2019 16th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE) Mexico City, Mexico. September 11-13, 2019

A Harmony Search Variant Based on a Novel Synthesized Approach for Constrained Numerical Optimization

Mario Alejandro Barbosa-Mendez
Instituto Politécnico Nacional
México
mbarbosam1801@alumno.ipn.mx

Edgar Alfredo Portilla-Flores
Instituto Politécnico Nacional
México
aportilla@ipn.mx

Eduardo Vega-Alvarado
Instituto Politécnico Nacional
México
evega@ipn.mx

Maria Barbara Calva-Yáñez
Instituto Politécnico Nacional
Mexico
b_calva@hotmail.com,

Gabriel Sepúlveda-Cervantes
Instituto Politécnico Nacional
México
gsepulvedac@ipn.mx

***Abstract*—In this work, a novel variant of the Harmony Search algorithm called Synthesized Harmony Search is presented. It is based on the application of a synthesizer to the process of music composition simulated in the original Harmony Search. Additionally, constraints are handled by integrating the rules of Deb to the algorithm. The effect of the synthesizer over the performance is evaluated with a well-known benchmark of constrained numerical optimization problems. The proposed algorithm was compared to the original Harmony Search and one of its previously developed variants, HS_Mod2, with a statistical analysis using the best solution obtained per each algorithm in thirty independent executions. In general, the solutions generated by the Synthesized Harmony Search surpassed the corresponding values produced by the original algorithm, and were similar to those solutions from HS_Mod2, but presenting a steadier behavior.**

***Index Terms*—Harmony Search, metaheuristics, numerical optimization, synthesizer, constraints.**

I. Introduction

Optimization is the search for the best solution, under given circumstances. The mathematical formulation of a Constraint Numerical Optimization Problem (CNOP) is defined as the search of a value set $\overrightarrow{x}$ such that:

$$min/max\ f_t(\overrightarrow{x}) \tag{1}$$

subject to:

$$g_j(\overrightarrow{x}) \leq 0 \qquad j = 1, 2, ..., p \tag{2}$$

$$h_k(\overrightarrow{x}) = 0 \qquad k = 1, 2, ..., q \tag{3}$$

where $f_t(\overrightarrow{x})$ corresponds to one or more objective functions; Eqs. (2) and (3) are the inequality and equality constraints, respectively; and $\overrightarrow{x}$ is the solution or design vector. $\overrightarrow{x}$ is a subset of the feasible region Ω and each design variable is bounded by $[L_i, U_i]$ with $i = 1, 2, ...n$.

Diverse methods have been used for solving CNOPs, with metaheuristics highlighting among then. Metaheuristics are inspired in natural and artificial processes, such as the Genetic Algorithm (GA) [1], Ant Colony Optimization (ACO) [2], Artificial Bee Colony (ABC) [3], Firefly Algorithm (FA) [4], Differential Evolution (DE) [5] and Particle Swarm Optimization (PSO) [6]. Harmony Search (HS) is a metaheuristic based on the musical composition process [7], that simulates the improvisation while composing a musical production with the adjustment of sounds or notes in specific ranges. The goal is to find a harmonic sound combination, from a aesthetic point of view.

A complete revision of the state of the art for HS, its variants and hybridizations can be consulted in [8], [9]. Among the most important variants are: Improved HS (IHS) [10], that controls the r_{pa} and bw parameters in a deterministic way; Adjustable Bandwidth HS (ABHS) [11] with adaptive control of bw; Self-adaptive Global-best Harmony Search (SGHS) [12], that modifies r_{accept}, r_{pa} and bw; Exploratory HS (EHS) [13] where bw is adaptive according to the standard deviation of the current harmony; Improved HS (ImHS) [8], that controls bw with a deterministic criteria and includes a new solution generator r_{ia} based on the global best solution; and Local opposition-based learning self-adaptation global HS (LHS) [9], that integrates local opposition-based learning for increasing the solution diversity, an adaptive global pitch adjustment, and a competition-selection mechanism for enhancing the ability of avoiding local optima solutions.

