''(x) \Big/ \sqrt{0.25(l^2 + (h + t)^2)}},\\
&\sigma(x) = \frac{504\,000}{t^2b}, \quad P_c(x) = 64\,746(1 - 0.0282346t)tb^3, \quad \delta(x) = \frac{2.1952}{t^3b},
\end{aligned}
\tag{24}
$$

where

$$
\tau' = \frac{6000}{\sqrt{2}hl} \quad \text{and} \quad \tau'' = \frac{6000(14 + 0.5l)\sqrt{0.25(l^2 + (h + t)^2)}}{2\{0.707hl(l^2/12 + 0.25(h + t)^2)\}}.
$$

The HS algorithm was applied to the welded beam design optimization problem. The optimal results are compared to earlier solutions reported by Ragsdell and Phillips [46] and Deb [36,47] in Table 10. Ragsdell and Phillips [46] compared optimal results obtained using different optimization methods that were based mainly on mathematical optimization algorithms. The best results obtained using these methods, such as the APPROX, DAVID, GP, SIMPLEX, and RANDOM algorithms, are listed in Table 10. Deb solved this problem using an efficient constraint handling method for the GA based on a penalty function approach [36] and a binary GA [47].

Table 10
Optimal results of the welded beam design

Methods	Optimal design variables (x)				Cost
	h	l	T	b	
Ragsdell and Phillips [46]					
APPROX	0.2444	6.2189	8.2915	0.2444	2.38
DAVID	0.2434	6.2552	8.2915	0.2444	2.38
GP	0.2455	6.1960	8.2730	0.2455	2.39
SIMPLEX	0.2792	5.6256	7.7512	0.2796	2.53
RANDOM	0.4575	4.7313	5.0853	0.6600	4.12
Deb [36]	Unavailable	Unavailable	Unavailable	Unavailable	2.38
Deb [47]	0.2489	6.1730	8.1789	0.2533	2.43
Present study	0.2442	6.2231	8.2915	0.2443	2.38

The HS result, which was obtained after approximately 110,000 searches, is comparable to the best solutions listed in Table 10, including the APPROX and DAVID algorithms, and the Deb [36] method. In the APPROX and DAVID methods, however, first-order derivatives of each design variable were required, which limits the application of these techniques in a variety of problems.

5.3.3. Structural sizing and configuration design optimization

The structural optimization problem for truss or frame structures can usually be described using three different types of design variables: (1) sizing variables, (2) geometric variables, and (3) topological variables [48]. Sizing optimization is concerned with determining the cross-sectional size of structural members. Configuration optimization searches for a set of geometric and sizing variables using a given topology, while topology optimization selects the topology from various structural types. In general, topology optimization is a combinatorial optimization problem.

In this study, pure sizing and configuration optimization problems for a truss structure with continuous design variables were introduced to demonstrate the efficiency of the HS meta-heuristic algorithm-based method. The general form of these truss optimization problems is

$$f(x) = \text{structural weight} = \gamma \sum_{i=1}^{n} L_i A_i, \tag{25}$$

$$\text{subject to, } {}_{\mathrm{L}}g_j \leqslant g_j \leqslant {}_{\mathrm{U}}g_j, \quad j = 1, 2, \ldots, q, \tag{26}$$

where γ is the material density, L_i is the member length, A_i is the member cross-sectional area, and ${}_{\mathrm{L}}g_j$ and ${}_{\mathrm{U}}g_j$ are the lower and upper bounds on the inequality constrained function g_j. Pure sizing optimization involves arriving at optimum values for member cross-sectional areas A that minimize an objective function $f(x)$, i.e., the structural weight. This minimum design has to satisfy q inequality constraints that limit the size of the design variables and the structural response. Configuration optimization involves simultaneously arriving at optimum values for the nodal coordinates R and member cross-sectional areas A that minimize the structural weight. This minimum design also has to satisfy q inequality constraints.

