SELF-ADAPTIVE HARMONY SEARCH APPLIED TO ENGINEERING SYSTEM DESIGN

Fran Sérgio Lobato, fslobato@feq.ufu.br
School of Chemical Engineering, FEQUI,
Valder Steffen Jr, vsteffen@mecanica.ufu.br
School of Mechanical Engineering, FEMEC,
Universidade Federal de Uberlândia, UFU, P.O. Box 593, 38400-902, Uberlândia-MG, Brazil

Abstract. *A meta-heuristic algorithm known as harmony search (HS), mimicking the improvisation process of music players, has been recently developed. This approach does not require derivative information and uses stochastic random search instead of a gradient search. In addition, the HS algorithm exhibits simple concept, few parameters to handle, and easy implementation. In this paper, the Self-adaptive Harmony Search (SHS) algorithm proposed by Pan et al. (2010) is used in engineering system design. In this approach, a new improvisation scheme is developed so that the good information captured in the current global best solution can be well utilized to generate new harmonies. The harmony memory consideration rate and pitch adjustment rate are dynamically adapted by learning mechanisms that have been included in the optimization strategy. The distance bandwidth is dynamically adjusted to favor exploration in the early stages and exploitation during the final stages of the search process. The results obtained are compared with those obtained from other classical evolutionary approaches, namely Genetic Algorithms and Particle Swarm Optimization.*

Keywords*: Self-adaptive Harmony Search, Optimization, Optimal Engineering System Design.*

1. INTRODUCTION

The Harmony search (HS) algorithm is a meta-heuristic approach inspired by the natural musical performance process that occurs when a musician searches for a better state of harmony. In this algorithm, proposed by Geem *et al.* (2001), the solution vector is analogous to the harmony in music, and the local and global search schemes are analogous to musicians' improvisations. According to Mahdavi *et al.* (2007) and Omran and Mahdavi (2008), the HS algorithm, in comparison to several meta-heuristics, requires fewer mathematical requirements and can be easily adapted for solving various kinds of engineering optimization problems. In addition, numerical comparisons demonstrated that the evolution of the HS algorithm is faster than other heuristics so that the genetic algorithms, for example (Lee and Geem, 2005; Lee *et al.*, 2005).

Various applications using the HS algorithm are found in the literature, such as structural optimization (Lee and Geem, 2005; Lee *et al.*, 2005), design optimization of water distribution networks (Geem, 2006), vehicle routing (Geem *et al.* 2005), combined heat and power economic dispatch (Vasebi *et al.*, 2007), and transport energy modeling (Ceylan *et al.*, 2008).

According to Mahdavi *et al.* (2007), the main disadvantage of the HS algorithm is its difficulty in performing local search. In this sense, a new mechanism to increase its ability for enhancing solution accuracy and convergence rate has been proposed. Mahdavi *et al.* (2007) presented an improved HS algorithm, the so-called Improved Harmony Search (IHS), by introducing a strategy to dynamically tune the key parameters. Omran and Mahdavi (2008) proposed a global HS algorithm, denoted as Global Harmony Search (GHS), by borrowing the concept of swarm intelligence from other techniques. The numerical experiments revealed that both improved variants of the method could find better solutions as compared to the basic HS algorithm. Particularly, the GHS algorithm outperformed the IHS algorithm.

It is important to emphasize that in spite of the performance and the number of applications encompassed when fixed parameters are used by meta-heuristic approaches, there is no guarantee that premature convergence will be avoided (Coelho and Mariani, 2006). In this context, the Self-Adaptive Harmony Search (SAHS) algorithm proposed by Pan *et al.* (2010) is used in engineering system design. This work is organized as follows. Section 2 provides a brief literature overview of both of the HS and the SAHS algorithms. The results and discussions are described in Section 3. Finally, the conclusions and suggestions for future work conclude the paper.

2. SELF-ADAPTIVE HARMONY SEARCH

This section describes the proposed Self-Adaptive Harmony Search (SAHS) algorithm. First, a brief overview of the HS is provided, and finally the modification procedures of the proposed SAHS algorithm are presented.

2.1. Harmony Search

In the HS canonical algorithm each solution is called a ‘‘harmony” and is represented by an *N*-dimension real vector. An initial population of harmony vectors are randomly generated and stored in a harmony memory (HM). Then a *new harmony* candidate is generated from all the solutions in the HM by using a memory consideration rule, a pitch adjustment rule and a random re-initialization. Finally, the HM is updated by comparing the *new harmony* candidate with the worst harmony vector in the HM. The worst harmony vector is replaced by the new candidate vector if it is better than the worst harmony vector in the HM. The above process is repeated until a given termination criterion is met. The basic HS algorithm consists of three basic phases, namely, initialization, improvisation of a harmony vector and updating the HM. The main steps are described below (Lee and Geem, 2005).