Other HS-based developments include parameter control: in [14] a modified harmony search is proposed where r_{accept}, bw and r_{pa} have a deterministic control parameter; in [15], the Adaptive with Best-based search strategy HS (ABHS) is presented, partially exploiting the global-best solution to improve the search ability; and in [16], the Enhanced Self-adaptive Global-best HS (ESGHS) is developed, that takes advantage of the information hidden in the harmony memory to devise a search strategy, and also integrating a self-adaptive

978-1-7281-4840-3/19/$31.00 ©2019 IEEE

mechanism to develop a parameter-setting-free technique. Additionally, new HS hybridizations with diverse metaheuristics have been generated, as in [17], where an algorithm called HS-SA was developed, based on a hybridization of HS and Simulated Annealing (SA).

In this work, a new variant of HS is presented, the Synthesized Harmony Search (SHS), that modifies the best global solution for each design variable based on a probability, using a codified synthesizer. The *r*-th best solution is used by the synthesizer, in a combination with the magnitude generated by a specific harmonic of a geometric signal, and a filter for guiding or canceling the participation of each harmonic. A deterministic parameter control is also included to modify the effect of the synthesizer during the algorithm execution. It allows an exploration during the initial cycles based not only in the best global solution, to increase diversity, and reduces this effect to allow more exploitation in final cycles.

The performance of SHS was tested by solving a well-known benchmark from the related literature, and the results were compared to the solutions obtained with the original HS and the reported values for a variant called HS_Mod2 [8]. The paper is organized as follows: Section 2 explains the basic Harmony Search algorithm while Section 3 describes the proposed modifications to HS, conforming SHS; experimental results are reported and analyzed in Section 4, and Section 5 corresponds to the conclusions and future work.

II. Harmony Search

A combination of music notes emitted as a sequence generates a harmony. When musicians compose music they improvise new harmonies by considering three sources [18]:

1) A previously generated harmony, that is, a combination from the music memory.
2) An adjustment of the immediate previous harmony by changing some of its notes.
3) A randomly-generated set of new notes.

Harmonies are changed during the composition process in order to improve the sound; this adjustment is simulated by HS.

A. Harmony Search Algorithm

HS is applied to the solution of optimization problems by associating the design vector $X^J = (x_1^J, x_2^J, ..., x_n^J)$ to a series of harmonies known as harmony vectors, with $J = 1, 2, ..., K$.. The array of K harmonies is the Harmony Memory (HM), shown in Eq. (4):

$$\text{HM} = \begin{bmatrix} X^1 \\ X^2 \\ \vdots \\ X^K \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_n^1 \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^K & x_2^K & \cdots & x_n^K \end{bmatrix} \quad (4)$$

A new harmony $newX = (newX_1, newX_2, ..., newX_n)$ is generated considering two probabilities: $r_{accept} \in (0, 1)$ and $r_{pa} \in (0, 1)$. Each variable $newX_{(i)}$ in the vector is obtained as follows:

1) With a probability r_{accept} a value previously stored in HM is selected randomly.
2) With a probability $r_{accept} * r_{pa}$ Eq. (5) is applied, where $index$ is an integer value in the range $[1, K]$, and bw is the bandwidth for tone adjustment, defined by Eq. (6):

$$newX_{(i)} = \text{HM}_{(index,i)} + bw_{(i)} * rand[-1, 1] \quad (5)$$

$$bw_{(i)} = \frac{U_{(i)} - L_{(i)}}{1000} \quad (6)$$

3) With a probability $(1 - r_{accept})$ a new random value is produced by applying Eq. (7):

$$newX_{(i)} = rand[L_{(i)}, U_{(i)}] \quad (7)$$

The new harmony $newX$ is accepted if and only if it is better than the worst harmony X^{worst}, replacing the worst and updating HM. This process is repeated until a stop condition is reached, given by the maximum number of iterations (*IterMax*) or evaluations of the objective function. They are equivalent, since HS generates only one new harmony per iteration. Algorithm 1 shows the pseudocode of HS.