For the truss sizing and configuration optimization problems presented in this study, the upper and lower bounds on the constraint functions of Eq. (26) included the following: (1) nodal coordinates (${}_{\mathrm{L}}R_i \leqslant R_i \leqslant {}_{\mathrm{U}}R_i$, $i = 1, \ldots, m$), (2) member cross-sectional areas (${}_{\mathrm{L}}A_i \leqslant A_i \leqslant {}_{\mathrm{U}}A_i$, $i = 1, \ldots, n$), (3) member stresses (${}_{\mathrm{L}}\sigma_i \leqslant \sigma_i \leqslant {}_{\mathrm{U}}\sigma_i$, $i = 1, \ldots, n$), (4) nodal displacements (${}_{\mathrm{L}}\delta_i \leqslant \delta_i \leqslant {}_{\mathrm{U}}\delta_i$, $i = 1, \ldots, m$), and (5) member buckling stresses (${}_{\mathrm{b}}\sigma_i \leqslant \sigma_i \leqslant 0$, $i = 1, \ldots, n$). Here, the nodal coordinates R_i were only required for the truss configuration optimization.

A 10-bar planar truss and an 18-bar planar truss that have been used previously by many authors as standard test cases were chosen to test the pure size and configuration optimization abilities of the HS algorithm, respectively. The HS algorithm structural optimization design procedure is illustrated in Fig. 10. The FEM displacement method was used for structural analysis. For both examples, the maximum number of searches considered was 50,000.

5.3.3.1. Sizing optimization: 10-bar plane truss. The 10-bar cantilever truss, shown in Fig. 17, was previously analyzed by Schmit and Farshi [49], Schmit and Miura [50], Venkayya [51], Dobbs and Nelson [52], Rizzi [53], and Khan and Willmert [54]. Most of these studies used mathematical optimization methods. The material density was 0.1 lb/in.^{3} and the modulus of elasticity was 10,000 ksi. The members were subjected to stress limitations of ±25 ksi, and displacement limitations of ±2.0 in. were imposed on all nodes in both the x- and y-directions. No design variable linking was used; thus there were 10 independent design variables. In this example, the single loading condition shown in Fig. 17 was considered. A minimum member cross-sectional area of 0.1 in.^{2} was enforced.

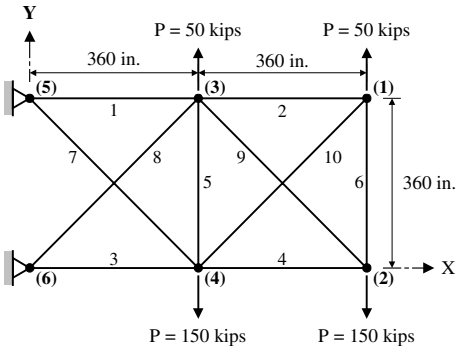


Fig. 17. 10-bar plane truss.

Table 11
Optimal results of the 10-bar plane truss

Methods	Optimal cross-sectional areas A (in.^{2})										Weight (lb)
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	
Schmit and Farshi [49]	24.29	0.100	23.35	13.66	0.100	1.969	12.67	12.54	21.97	0.100	4691.84
Schmit and Miura [50]											
NEW SUMT	23.55	0.100	25.29	14.36	0.100	1.970	12.39	12.81	20.34	0.100	4676.96
CONMIN	23.55	0.176	25.20	14.39	0.100	1.967	12.40	12.86	20.41	0.100	4684.11
Venkayya [51]	25.19	0.363	25.42	14.33	0.417	3.144	12.08	14.61	20.26	0.513	4895.60
Dobbs and Nelson [52]	25.81	0.100	27.23	16.65	0.100	2.024	12.78	14.22	22.14	0.100	5059.70
Rizzi [53]	23.53	0.100	25.29	14.37	0.100	1.970	12.39	12.83	20.33	0.100	4676.92
Khan and Willmert [54]	24.72	0.100	26.54	13.22	0.108	4.835	12.66	13.78	18.44	0.100	4792.52
Present study	23.25	0.102	25.73	14.51	0.100	1.977	12.21	12.61	20.36	0.100	4668.81

The HS algorithm-based method found 20 different solution vectors (i.e., values of the 10 design independent variables) after 50,000 searches. Table 11 shows the best solution vector and also provides a comparison between the optimal design results reported in the literature and those obtained in the present study. The best HS solution vector was $x = (23.25, 0.102, 25.73, 14.51, 0.100, 1.977, 12.21, 12.61, 20.36, 0.100)$ with an objective function value equal to $f(x) = 4668.81$ lb. This vector was obtained after approximately 20,000 searches that took 3 min on a Pentium 600 MHz computer. The HS algorithm obtained better solutions than the other methods reported in the literature.