Step 1. Initialize the problem and algorithm parameters. Consider the following optimization problem:

$$\min f(x) \text{ subject to } x_i \in X_i \quad (i=1, 2, ...,N) \tag{1}$$

where $f(x)$ is the objective function, $\mathbf{x}$ is the design variable vector, N is the number of design variables, x_i is the set of the possible range of values for each design variable, i.e., $x_i^L \le X_i \le x_i^U$ and x_i^L and x_i^U are the lower and upper bounds for each design variable. The HS algorithm parameters are also specified in this step, as follows: harmony memory size (HMS) - or the number of solution vectors in the harmony memory; harmony memory considered rate (HMCR); pitch adjusting rate (PAR); and the number of improvisations (NI) - or stopping criterion.

The harmony memory (HM) is a memory location where all the solution vectors (sets of decision variables) are stored. This HM is similar to the genetic pool in the GA (Geem *et al.*, 2002). Here, HMCR and PAR are parameters that are used to improve the solution vector. Both are defined in Step 3.

Step 2. Initialize the harmony memory. In this step, the HM matrix is filled with as many randomly generated solution vectors as the HMS

$$\text{HM}=\begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_{N-1}^1 & x_N^1 \\ x_1^2 & x_2^2 & \cdots & x_{N-1}^2 & x_N^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^{\text{HMS-1}} & x_2^{\text{HMS-1}} & \cdots & x_{N-1}^{\text{HMS-1}} & x_N^{\text{HMS-1}} \\ x_1^{\text{HMS}} & x_2^{\text{HMS}} & \cdots & x_{N-1}^{\text{HMS}} & x_N^{\text{HMS}} \end{bmatrix} \tag{2}$$

Step 3. Improvise a new harmony. A new harmony vector, $x'=(x_1' \; x_2' \; x_3' \; ... \; x_N')$, is generated based on three rules: (1) memory consideration, (2) pitch adjustment and (3) random selection. Generating a new harmony is called improvisation.

In the memory consideration, the value of the first decision variable (x_1') for the new vector is chosen from any of the values in the specified HM range ($x_1' - x_1^{\text{HMS}}$). Values of the other decision variables ($x_2' \; x_3' \; ... \; x_N'$) are chosen in the same manner. The HMCR, which varies between 0 and 1, is the rate of choosing a value from the historical values stored in the HM, while (1–HMCR) is the rate of randomly selecting a value from the possible range of values.

$$x_i' \leftarrow \begin{cases} x_i' \in \left\{ x_i^1, \; \cdots, \; x_i^{\text{HMS}} \right\} & \text{with probability HMCR,} \\ x_i' \in X_i & \text{with probability (1-HMCR)} \end{cases} \tag{3}$$

For example, a HMCR of 0.85 indicates that the HS algorithm will choose the decision variable value from historically stored values in the HM with an 85% probability or from the entire possible range with a (100–85)% probability. Every component obtained by the memory consideration is examined to determine whether it should be pitch-adjusted. This operation uses the PAR parameter, which is the rate of pitch adjustment, as follows:

$$\text{Pitch adjusting decision for } x_i' \leftarrow \begin{cases} \text{Yes} & \text{with probability PAR,} \\ \text{No} & \text{with probability (1-PAR)} \end{cases} \tag{4}$$

The value of (1–PAR) sets the rate of doing nothing. If the pitch adjustment decision for x_i' is YES, x_i' is replaced as follows:

$$x_i' \leftarrow x_i' \pm \text{rand}() \times b_w \tag{5}$$

where b_w is an arbitrary distance bandwidth and rand( ) is a random number between 0 and 1.

In step 3, HM consideration, pitch adjustment or random selection is applied to each variable of the new harmony vector.

Step 4. Update harmony memory. If the new harmony vector, $x'=(x_1' \, x_2' \, x_3' \, ... \, x_N')$, is better than the worst harmony in the HM, judged 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.

Step 5. Check stopping criterion. If the stopping criterion (maximum number of improvisations) is satisfied, computation is terminated. Otherwise, Steps 3 and 4 are repeated.

2.2. Self-Adaptive Harmony Search

In the HS algorithm, three control parameters HMCR, PAR and b_w are closely related to the problem being solved and the phase of the search process that may be either exploration or exploitation. In this work the Self-Adaptive Harmony Search (SAHS) is proposed, where the parameters are dynamically adapted with respect to the favorable evolution of the search process, as presented below.

According to Lee and Geem (2005) and Pan *et al.* (2010), HMCR describes the probability of choosing one value from the historic values stored in the HM. A large HMCR value is in favor of local search thereby increasing the convergence rate of the algorithm, while a small HMCR value increases the diversity of the harmony memory. A large PAR value favors passing the information of x_i' to next generation thereby enhancing the local exploitation ability of the algorithm around x_i', whereas a small PAR value enables the new harmony vector to select its dimensional values by perturbing the corresponding values in the harmony memory, thus enlarging the search area and diversity of the harmony memory. Since local exploitation and global exploration are always twisted together in the search process, it is difficult to fix the values for HMCR and PAR.