Algorithm 1: Harmony Search

```
define objective function f(x); x = (x_1; x_2; ...; x_n)
define harmony memory size (K), accepting rate
  r_accept and pitch adjusting rate r_pa
generate randomly first harmony memory HM
  between [U_(i) − L_(i)]
calculate bw_(i) = (U_(i) − L_(i))/1000
while (iteration ≤ IterMax) do
    for i=1:n do
        if rand(0, 1)< r_accept then
            index = random integer [1, K]
            if rand(0, 1)< r_pa then
                newX_(i) =
                  HM_(index,i) + bw_(i) * rand[−1, 1]
            else
                newX_(i) =HM_(index,i)
            end
        else
            newX_(i) = rand[L_(i), U_(i)]
        end
    end
    iteration = iteration + 1
    accept newX as new solution, if better than
      X^worst, applying rules of Deb
end
return best harmony found
```

The proposed algorithm is intended to solve optimization problems with equality and inequality constraints as represented by Eqs. (2) and (3), respectively. These expressions define the feasible region, in combination with the boundaries. For this reason, a constraint handler is required; in this case, the feasibility rules of Deb were selected [19]. These rules are as follows:

1) Between two feasible solutions, the one with the best objective function is selected.
2) Between a feasible and an unfeasible solution, the feasible one is selected.
3) Between two unfeasible solutions, the one with the lower Constraint Violation Sum (CVS) is selected.

Inequality constraints are converted into equality constraints by applying Eq. (8), using a previously established tolerance with a value ε close to zero, known as slack [20], to diminish the computational complexity of the problem.

$$|h_k(\overrightarrow{x})| - \varepsilon \leq 0 \tag{8}$$

III. Synthesized Harmony Search

A music note is composed by harmonics, and the Fourier transform can be applied to obtain the frequency and energy supplied by each harmonic. Simple geometric waves are regularly used for sound synthesis, since they contain a rich content of harmonics. Harmonics can be filtered to produce a variety of sounds. Among the most used geometric waves for synthesizing are:

a) Square: Contains only odd harmonics with amplitude $A_r = A_1/r$.

b) Sawtooth: It is useful for the synthesis since contains all the harmonics in the geometric relation $A_r = A_1/r$.

c) Triangular: Contains only odd harmonics with amplitude $A_r = A_1/r^2$.

Music composition has evolved with the support of the analog and digital electronics. New notes can be generated by using synthesizers, that combine harmonics in diverse proportions. A synthesizer accelerates the composition process by varying the notes according to the experience of the composer. This knowledge can be implemented in HS, modifying the notes with the harmonics and filters generated by a codified synthesizer, This represents a novel variant that follows the line of inspiration of the original algorithm, the musical composition, and to the best knowledge of the authors there is not other development in the related literature that applies a similar approach.

A. Codified synthesizer

The synthesizer tries to modify the best note using the following best notes (second best, third best, etc.). The magnitude of the modification depends on the type of the applied geometric wave, but excluding those harmonics whose magnitude is zero. For instance, in a square wave the modification because of the second best note is given by $A_2 = 1/3$, with $A_3 = 1/5$ for the third best note, and so on, until considering the r notes better or equal to K, where K is the number of harmonies. Fig. 1 shows the harmonic composition of a square wave. Additionally, a filter is applied to each note, in order to randomly guide or cancel its effect ($filter \in [-1, 0, 1]$). The number of notes to consider is a criteria given by the designer. The complete process is reflected in Eq. (9).