5.3.3.2. Configuration optimization: 18-bar plane truss. The 18-bar cantilever plane truss, shown in Fig. 18, is one of the most popular classical optimization design problems. Due to its simple configuration, this structure has been used as a benchmark to verify the efficiency of various optimization methods (Imai and Schmit [55], Felix [56], Yang [57], Soh and Yang [58], Rajeev and Krishnamoorthy [59], and Yang and Soh [60]). In the benchmark problem, the material density was 0.1 lb/in.^{3} and the modulus of elasticity was 10,000 ksi. The cross-sectional areas of the members were categorized into four groups: (1) $A_1 = A_4 = A_8 = A_{12} = A_{16}$, (2) $A_2 = A_6 = A_{10} = A_{14} = A_{18}$, (3) $A_3 = A_7 = A_{11} = A_{15}$, and (4) $A_5 = A_9 = A_{13} = A_{17}$. The single loading condition considered in this study was a set of vertical loads, $P = 20$ kips, acting on the upper nodal points of the truss, as illustrated in Fig. 18. The lower nodes 3, 5, 7, and 9 were allowed to move in any direction in the x–y plane. Thus, there were a total of 12 independent design variables that included four sizing and eight coordinate variables.

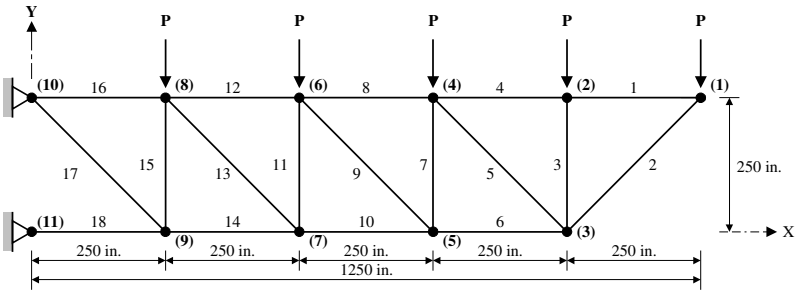


Fig. 18. 18-bar plane truss (initial configuration).

The purpose of the optimization is to design a configuration for the truss to produce a minimum design weight that meets both the allowable stress and the buckling constraints. In this problem, the allowable tensile and compressive stresses were 20 ksi. The Euler buckling compressive stress limit for truss member i used for the buckling constraints was computed as:

$$_{\mathrm{b}}\sigma_i = \frac{-KEA_i}{L_i^2}, \quad i = 1, \ldots, 18, \tag{27}$$

where K is a constant determined from the cross-sectional geometry, E is the modulus of elasticity of the material, and L_i is the member length. In this study, the buckling constant was taken to be $K = 4$, and the bounds on the member cross-sectional areas were 3.5 and 18.0 in.^{2}.

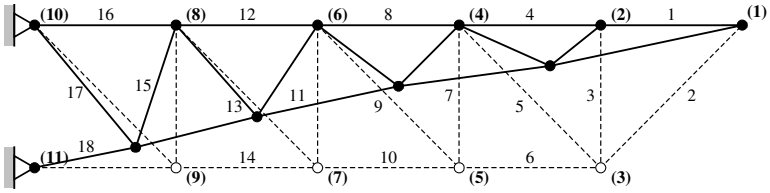
The HS found 20 different solution vectors after 50,000 searches that contained the values of the 12 design independent variables. Table 12 lists the best solution vector from the HS algorithm and also the results obtained using other mathematical methods (Imai and Schmit [55] and Felix [56]) and GA-based approaches (Yang [57], Soh and Yang [58], Rajeev and Krishnamoorthy [59], and Yang and Soh [60]). The best vector found using the HS approach was $x = (12.65, 17.22, 6.17, 3.55, 903.1, 174.3, 630.3, 136.3, 402.1, 90.5, 195.3, 30.6)$, and the corresponding objective function value (weight of the structure) was 4515.6 lb. This optimal configuration was obtained after approximately 25,000 searches, and is illustrated in Fig. 19. The HS algorithm obtained better solutions than the six other algorithms reported in the literature.