In this paper, HMCR and PAR are dynamically adapted to a suitable range by recording their historic values corresponding to generated harmonies entering the HM, as proposed by Pan *et al.* (2010). We assume that the HMCR (PAR) value is normally distributed in the range of [0.9, 1.0] ([0.0, 1.0]) with mean HMCR (PAR) and standard deviation 0.01 (0.05). Initially, HMCR (PAR) is set at 0.98 (0.9), and then SAHS starts with a HMCR (PAR) value generated according to the normal distribution. During the evolution, the HMCR (PAR) value associated with the generated harmony successfully replacing the worst member in the HM is recorded. After a specified number of generations, HMCR (PAR) is recalculated by averaging all the recorded HMCR (PAR) values during this period. With the new mean and the given standard deviation of 0.01 (0.05), new HMCR (PAR) value is produced and used in the subsequent iterations. The above procedure is repeated. As a result, an appropriate HMCR (PAR) value can be gradually learned to suit the particular problem and the particular phases of the search process.

The parameter b_w is a distance bandwidth for the continuous design variable. As observed by Pan *et al.* (2010), a large value is in favor of the algorithm searching in a large scope, while a small value is appropriate for fine-tuning of the best solution vectors. To well balance the exploration and exploitation of the proposed SGHS algorithm, the b_w value decreases dynamically with increasing generations (i_{gen}) as follows:

$$b_w \leftarrow \begin{cases} b_{wmax} - \dfrac{b_{wmax} - b_{wmin}}{i_{gen}} & \text{if } i < 0.5 \times i_{gen} \\ b_{wmin} & \text{if } i \geq 0.5 \times i_{gen} \end{cases} \tag{6}$$

where b_{wmax} and b_{wmin} are the maximum and minimum distance bandwidths, respectively, and i is the current generation. It should be emphasized that other strategies to parameters update can be found in the literature, for instance, Wang and Huang (2010), Sarvari and Zamanifar (2010) and Hasançebi et al. (2010).

3. RESULTS AND DISCUSSION

For evaluating the methodology used in this work, some practical points should be emphasized:

- In all case studies the following parameters for the HS were used: N equal to 20; HMCR equal to 0.9; PAR equal to 0.3; 10000 generations (improvisations), and penalization parameter equal to 10^8 (designed to penalize any constraint violation). In the SAHS b_{wmin} and b_{wmax} were considered equal to 0.005 and 100, respectively.
- In this paper, all case studies were run 20 times independently to obtain the values and standard deviations shown in the upcoming tables.

3.1. Welded beam design problem

The welded beam design problem is taken from Rao (1996) and He and Wang (2007), in which a welded beam is designed for minimum cost subject to constraints on shear stress (τ), bending stress in the beam, buckling load on the bar (P_c), end deflection of the beam (δ), and side constraints. There are four design variables as shown in Fig. 1, i. e., h (x_1), l (x_2), t (x_3), and b (x_4).

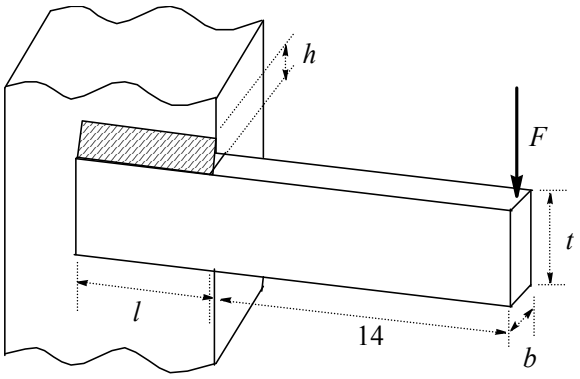

Figure 1: Welded beam design problem.

Mathematically, the problem can be formulated as follows (Rao, 1996):

$$\min\ f(x) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2) \tag{7}$$

subject to

$$g_1(x) = \tau(x) \text{-} 13000 \le 0 \tag{8}$$

$$g_2(x) = \sigma(x) \text{-} 30000 \le 0 \tag{9}$$

$$g_3(x) = x_1 \text{-} x_4 \le 0 \tag{10}$$

$$g_4(x) = 1.10471 x_1^2 + 0.04811 x_3 x_4 (14 + x_2) - 5 \le 0 \tag{11}$$

$$g_5(x) = 0.125 \text{-} x_1 \le 0 \tag{12}$$

$$g_6(x) = \delta(x) \text{-} 0.25 \le 0 \tag{13}$$

$$g_7(x) = 6000 \text{-} P_c(x) \le 0 \tag{14}$$

where

$$\tau(x) = \sqrt{(\tau_1)^2 + 2\tau_1\tau_2 \frac{x_2}{2R} + (\tau_2)^2}, \quad \tau_1 = \frac{6000}{\sqrt{2} x_1 x_2}, \quad \tau_2 = \frac{MR}{J}, \quad M = 6000\left(14 + \frac{x_2}{2}\right), \quad R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},$$

$$J = 2\left(\sqrt{2} x_1 x_2 \left(\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right)\right), \ \sigma(x) = \frac{504000}{x_4 x_3^2}, \ \delta(x) = \frac{2.1952}{x_4 x_3^3}, \ P_c(x) = 64746.022\left(1 \text{-} 0.028234 x_3\right) x_3 x_4^3.$$

The approaches applied to this problem include genetic algorithm with binary representation and traditional penalty function (Deb, 1991), a GA-based co-evolution model (Coello, 2000), and a co-evolutionary particle swarm optimization (He and Wang, 2007).