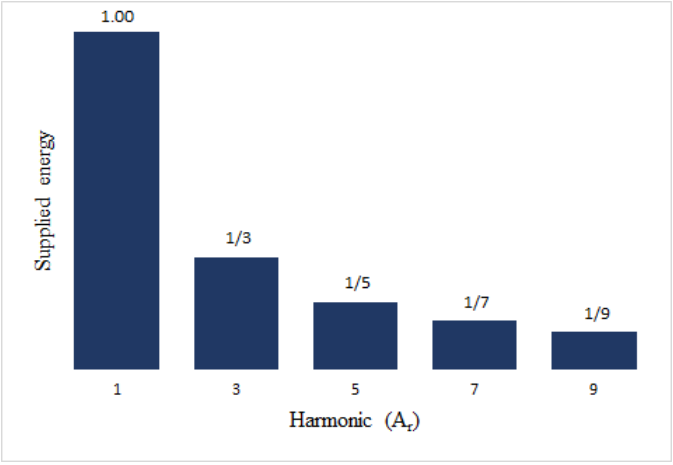

Fig. 1: Harmonic composition of a square wave

$$\begin{aligned} Syn = \quad & (filter)(A_2)\mathrm{HM}_{(2nd-best,i)} + \\ & (filter)(A_3)\mathrm{HM}_{(3nd-best,i)} + \ldots + \\ & (filter)(A_r)\mathrm{HM}_{(rth-best,i)} \\ \text{with } & 2 \leq r \leq K \end{aligned} \tag{9}$$

B. Rate of synthesizer adjustment

This work presents the development of a HS variant, Synthesized Harmony Search (SHS). The functionality of a codified synthesizer was integrated to the algorithm, in order to improve its performance by including the knowledge of r best solutions to upgrade the global best solution. The rate of synthesizer adjustment, defined as r_s, represents the probability of an adjustment by the synthesizer to the global best note. So, the total probability of this adjustment is given by $(r_{accept})(1-r_{pa})(r_s)$. An additional parameter was integrated to SHS to enhance the exploration and the exploitation in the initial and final iterations, respectively. It results from dividing a real value in $[0, 1]$ by the number of the current iteration.

Algorithm 2 presents the pseudocode of SHS. The instructions in lines 13-17 correspond to the application of the synthesizer adjustment, validated with the probability r_s. If the condition is fulfilled, the proposed new note is obtained in line 14 by the synthesizer, based on both Eq. (9) and the corresponding note in the best global solution.

IV. Experimentation and Results

A well-known benchmark with twenty four CNOPs from the related literature [21] was selected to evaluate the performance of the proposed algorithm, comparing the efficiency of SHS in relation to the original HS and a reported variant called Hs_Mod2 [8]. This variant was selected since both the proposed algorithm and HS_Mod2 are based on using the best global solution and a rate of application. A summary with the main features of the selected CNOPs is shown in Table I, where P indicates the number of the problem, n is the number of variables, LI is the number of linear inequality constraints,

TABLE I: main features of the selected benchmark problems

P	n	Type of Funtion	ρ (%)	LI	NI	LE	NE	a
g01	13	quadratic	0.011	9	0	0	0	6
g02	20	nonlinear	99.997	0	2	0	0	1
g03	10	polynomial	0.000	0	0	0	1	1
g04	5	quadratic	52.123	0	6	0	0	2
g05	4	cubic	0.000	2	0	0	3	3
g06	2	cubic	0.006	0	2	0	0	2
g07	10	quadratic	0.000	3	5	0	0	6
g08	2	nonlinear	0.856	0	2	0	0	0
g09	7	polynomial	0.512	0	4	0	0	2
g10	8	linear	0.001	3	3	0	0	6
g11	2	quadratic	0.000	0	0	0	1	1
g12	3	quadratic	4.771	0	1	0	0	0
g13	5	nonlinear	0.000	0	0	0	3	3
g14	10	nonlinear	0.000	0	0	3	0	3
g15	3	quadratic	0.000	0	0	1	1	2
g16	5	nonlinear	0.020	4	34	0	0	4
g17	6	nonlinear	0.000	0	0	0	4	4
g18	9	quadratic	0.000	0	13	0	0	6
g19	15	nonlinear	33.476	0	5	0	0	0
g20	24	linear	0.000	0	6	2	12	16
g21	7	linear	0.000	0	1	0	5	6
g22	22	linear	0.000	0	1	8	11	19
g23	9	linear	0.000	0	2	3	1	6
g24	2	linear	79.655	0	2	0	0	2

NI is the number of nonlinear inequality constrains, LE is the number of linear equality constraints, NE is the number of nonlinear inequality constraints, a is the number of active constraints, and ρ is a metric to estimate the complexity of the problem.

HS and SHS were programmed in MATLAB® $R2017b$, in a platform with a Windows 8.1 operating system, Intel® Core™ i7-4510u @2.00 GHz processor, and 16 GB RAM. The results obtained for SHS are compared to the values reported for HS_Mod2 [8]. Thirty independent runs were executed for each of the benchmark problems to get a representative sample for a statistical analysis.