Table 12
Optimal results of the 18-bar plane truss

Methods	Optimal cross-sectional areas A (in.^{2})				Optimal coordinates R (in.)								W[a] (lb)
	G_1[b]	G_2[b]	G_3[b]	G_4[b]	X_3	Y_3	X_5	Y_5	X_7	Y_7	X_9	Y_9	
Imai and Schmit [55]	11.24	15.68	7.93	6.49	891.1	143.6	608.2	105.4	381.7	57.1	181.0	−3.2	4667.9
Felix [56]	11.34	19.28	10.97	5.30	994.6	162.3	747.4	102.9	482.9	33.0	221.7	17.1	5713.0
Yang [57]	12.61	18.10	5.47	3.54	914.5	183.0	647.0	147.4	414.2	100.4	200.0	31.9	4552.8
Soh and Yang [58]	12.59	17.91	5.50	3.55	909.8	184.5	640.3	147.8	410.0	97.0	200.9	32.0	4531.9
Rajeev and Krishnamoorthy [59]	12.50	16.25	8.00	4.00	891.9	145.3	610.6	118.2	385.4	72.5	184.4	23.4	4616.8
Yang and Soh [60]	12.33	17.97	5.60	3.66	907.2	184.2	643.3	149.2	413.9	102.0	202.1	30.9	4520.0
Present study	12.65	17.22	6.17	3.55	903.1	174.3	630.3	136.3	402.1	90.5	195.3	30.6	4515.6

[a] W shows the optimal structural weight.

[b] G_1, G_2, G_3 and G_4 represent four cross-sectional area groups, i.e., (1) $A_1 = A_4 = A_8 = A_{12} = A_{16}$, (2) $A_2 = A_6 = A_{10} = A_{14} = A_{18}$, (3) $A_3 = A_7 = A_{11} = A_{15}$, and (4) $A_5 = A_9 = A_{13} = A_{17}$.



(Optimal structural weight $W = 4515.6$ lb)

Fig. 19. 18-bar plane truss (optimal configuration).

5.3.4. Hydrologic model parameter calibration

The parameter calibration of hydrologic model is an error minimization problem between observed and computed hydrologic values. The objective function in this example is the residual sum of squares (SSQ) between observed and computed hydrologic values (river outflows), which is expressed as

$$\text{Minimize } f(x) = \text{SSQ} = \sum_{t=1}^{n} (O_{\text{oro}} - O_{\text{cro}})^2, \tag{28}$$

where O_{oro} is an observed river outflow; and O_{cro} is a computed river outflow.

The river outflow O_{cro} can be calculated using the nonlinear Muskingum model as follows:

$$O_{\text{cro}} = \left(\frac{1}{1-h}\right)\left(\frac{S_t}{K}\right)^{1/r} - \left(\frac{h}{1-h}\right) I_t. \tag{29}$$

In the model, channel storage S_t and river inflows I_t can be calculated and provided, respectively. However, three model parameters (K, h, and r) should be calibrated.

The hydrologic model parameter calibration was previously investigated by Gill [61], Tung [62], Yoon and Padmanabhan [63], and Mohan [64]. Gill [61] suggested a technique for finding the values of the three parameters using the least squares method. Tung [62] also proposed a technique for parameter calibration using various curve fitting techniques. He combined the Hooke–Jeeves pattern search and the Davidon–Fletcher–Powell techniques. Yoon and Padmanabhan [63] proposed a technique for linearity determination of hydrologic data and applied it to find parameter values. Mohan [64] suggested a calibration technique for the nonlinear Muskingum model using the GA. The results showed that the calibration technique using the GA is better than the above techniques and does not require the process of assuming initial values close to the optimum.

In order to apply the HS heuristic algorithm to parameter calibration problem, in this example, three model parameters, i.e., K, h, and r were considered as continuous design variables. The bounds of three parameters were 0.01–0.20 for K, 0.2–0.3 for h, and 1.5–2.5 for r, respectively. These parameter bounds are based on the various results in the literature. Table 13 lists the optimal values of each design variable obtained by the HS algorithm for the hydrologic model parameter calibration, and compares them with earlier results

Table 13
Optimal results for hydrologic model parameter calibration

Optimal design variables	Gill [61]	Tung [62]	Yoon and Padmanabhan [63]	Mohan [64]	Present study
K	0.0100	0.0764	0.0600	0.1033	0.0870
h	0.2500	0.2677	0.2700	0.2813	0.2870
r	2.3470	1.8978	2.3600	1.8282	1.8661
SSQ	143.60	45.54	43.26	38.23	36.77