The following design space is adopted (He and Wang, 2007): 0.1 in $\leq x_1 \leq$ 2 in, 0.1 in $\leq x_2 \leq$ 10 in, 0.1 in $\leq x_3 \leq$ 10 in, 0.1 in $\leq x_4 \leq$ 2 in. The best solutions obtained by the above mentioned approaches are listed in Table 1. In this table, it can be seen that the best solution found by HS and SAHS has the same quality than the best solutions found by other techniques (Deb, 1991; Coello, 2000), but slightly inferior to the result obtained by He and Wang (2007). In addition, it is important to observe that the SAHS yields better results in terms of the number of objective function evaluations as compared with the HS algorithm. This characteristic demonstrates the efficacy of the methodology proposed.

Table 1. Comparison of the best solutions for the welded beam design problem using different techniques (N_{eval} is the number of objective function evaluations).

Design variables	Deb (1991)	Coello (2000)	He and Wang (2007)	HS (standard deviation)	SAHS (standard deviation)
x_1 (in)	0.248900	0.208800	0.202369	0.208796 (0.00026)	0.208795 (0.00011)
x_2 (in)	6.173000	3.420500	3.544214	3.412588 (0.00004)	3.412585 (0.00009)
x_3 (in)	8.178900	8.997500	9.048210	8.910008 (0.00005)	8.910004 (0.00001)
x_4 (in)	0.253300	0.210000	0.205723	0.210000 (0.00002)	0.210001 (0.00001)
g_1 (psi)	-5758.607	-0.337812	-12.839796	-23896.252	-23896.252
g_2 (psi)	-255.5769	-353.9026	-1.247467	-230.95874	-230.95874
g_3 (in)	-0.004400	-0.001200	-0.001498	-0.001204	-0.001204
g_4 ($)	-2.982866	-3.141865	-3.429347	-3.384378	-3.384378
g_5 (in)	-0.123900	-0.083800	-0.079381	-0.083796	-0.083796
g_6 (in)	-0.234160	-0.235649	-0.235536	-0.235222	-0.235222
g_7 (lb)	-44.65270	-363.2323	-11.681355	-808.56989	-808.56989
f($)	2.433116	1.748309	1.728024	1.7318117 (0.0001)	1.7318116 (0.0001)
N_{eval}	Unavailable	Unavailable	Unavailable	10020	8820

3.2. Tension/compression string design problem

This problem is from Arora (1989), Belegundu (1982) and He and Wang (2007). It is devoted to the minimization of the weight of a tension/compression spring as shown in Fig. 2. The design variables are the wire diameter d (x_1), the mean coil diameter D (x_2), and the number of active coils P (x_3).

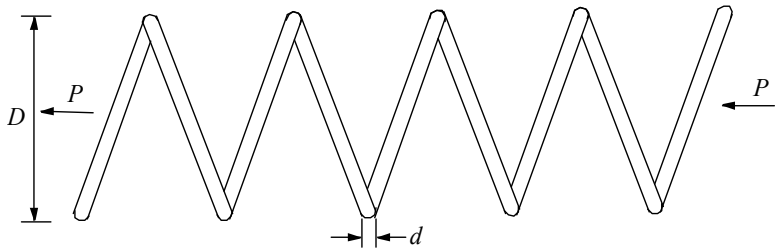

Figure 2: Tension/compression string design problem.

The mathematical formulation of this problem can be described as follows:

$$\min\ f(x) = (x_3 + 2)\, x_2 x_1^2 \tag{15}$$

subject to constraints on minimum deflection - Eq. (16), shear stress - Eq. (17), surge frequency - Eq. (18), limits on the outside diameter - Eq. (19), and on the side constraints:

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0 \tag{16}$$

$$g_2(x) = \frac{4x_2^3 - x_1 x_2}{12566\left(x_2 x_1^3 - x_1^4\right)} + \frac{1}{5108 x_1^2} - 1 \leq 0 \tag{17}$$

$$g_3(x) = 1 - \frac{140.45 x_1}{x_3 x_2^2} \leq 0 \tag{18}$$

$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \tag{19}$$

The approaches applied to this problem include eight different numerical optimization techniques (Belegundu, 1982), a numerical optimization technique called constraint correction at constant cost (Arora, 1989), a GA-based co-evolution model (Coello, 2000), and a co-evolutionary particle swarm optimization (He and Wang, 2007).