Table II shows the parameter tuning applied for the solution of the benchmark. The three algorithms were tuned with the same parameters. The values of A_r are referenced to a triangular wave, including only the harmonics 3, 5, and 7, corresponding to the values 2, 3, and 4 for r, respectively. The configuration was set considering error-and-trial techniques and previously reported developments. The tolerance for Eq. (8) is $\varepsilon = 1\text{x}10^{-4}$.

TABLE II: Parameters used in the implementation

Parameter	Source	Value
K	HS	10
IterMax		100,000
r_{accept}		0.99
r_{pa}		0.3
bw		$(U_{(i)} - L_{(i)})/1000$
r_s	SHS	0.7
r		$\{2, 3, 4\}$
A_r		$\{1/9, 1/25, 1/49\}$

Algorithm 2: Synthesized Harmony Search

define objective function $f(x); x = (x_1; x_2; ...; x_n)$
define harmony memory size (K), accepting rate r_{accept} and pitch adjusting rate r_{pa}
define synthesizer adjusting rate r_s
generate randomly first harmony Memory HM between $(U_i - L_i)$
calculate $bw_{(i)} = (U_i - L_i)/1000$
while *(iteration* $\leq$ *IterMax)* **do**
  for *i=1:n* **do**
    if *rand(0, 1)*$< r_{accept}$ **then**
      $index =$ random integer $[1, K]$
      if *rand(0, 1)*$< r_{pa}$ **then**
        $newX_{(i)} =$ HM$_{(index,i)} + bw_{(i)} * rand[-1, 1]$
      else
        if *rand(0, 1)*$< r_s$ **then**
          $newX_{(i)} =$ HM$_{(best,i)} + \frac{rand[0,1]}{iteration} * Syn$
        else
          $newX_{(i)} =$HM$_{(index,i)}$
        end
      end
    else
      $newX_{(i)} = rand[L_{(i)}, U_{(i)}]$
    end
  end
  $iteration = iteration + 1$
  accept $newX$ as new solution, if better than X^{worst}, applying rules of Deb
end
return best harmony found

A. Analysis of results

The obtained results for HS and SHS, and the reported results for HS_Mod2 are shown in Table III, that includes the best and worst solutions, and the Standard Deviation. The highlighted values correspond to the best global solution found for each problem.

SHS vs HS: The results of the paired comparison between SHS and HS indicate that SHS produced better solutions for problems g01, g03, g04, g05, g06, g07, g09, g10, g13, g14, g15, g18, and g19; SHS and HS reached the same best solution for g02, g08, g11, g12 and g24, and neither algorithm produced a feasible solution for cases g20 and g22. Finally, HS got better solutions for g16, g17, g21, and g23.

SHS vs HS_Mod2: SHS generated better solutions for problems g06, g10, g13, g14, g15, and g18; SHS and HS_mod2 reached the same best solution for g01, g03, g04, g08, g11, g12, and g24, and neither algorithm produced a feasible solution for cases g20 and g22. Finally, HS_Mod2 got better solutions for g02, g05, g07, g09, g16, g17, g19, g21 and g23.

It is important to highlight that for g15 SHS obtained a better solution than the reported in the original document of the benchmark, with a new value for the objective function of $f(x^*) = 961.403214796326$. According to the main features presented in Table I, SHS has a better behavior when solving CNOPs with active constrains and low values of ρ, representing very complex problems.