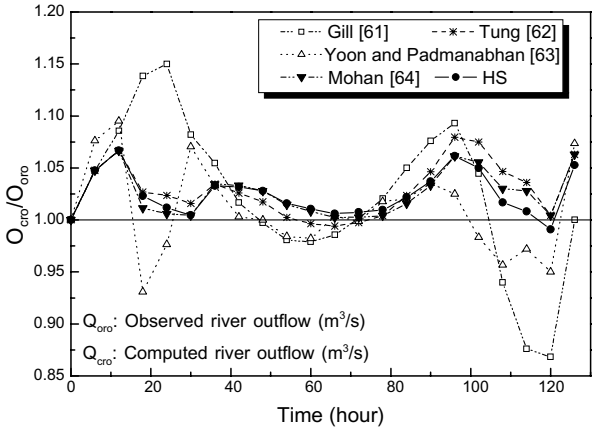


Fig. 20. Comparison of optimal results for hydrologic model parameter calibration.

reported by Gill [61], Tung [62], Yoon and Padmanabhan [63], and Mohan [64]. The HS algorithm achieved a design with a best solution vector of $(0.087, 0.287, 1.8661)$ and a SSQ of 36.77 after approximately 20,000 searches. Fig. 20 shows the ratio of actual data between O_{oro} (observed river outflow) and O_{cro} (computed river outflow) in terms of river outflow time, together with those estimated by previous endeavors. The computed river outflow data were calculated by Eq. (29) using the optimal values for the nonlinear Muskingum model parameters (K, h, r) shown in Table 13. As shown in Table 13 and Fig. 20, the optimal design obtained using the HS algorithm outperforms the other four SSQ results reported in literature.

6. Conclusions

The recently developed HS meta-heuristic optimization algorithm was conceptualized using the musical process of searching for a perfect state of harmony. Compared to gradient-based mathematical optimization algorithms, the HS algorithm imposes fewer mathematical requirements and does not require initial value settings of the decision variables. As the HS algorithm uses stochastic random searches, derivative information is also unnecessary. Furthermore, the HS algorithm generates a new vector, after considering all of the existing vectors based on the harmony memory considering rate (HMCR) and the pitch adjusting rate (PAR), whereas the GA only consider the two parent vectors. These features increase the flexibility of the HS algorithm and produce better solutions.

This paper described the new HS meta-heuristic algorithm-based approach for engineering optimization problems with continuous design variables. Various engineering optimization problems, including (1) six unconstrained and (2) six constrained function minimization problems, and (3) five structural optimization problems (i.e., pressure vessel design, welded beam design, truss sizing optimization, truss configuration optimization, and hydrologic model parameter calibration) were presented to demonstrate the effectiveness and robustness of the new algorithm compared to other optimization methods, especially meta-heuristic algorithm-based optimization methods. These examples revealed that the new HS algorithm is a global search algorithm that can be easily applied to various engineering optimization problems. The results obtained using the HS algorithm may yield better solutions than those obtained using current algorithms, such as conventional mathematical optimization algorithms or GA-based approaches. Our study, therefore, suggests that the new HS algorithm is potentially a powerful search and optimization technique for solving complex engineering optimization problems.