The following design space is adopted (He and Wang, 2007): 0.05 in $\leq x_1 \leq$ 2 in, 0.25 in $\leq x_2 \leq$ 1.3 in, 2 in $\leq x_3 \leq$ 15 in. Table 2 presents the best solutions obtained by the above mentioned approaches. In this table, it can be seen that the best solution found by HS and SAHS has the same quality of those obtained by other techniques.

Table 2. Comparison of the best solutions for the tension/compression spring design problem using different methods (N_{eval} is the number of objective function evaluations).

Design variables	Belegundu (1982)	Arora (1989)	He and Wang (2007)	HS (standard deviation)	SAHS (standard deviation)
x_1 (in)	0.050000	0.053396	0.051728	0.053513 (0.00002)	0.053528 (0.00001)
x_2 (in)	0.315900	0.399180	0.357644	0.402208 (0.00037)	0.402210 (0.00033)
x_3 (in)	14.25000	9.185400	11.244543	9.047566 (0.01082)	9.047565 (0.01090)
g_1 (in)	-0.000014	0.000019	-0.000845	-0.000020	-0.000010
g_2 (ksi)	-0.003782	-0.000018	-1.260E-05	-0.000000	-0.000000
g_3 (-)	-3.938302	-4.123832	-4.051300	-4.135091	-4.135099
g_4 (in)	-0.727090	-0.698283	-0.727090	-0.696185	-0.696100
f (lb)	0.012674	0.012730	0.012674	0.012724 (0.00001)	0.012725 (0.00001)
N_{eval}	Unavailable	Unavailable	Unavailable	10020	7820

3.3. Pressure vessel design problem

The pressure vessel design problem was proposed by Kannan and Kramer (1994) and is devoted to the minimization of the total cost of the specimen, including the cost of the material, forming and welding. A cylindrical vessel is capped at both ends by hemispherical heads as shown in Fig. 3. There are four design variables: T_s (x_1, thickness of the shell), T_h (x_2, thickness of the head), R (x_3, inner radius) and L (x_4, length of the cylindrical section of the vessel, not including the head). Among the four variables, T_s and T_h are integer multiples of 0.0625 in (corresponding to the available thicknesses of rolled steel plates), and R and L are continuous variables.

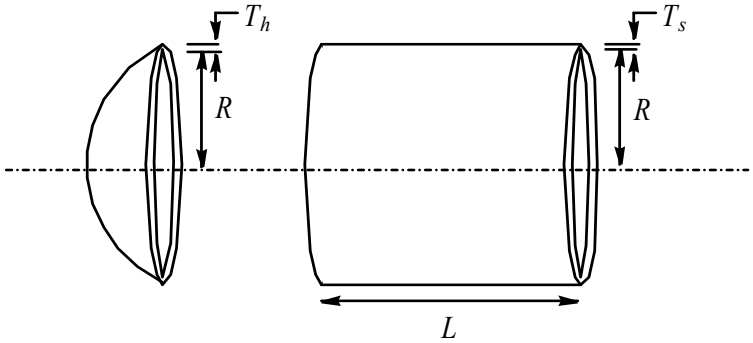

Figure 3: Pressure vessel design problem.

The problem can be formulated as follows (Kannan and Kramer, 1994):

$$\min f(x) = 0.6224 x_1 x_3 x_4 + 1.7781 x_2 x_3^2 + 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3 \tag{20}$$

subject to

$$g_1(x) = -x_1 + 0.0193x_3 \le 0 \tag{21}$$

$$g_2(x) = -x_2 + 0.00954x_3 \le 0 \tag{22}$$

$$g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0 \tag{23}$$

$$g_4(x) = x_4 - 240 \le 0 \tag{24}$$

In the literature, this problem has been solved by using an augmented Lagrangian multiplier approach (Kannan and Kramer, 1994), a genetic adaptive search (Deb, 1997), and a co-evolutionary particle swarm optimization (He and Wang, 2007).

In the present work, the following design space is adopted (He and Wang, 2007): 1 in $\le x_1 \le$ 99 in, 1 in $\le x_2 \le$ 99 in, 10 in $\le x_3 \le$ 200 in, 10 in $\le x_4 \le$ 200 in. The best solutions obtained by the above mentioned approaches are shown in Table 3. From Table 3, it can be seen that the best solution found by HS and SAHS is better than the best solutions found by other techniques (Kannan and Kramer, 1994; Deb, 1997), and exhibits the same quality as the one obtained by He and Wang (2007).

Table 3. Comparison of the best solutions for the pressure vessel design problem using different methods (N_{eval} is the number of objective function evaluations).