TABLE III: Benchmark results

Problem		Algorithm		
		HS	**HS_Mod2**	**SHS**
g01 -15.0	Best	-14,9990	**-14.9998**	**-14,9998**
	Worst	-12,999	-14.998	-14,998
	SD	3,65E-01	2,39E-04	2,97E-04
g02 -0.803619014	Best	-0,7979	**-0.8035**	-0,7979
	Worst	-0,7571	-0.7852	-0,5064
	SD	9,39E-03	6.80E-03	5,83E-02
g03 -1.0005001	Best	-0,9999	**-1,0002**	**-1,0002**
	Worst	-0,9969	0.9999	-0,4438
	SD	6,60E-04	2.07E-04	1,41E-01
g04 -30665.53867	Best	-30665,4880	**-30665.5104**	**-30665.5104**
	Worst	-30454,9570	-30665.302	-30114,0378
	SD	6,56E+01	5.88E+02	1,23E+02
g05 5126.496714	Best	5131,2660	**5126.747**	5129,2207
	Worst	5703,4640	5535.400	5875,8716
	SD	1,77E+02	1.17E-02	2,05E+02
g06 -6961.813	Best	-6961,1100	-6960.913	**-6961,8138**
	Worst	-6958,9890	-6959.991	-6961,8121
	SD	4,87E-01	2.65E-01	4,39E-04
g07 24.306209	Best	24.6291	**24.401**	24,5930
	Worst	25,5610	25.225	37,5137
	SD	2,40E-01	2.34E-01	3,63E+00
g08 -0.095825041	Best	**-0,0958**	**-0,0958**	**-0,0958**
	Worst	-0,0958	-0,0958	-0,0258
	SD	0,00E+00	0,00E+00	3,43E-02
g09 680.630057	Best	680,7717	**680.6620**	680.6690
	Worst	681,3250	680.892	684,6690
	SD	1,47E-01	4.24E-02	7,52E-01
g10 7049.24802	Best	7103,6740	7076.232	**7058.4884**
	Worst	11126,8770	11369.521	19082,5212
	SD	1,31E+03	1.40E+03	3,18E+03
g11 0.7499	Best	**0,7499**	**0,7499**	**0,7499**
	Worst	0,7499	0,7499	0,9916
	SD	0,00E+00	0,00E+00	6,79E-02
g12 -1	Best	**-1,0000**	**-1,0000**	**-1,0000**
	Worst	-1,0000	-1,0000	-1,0000
	SD	0,00E+00	0,00E+00	0,00E+00
g13 0.053941514	Best	0.0696	0.1095	**0,0670**
	Worst	0,9999	0.9378	18,4392
	SD	2,62E-01	2.53E-01	3,37E+00
g14 -47.764888459	Best	-9,6430	-22.914	**-45,7330**
	Worst	-0,0010	-0.115	-38,8800
	SD	3,41E+00	5.95E+00	1,74E+00
g15 961.71502229	Best	961,7150	961.771	**961,4032**
	Worst	965,6130	964.790	974,4050
	SD	1,18E+00	9.63E+01	4,23E+00
g16 -1.90515528	Best	-1,6540	**-1.731**	-1.6288
	Worst	-1,4740	-1.505	-0.5119
	SD	4,41E-02	6.53E-02	2.83E-01
g17 8853.539674	Best	8890,5230	**8874.375**	8962,0730
	Worst	9302,8990	9216.352	9002,0517
	SD	1,33E+02	9.36E+01	1,73E+01
g18 -0.866025403	Best	-0,8630	-0.859	**-0,8653**
	Worst	-0,4990	-0.664	-0,4999
	SD	1,24E-01	9.12E-02	1,56E-01
g19 32.65559295	Best	39,5640	**34.592**	38.0845
	Worst	95,0510	71.399	93.8135
	SD	1,46E+01	1.16E+01	1.25E+01
g20 0.2049794	Best	-	-	-
	Worst	-	-	-
	SD	-	-	-
g21 193.72451007	Best	217,2300	**204.327**	-
	Worst	399,3700	366.931	-
	SD	1,21E+03	4.31E+01	-
g22 263.430975504	Best	-	-	-
	Worst	-	-	-
	SD	-	-	-
g23 -400.05509999	Best	-393,5200	**-397.389**	44,3357
	Worst	-97,2010	-105.827	340,3867
	SD	9,16E+01	7.73E+01	1,55E+02
g24 -5.5080132716	Best	**-5,5080**	**-5.5080**	**-5,5080**
	Worst	-4,0530	-5.507	-5,5080
	SD	2,66E-01	1.78E-05	2,69E-08

V. Conclusions

A novel variant for the Harmony Search Algorithm called Synthesized Harmony Search (SHS), is presented in this work. It is based on a codified synthesizer for note modification over the best global solution, and was tested by solving a well-known benchmark. SHS presented a better performance than HS, not only because of the quality of its solutions but for its steady behavior with a considerably lower dispersion. In relation to HS_Mod2, SHS presented a similar behaviour.