References

[1] L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial Intelligence Through Simulated Evolution, John Wiley, Chichester, UK, 1966.
[2] K. De Jong, Analysis of the behavior of a class of genetic adaptive systems, Ph.D. Thesis, University of Michigan, Ann Arbor, MI, 1975.
[3] J.R. Koza, Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems, Rep. No. STAN-CS-90-1314, Stanford University, CA, 1990.
[4] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
[5] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, Boston, MA, 1989.
[6] F. Glover, Heuristic for integer programming using surrogate constraints, Decision Sci. 8 (1) (1977) 156–166.
[7] S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[8] Z.W. Geem, J.-H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68.
[9] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equations of state calculations by fast computing machines, J. Chem. Phys. 21 (1953) 1087–1092.
[10] M. Pincus, A Monte Carlo method for the approximate solution of certain types of constrained optimization problems, Oper. Res. 18 (1970) 1225–1228.
[11] H.-P. Schwefel, On the evolution of evolutionary computation, in: J. Zurada, R. Marks, C. Robinson (Eds.), Computational Intelligence: Imitating Life, IEEE Press, New York, NY, 1994, pp. 116–124.
[12] J. Kennedy, R.C. Eberhat, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, No. IV, IEEE Service Center, Piscataway, NJ, 1995, pp. 1942–1948.
[13] M. Pelikan, D.E. Goldberg, F.G. Lobo, A survey of optimization by building and using probabilistic models, IlliGAL Rep. No. 99018, University of Illinois Genetic Algorithms Laboratory, Urbana, IL, 1999.
[14] P. Larranaga, J.A. Lozano, Estimation of Distribution Algorithms, Kluwer Academic Publishers, New York, 2002.
[15] O. Jiri, Parallel estimation of distribution algorithms, Ph.D. Thesis, BRNO University of Technology, Czech Republic, 2002.
[16] S. Baluja, Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning, Technical Report No. CMU-CS-94-163, Carnegie Mellon University, PA, 1994.
[17] H. Muhlenbein, The equation for response to selection and its use for prediction, Evolution. Comput. 5 (3) (1997) 303–346.
[18] G.R. Harik, F.G. Lobo, D.E. Goldburg, The compact genetic algorithm, in: Proceedings of the International Conference on Evolutionary Computation 1998 (ICEC'98), IEEE Service Center, Piscataway, NJ, 1998, pp. 523–528.
[19] J.S. De Bonet, C.L. Isbell, P. Viola, MIMC: finding optima by estimating probability densities, in: M.C. Mozer, M.I. Jordan, T. Petsche (Eds.), Advanced in Neural Information Processing Systems, vol. 9, MIT Press, Cambridge, MA, 1997.
[20] S. Baluja, S. Davies, Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space, in: Proceedings of the 14th International Conference on Machine Learning, Morgan Kaufmann, 1997, pp. 30–38.
[21] M. Pelikan, H. Muhlenbein, The bivariate marginal distribution algorithm, in: R. Roy, T. Furuhashi, P.K. Chawdhry (Eds.), Advances in Soft Computing-Engineering Design and Manufacturing, Springer-Verlag, London, 1999, pp. 521–535.
[22] G. Harik, Lingkage learning via probabilistic modeling in the ECGA, IlliGAL Rep. No. 99010, University of Illinois Genetic Algorithms Laboratory, Urbana, IL, 1999.
[23] H. Muhlenbein, T. Mahnig, Convergence theory and applications of the factorized distribution algorithm, J. Comput. Inform. Technol. 7 (1999) 19–32.
[24] M. Pelikan, D.E. Goldberg, E. Cantu-Paz, BOA: the Bayesian optimization algorithm, in: W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, R.E. Smith (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99), vol. I, Morgana Kaufmann Publisher, San Francisco, CA, 1999, pp. 525–532.
[25] H. Muhlenbein, T. Mahnig, FDA-A scalable evolutionary algorithm for the optimization of additively decomposed functions, Evolution. Comput. 7 (4) (1999) 353–376.
[26] R. Etxeberria, P. Larranaga, Global optimization with Bayesian networks, in: II Symposium on Artificial Intelligence (CIMAF99), Special Session on Distributions and Evolutionary Optimization, 1999, pp. 332–339.
[27] L.C.W. Dixon, G.P. Szego, Towards Global Optimization, North Holland, Amsterdam, 1975.
[28] H.H. Rosenbrock, An automatic method for finding the greatest or least value of a function, Comput. J. 3 (3) (1960) 175–184.
[29] A.A. Goldstein, J.F. Price, On descent from local minima, Math. Comput. 25 (1971) 569–574.
[30] E.D. Eason, R.G. Fenton, A comparison of numerical optimization methods for engineering design, ASME J. Engrg. Ind. 96 (1) (1974) 196–200.
[31] A.R. Colville, A comparative study of nonlinear programming, Tech. Report No. 320-2949, IBM New York Scientific Center, 1968.
[32] A.R. Conn, K. Scheinberg, P.L. Toint, On the convergence of derivative-free methods for unconstrained optimization, in: A. Iserles, M. Buhmann (Eds.), Approximation Theory and Optimization: Tributes to M.J.D. Powell, Cambridge University Press, Cambridge, UK, 1997.