Design variables	Kannan and Kramer (1994)	Deb (1997)	He and Wang (2007)	HS (standard deviation)	SAHS (standard deviation)
x_1 (in)	1.125000	0.937500	0.812500	0.812500 (0.0059)	0.812500 (0.0008)
x_2 (in)	0.625000	0.500000	0.437500	0.437500 (0.0050)	0.437500 (0.0005)
x_3 (in)	58.29100	48.32900	42.09126	42.09127 (0.0026)	42.09127 (0.0006)
x_4 (in)	43.69000	112.6790	176.7465	176.7466 (0.0147)	176.7466 (0.0077)
g_1 (in)	0.000016	-0.004750	-0.000139	-0.000139	-0.000139
g_2 (in)	-0.068904	-0.038941	-0.035949	-0.035949	-0.035949
g_3 (in^3)	-21.22010	-3652.876	-116.3827	-116.3827	-116.3827
g_4 (in)	-196.3100	-127.3210	-63.25350	-63.25350	-63.25350
f (\$)	7198.0428	6410.3811	6061.0777	6061.0778 (0.0066)	6061.0778 (0.0089)
N_{eval}	Unavailable	200000	Unavailable	10020	7020

3.4. Binary distillation column design problem

Next case study is a binary distillation system from the MINOPT User's Guide (Schweiger *et al.*, 1997) and Bansal *et al.* (2003). The column has a fixed number of trays and the objective is to determine the optimal feed location (discrete decision), vapour boil-up, V, and reflux flow rate, R (continuous decisions), in order to minimize the integral square error (ISE) between the bottoms and distillate compositions and their respective set-points. The superstructure of the system is depicted in Fig. 4.

The following modeling assumptions were used by Schweiger *et al.* (1997): (i) constant molar overflow; (ii) constant relative volatility, a; (iii) phase equilibrium; (iv) constant liquid hold-ups, equal to m for each tray and 10m for the re-boiler and condenser; (v) no tray hydraulics; (vi) negligible vapour hold-ups; and (vii) no pressure drops. The system is initially at steady-state; at t=0 there is a step change in the feed composition, z_f; and the inequality constraints are that the distillate composition must be greater than 0.98, and the bottoms composition must be less than 0.02, at the end of the time horizon of 400 min. The problem can be stated mathematically as:

$$\min\ f(x) = ISE(t_f) \tag{25}$$

where

$$\frac{d(ISE)}{dt} = \left(x_b - x_b^*\right)^2 + \left(x_{N+1} - x_{N+1}^*\right)^2 \tag{26}$$

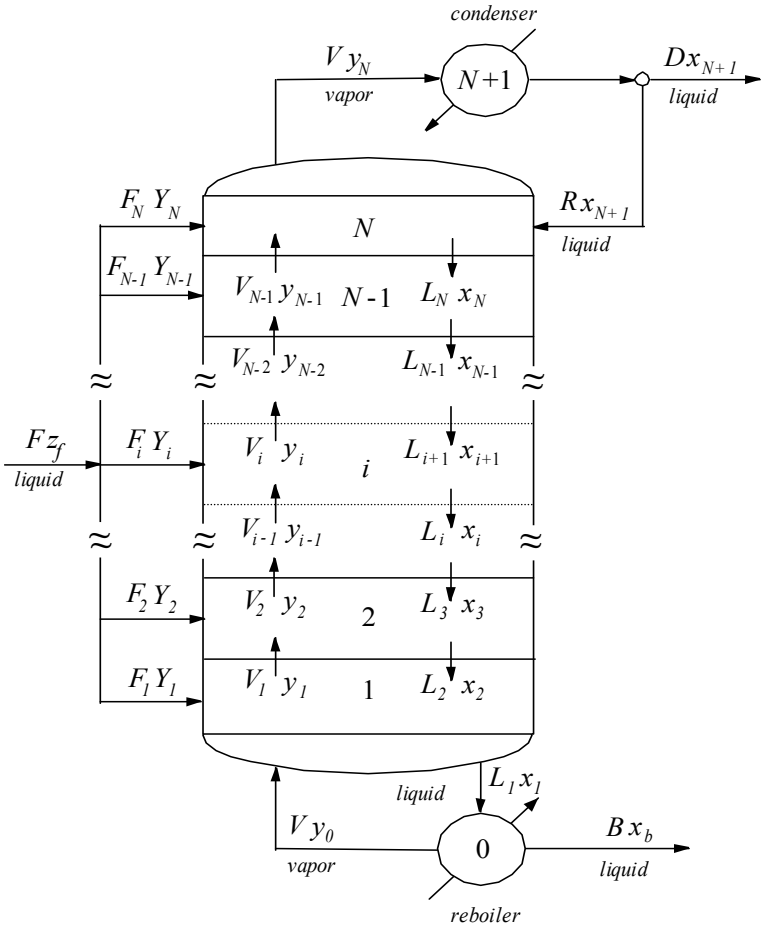

Figure 4: Binary distillation column.