As future work, it is proposed the application of this synthesizer technique simultaneously with other variants of HS, in order to verify if the combined effect of the improvements can produce a synergy. Additionally, it is required to test different synthesizing techniques. For its conditions, SHS is a useful tool for solving real-world engineering optimization problems. So, another future work is to test SHS applied to the solution of this kind of problems.

Acknowledgment

The authors thank the Instituto Politécnico Nacional of México for its support through the COFAA, and the SIP by the projects SIP20195772 and SIP20196288. The first author thanks CONACyT of México for its support through a scholarship for Graduate Studies.

REFERENCES

[1] J. H. Holland *et al.*, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[2] A. Colorni, M. Dorigo, V. Maniezzo *et al.*, "Distributed optimization by ant colonies," in *Proceedings of the first European conference on artificial life*, vol. 142. Cambridge, MA, 1992, pp. 134–142.

[3] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical report-tr06, Erciyes university, engineering faculty, computer ?, Tech. Rep., 2005.

[4] X.-S. Yang, *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[5] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[6] J. Kennedy, "Particle swarm optimization," *Encyclopedia of machine learning*, pp. 760–766, 2010.

[7] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[8] E. A. Portilla-Flores, Á. Sánchez-Márquez, L. Flores-Pulido, E. Vega-Alvarado, M. B. C. Yáñez, J. A. Aponte-Rodríguez, and P. A. Niño-Suárez, "Enhancing the harmony search algorithm performance on constrained numerical optimization," *IEEE Access*, vol. 5, pp. 25 759–25 780, 2017.

[9] H.-b. Ouyang, L.-q. Gao, S. Li, X.-y. Kong, Q. Wang, and D.-x. Zou, "Improved harmony search algorithm: Lhs," *Applied Soft Computing*, vol. 53, pp. 133–167, 2017.

[10] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied mathematics and computation*, vol. 188, no. 2, pp. 1567–1579, 2007.

[11] J. Contreras, I. Amaya, and R. Correa, "An improved variant of the conventional harmony search algorithm," *Applied Mathematics and Computation*, vol. 227, pp. 821–830, 2014.

[12] Q.-K. Pan, P. N. Suganthan, M. F. Tasgetiren, and J. J. Liang, "A self-adaptive global best harmony search algorithm for continuous optimization problems," *Applied Mathematics and Computation*, vol. 216, no. 3, pp. 830–848, 2010.

[13] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, and B. K. Panigrahi, "Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 1, pp. 89–106, 2010.

[14] E. E. Elattar, "Modified harmony search algorithm for combined economic emission dispatch of microgrid incorporating renewable sources," *Energy*, vol. 159, pp. 496–507, 2018.

[15] Z. Guo, H. Yang, S. Wang, C. Zhou, and X. Liu, "Adaptive harmony search with best-based search strategy," *Soft Computing*, vol. 22, no. 4, pp. 1335–1349, 2018.

[16] K. Luo, J. Ma, and Q. Zhao, "Enhanced self-adaptive global-best harmony search without any extra statistic and external archive," *Information Sciences*, vol. 482, pp. 228–247, 2019.

[17] A. Assad and K. Deep, "A hybrid harmony search and simulated annealing algorithm for continuous optimization," *Information Sciences*, vol. 450, pp. 246–266, 2018.

[18] X.-S. Yang, "Harmony search as a metaheuristic algorithm," in *Music-inspired harmony search algorithm*. Springer, 2009, pp. 1–14.

[19] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, vol. 186, no. 2-4, pp. 311–338, 2000.

[20] E. Mezura-Montes and O. Cetina-Domínguez, "Empirical analysis of a modified artificial bee colony for constrained numerical optimization," *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 10 943–10 973, 2012.

[21] J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. C. Coello, and K. Deb, "Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization," *Nanyang Technol. Univ.*, 2006.