[33] J. Bracken, G.P. McCormick, Selected Applications of Nonlinear programming, John Wiley & Sons, New York, 1968.
[34] A. Homaifar, S.H.-V. Lai, X. Qi, Constrained optimization via genetic algorithms, Simulation 62 (4) (1994) 242–254.
[35] D.B. Fogel, A comparison of evolutionary programming and genetic algorithms on selected constrained optimization problems, Simulation 64 (6) (1995) 399–406.
[36] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Engrg. 186 (2000) 311–338.
[37] Z. Michalewicz, M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, Evolution. Comput. 4 (1) (1996) 1–13.
[38] D.M. Himmelblau, Applied Nonlinear Programming, McGraw-Hill, New York, 1972.
[39] C.A. Coello, Use of a self-adaptive penalty approach for engineering optimization problems, Comput. Ind. 41 (2) (2000) 113–127.
[40] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of the International Congress on Evolutionary Computation 1998 (ICEC'98), IEEE Service Center, Piscataway, NJ, 1998, pp. 69–73.
[41] Z. Michalewicz, Genetic algorithms, numerical optimization, and constraints, in: L. Esheman (Ed.), Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kauffman, San Mateo, 1995, pp. 151–158.
[42] E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization, J. Mech. Des. ASME 112 (1990) 223–229.
[43] S.J. Wu, P.T. Chow, Genetic algorithms for nonlinear mixed discrete-integer optimization problems via meta-genetic parameter optimization, Engrg. Optim. 24 (1995) 137–159.
[44] G.V. Reklaitis, A. Ravindran, K.M. Ragsdell, Engineering Optimization Methods and Applications, Wiley, New York, 1983.
[45] J.N. Siddall, Analytical Decision-Making in Engineering Design, Prentice-Hall, Englewood Cliffs, NJ, 1972.
[46] K.M. Ragsdell, D.T. Phillips, Optimal design of a class of welded structures using geometric programming, ASME J. Engrg. Ind. Ser. B 98 (3) (1976) 1021–1025.
[47] K. Deb, Optimal design of a welded beam *via* genetic algorithms, AIAA J. 29 (11) (1991).
[48] B.H. Topping, Shape optimization of skeletal structures: a review, ASCE J. Struct. Engrg. 109 (8) (1983) 1933–1951.
[49] L.A. Schmit Jr., B. Farshi, Some approximation concepts for structural synthesis, AIAA J. 12 (5) (1974) 692–699.
[50] L.A. Schmit Jr., H. Miura, Approximation concepts for efficient structural synthesis, NASA CR-2552, NASA, Washington, DC, 1976.
[51] V.B. Venkayya, Design of optimum structures, Comput. Struct. 1 (1–2) (1971) 265–309.
[52] M.W. Dobbs, R.B. Nelson, Application of optimality criteria to automated structural design, AIAA J. 14 (10) (1976) 1436–1443.
[53] P. Rizzi, Optimization of multi-constrained structures based on optimality criteria, in: Conference on AIAA/ASME/SAE 17th Structures, Structural Dynamics, and Materials, King of Prussia, PA, 1976.
[54] M.R. Khan, K.D. Willmert, W.A. Thornton, An optimality criterion method for large-scale structures, AIAA J. 17 (7) (1979) 753–761.
[55] K. Imai, A.L. Schmit Jr., Configuration optimization of trusses, ASCE J. Struct. Div. 107 (ST5) (1981) 745–756.
[56] J.E. Felix, Shape optimization of trusses subjected to strength, displacement, and frequency constraints, Master 's Thesis, Naval Postgraduate School, 1981.
[57] J.P. Yang, Development of genetic algorithm-based approach for structural optimization, Ph.D. Thesis, Nanyang Technology University, Singapore, 1996.
[58] C.K. Soh, J.P. Yang, Fuzzy controlled genetic algorithm search for shape optimization, ASCE J. Comput. Civ. Engrg. 10 (2) (1996) 143–150.
[59] S. Rajeev, C.S. Krishnamoorthy, Genetic algorithm-based methodologies for design optimization of trusses, ASCE J. Struct. Engrg. 123 (3) (1997) 350–358.
[60] J.P. Yang, C.K. Soh, Structural optimization by genetic algorithms with tournament selection, ASCE J. Comput. Civ. Engrg. 11 (3) (1997) 195–200.
[61] M.A. Gill, Flood routing by the Muskingum method, J. Hydrol. 36 (1978) 353–363.
[62] Y.K. Tung, River flood routing by nonlinear Muskingum method, J. Hydraul. Engrg. ASCE 111 (12) (1985) 1147–1460.
[63] J. Yoon, G. Padmanabhan, Parameter estimation of linear and nonlinear Muskingum models, J. Water Resour. Plan. Manage. ASCE 119 (5) (1993) 600–610.
[64] S. Mohan, Parameter estimation of nonlinear Muskingum models using genetic algorithm, J. Hydraul. Engrg. ASCE 123 (2) (1997) 137–142.