subject to component balances - Eqs. (27)-(30), overall balances - Eq. (31)-(34), vapour-liquid equilibrium - Eqs. (35)-(36), and step disturbance – Eq. (37):

$$10m\frac{dx_{\mathrm{b}}}{dt} = L_{\mathrm{1}}x_{\mathrm{1}} - Vy_{\mathrm{o}} - Bx_{\mathrm{b}}, \quad \left.\frac{dx_{\mathrm{b}}}{dt}\right|_{t=0} = 0 \tag{27}$$

$$m\frac{dx_{\mathrm{i}}}{dt} = L_{\mathrm{i+1}}x_{\mathrm{i+1}} - L_{\mathrm{i}}x_{\mathrm{i}} + V\left(y_{\mathrm{i-1}} - y_{\mathrm{i}}\right) + Fyf_i z_f, \quad \left.\frac{dx_{\mathrm{i}}}{dt}\right|_{t=0} = 0, \quad \mathrm{i=1, ..., N\text{-}1} \tag{28}$$

$$m\frac{dx_{\mathrm{N}}}{dt} = -L_{\mathrm{N}}x_{\mathrm{N}} + V\left(y_{\mathrm{N\text{-}1}} - y_{\mathrm{N}}\right) + Fyf_N z_f + Rx_{\mathrm{N+1}}, \quad \left.\frac{dx_{\mathrm{N}}}{dt}\right|_{t=0} = 0 \tag{29}$$

$$10m\frac{dx_{\mathrm{N+1}}}{dt} = V\left(y_{\mathrm{N}} - x_{\mathrm{N+1}}\right), \quad \left.\frac{dx_{\mathrm{N+1}}}{dt}\right|_{t=0} = 0 \tag{30}$$

$$0 = L_{\mathrm{1}} - V - B \tag{31}$$

$$0 = L_{\mathrm{i+1}} - L_{\mathrm{i}} + F\,yf_{\mathrm{i}}, \quad \mathrm{i=1, ..., N\text{-}1} \tag{32}$$

$$0 = -L_{\mathrm{N}} + V + F\,yf + R \tag{33}$$

$$0 = V - D - R \tag{34}$$

$$y_o = \frac{\alpha x_b}{1+(\alpha - 1)x_b} \tag{35}$$

$$y_i = \frac{\alpha x_i}{1+(\alpha - 1)x_i}, \quad i=1, ..., N \tag{36}$$

$$z_f = 0.54 - 0.09\exp(-10t) \tag{37}$$

The following design space is adopted (Bansal *et al.*, 2003): 0.05 kmol/min $\leq V \leq$ 2 kmol/min, 0.25 kmol/min $\leq R \leq$ 1.3 kmol/min, $1 \leq yf \leq 30$. Model parameters (Bansal *et al.*, 2003): number of trays (N) equal to 30; relative volatility, α equal to 2.5; tray liquid hold-up equal to 0.175 kmol; feed flow rate (F) equal to 1 kmol/min; distillate set-point (x^*_{N+1}) equal to 0.98 and bottoms set-point (x^*_b) equal to 0.02.

Table 4 presents the results obtained by BFOA and by other competing techniques. In this table, it can be seen that the best solution found by HS and SAHS are similar to the results found by Bansal *et al.* (2003).

Table 4. Comparison of the best solutions for the binary distillation column design problem (N_{eval} is the number of objective function evaluations).

Design variables	Bansal *et al.* (2003)	HS (average)	SAHS (average)
Feed tray	25	25 (25)	25 (25)
V (kmol/min)	1.5426	1.5426 (0.0019)	1.5426 (0.0001)
R (kmol/min)	1.0024	1.0024 (0.0068)	1.0024 (0.0001)
f (-)	0.1817	0.18179 (0.0001)	0.18179 (0.0002)
N_{eval}	Unavailable	10020	6020

4. CONCLUSIONS

In the present contribution, the Self-Adaptive Harmony Search (SAHS) algorithm, inspired by the natural musical performance process that occurs when a musician searches for a better state of harmony and dynamic update of parameters, was applied to solve different design problems in engineering. The simulation results were compared with those obtained from other competing evolutionary algorithms. Besides, the results showed that the methodology is configured as a promising alternative for a number of engineering applications. In addition, it is important to observe that the SAHS yields better results, in terms of the number of objective function evaluations, as compared with the HS algorithm.

Consequently, considering the number of objective function evaluations, the present approach needs yet to be better studied, so that final conclusions can be drawn. This particular characteristic is inherent to the methodology used due to the large number of loops to be performed. Consequently, it is expected that a high number of objective function evaluations is necessary in the present version of the algorithm.

Further research work will be focused on the study of other mechanisms to update the parameters required by HS to improve the quality of the solution.

5. ACKNOWLEDGEMENTS

The authors acknowledge the financial support provided by FAPEMIG. Dr. Steffen also acknowledges the financial support provided by CNPq and FAPEMIG (INCT-EIE).

6. REFERENCES

Arora, J. S., 1989. “Introduction to Optimum Design”. McGraw-Hill, New York.

Bansal, V., Sakizlis, V., Ross, R., Perkins and J. D., Pistikopoulos, E. N., 2003, “New Algorithms for Mixed-integer Dynamic Optimization”, Computers and Chemical Engineering, 27, 647-668.

Belegundu, A. D., 1982. “A Study of Mathematical Programming Methods for Structural Optimization”. Department of Civil and Environmental Engineering, University of Iowa, Iowa City, Iowa.

Ceylan, H. and HaIdenbilen, S., 2008, “Transport Energy Modeling with Meta-heuristic Harmony Search Algorithm, An Application to Turkey, Energy Policy 36 (7), 2527-2535.

Coello, C. A. C., 2000. “Use of a Self-adaptive Penalty Approach for Engineering Optimization Problems”. Computers in Industry 41, 113–127.

Coelho, L. S. and Mariani, V. C., 2006, "Combining of Chaotic Differential Evolution and Quadratic Programming for Economic Dispatch Optimization with Valve-Point Effect", IEEE Transactions on Power Systems, 21, 2, 989-996.

Deb, K., 1991. "Optimal Design of a Welded Beam via Genetic Algorithms". AIAA Journal, 29 (11), 2013–2015.

Deb, K., 1997. "GeneAS: A Robust Optimal Design Technique for Mechanical Component Design". In: Dasgupta, D., Michalewicz, Z. (Eds.), Evolutionary Algorithms in Engineering Applications. Springer, Berlin, pp. 497–514.

Geem, Z. W., Kim, J. H. and Loganathan, G. V., 2001, "A New Heuristic Optimization Algorithm: Harmony Search", Simulations 76 (2001) 60–68.

Geem, Z. W., Kim, J. H. and Loganathan, G. V., 2002, "Harmony Search Optimization: Application to Pipe Network Design", Int. J. Model. Simul. 22 (2) (2002) 125–133.

Geem, Z. W., Lee, K. S. and Park, Y. J., 2005, "Application of Harmony Search to Vehicle Routing", Am. J. Appl. Sci. 2, 1552–1557.

Geem, Z. W., 2006, "Optimal Cost Design of Water Distribution Networks using Harmony Search", Eng. Optim. 38, 259–280.

Hasançebi, O, Erdal, F. and Saka, M. P., 2010, "Adaptive Harmony Search Method for Structural Optimization", Journal of Structural Engineering, vol. 136 (4), pp. 419-431.

He, Q. and Wang, L., 2007, "An Effective Co-evolutionary Particle Swarm Optimization for Constrained Engineering Design Problems", Engineering Applications of Artificial Intelligence, 20, 89-99.

Kannan, B. K. and Kramer, S. N., 1994. "An Augmented Lagrange Multiplier based Method for Mixed Integer Discrete Continuous Optimization and its Applications to Mechanical Design". Transactions of the ASME, Journal of Mechanical Design 116, 318–320.

Lee, K. S. and Geem, Z. W., 2005, A New Meta-heuristic Algorithm for Continuous Engineering Optimization, harmony search theory and practice, Comput. Methods Appl. Mech. Eng. 194, 3902–3933.

Lee, K. S., Geem, Z. W., Lee, S. H. and Bae, K. W. 2005, "The Harmony Search Heuristic Algorithm for Discrete Structural Optimization", Eng. Optim. 37, 663–684.

Mahdavi, M. Fesanghary, M. and Damangir, E., 2007, "An Improved Harmony Search Algorithm for Solving Optimization Problems", Appl. Math. Comput. 188,1567–1579.

Omran, M. G. H. and Mahdavi, M., 2008, "Global-best Harmony Search", Appl. Math. Comput. 198, 643–656.

Pan, Q-K., Suganthan, P. N., Tasgetiren, M. F. and Liang, J. J., "A Self-adaptive Global Best Harmony Search Algorithm for Continuous Optimization Problems", Applied Mathematics and Computation 216 (2010) 830–848.

Rao, S. S., 1996. "Engineering Optimization". Wiley, New York.

Sarvari, H. and Zamanifar, K., 2010. "A Self-Adaptive Harmony Search Algorithm for Engineering and Reliability Problems", International Conference on Computational Intelligence, Modelling and Simulation, pp. 59-64.

Schweiger, C. A., Rojnuckarin, A. and Floudas, C. A., 1997. "MINOPT: User's Guide". Princeton University. Software Version 2.0.

Vasebi, A., Fesanghary, M. and Bathaee, S. M. T., 2007, "Combined Heat and Power Economic Dispatch by Harmony Search Algorithm", Electr. Power Energy Syst. 29, 713–719.

Wang, C-M. and Huang, Y-F., 2010. "Self-Adaptive Harmony Search Algorithm for Optimization", Expert Systems with Applications, vol. 37 (4), pp. 2826-2837.