Research Article

Iyad Abu Doush* and Eugene Santos

Best Polynomial Harmony Search with Best β-Hill Climbing Algorithm

https://doi.org/10.1515/jisys-2019-0101
Received Apr 19, 2019; accepted Oct 08, 2019

Abstract: Harmony Search Algorithm (HSA) is an evolutionary algorithm which mimics the process of music improvisation to obtain a nice harmony. The algorithm has been successfully applied to solve optimization problems in different domains. A significant shortcoming of the algorithm is inadequate exploitation when trying to solve complex problems. The algorithm relies on three operators for performing improvisation: memory consideration, pitch adjustment, and random consideration. In order to improve algorithm efficiency, we use roulette wheel and tournament selection in memory consideration, replace the pitch adjustment and random consideration with a modified polynomial mutation, and enhance the obtained new harmony with a modified β-hill climbing algorithm. Such modification can help to maintain the diversity and enhance the convergence speed of the modified HS algorithm. β-hill climbing is a recently introduced local search algorithm that is able to effectively solve different optimization problems. β-hill climbing is utilized in the modified HS algorithm as a local search technique to improve the generated solution by HS. Two algorithms are proposed: the first one is called PHSβ–HC and the second one is called Imp. PHSβ–HC. The two algorithms are evaluated using 13 global optimization classical benchmark function with various ranges and complexities. The proposed algorithms are compared against five other HSA using the same test functions. Using Friedman test, the two proposed algorithms ranked 2nd (Imp. PHSβ–HC) and 3rd (PHSβ–HC). Furthermore, the two proposed algorithms are compared against four versions of particle swarm optimization (PSO). The results show that the proposed PHSβ–HC algorithm generates the best results for three test functions. In addition, the proposed Imp. PHSβ–HC algorithm is able to overcome the other algorithms for two test functions. Finally, the two proposed algorithms are compared with four variations of differential evolution (DE). The proposed PHSβ–HC algorithm produces the best results for three test functions, and the proposed Imp. PHSβ–HC algorithm outperforms the other algorithms for two test functions. In a nutshell, the two modified HSA are considered as an efficient extension to HSA which can be used to solve several optimization applications in the future.

Keywords: Harmony Search Algorithm; Evolutionary Algorithms; Hill Climbing; Polynomial Mutation; β-Hill Climbing

2010 Mathematics Subject Classification: 68W50; 68T20; 90C59

1 Introduction

As our society continues to advance, we face more complex problems in areas of science, engineering, economic and business. Proposing efficient techniques for resolving optimization problems is important for the aforementioned fields [57]. Metaheuristic algorithms are a class of approximation methods which cannot

***Corresponding Author: Iyad Abu Doush:** Computer Science Department, Yarmouk University, Irbid, Jordan; Department of Computer Science Department and Information Systems, American University of Kuwait, Salmiya, Kuwait; Email: idoush@auk.edu.kw
Eugene Santos: Thayer Engineering School, Dartmouth College, 8000 Cummings Hall. Hanover, NH 03755

pledge to produce the global optima, but they can still help solve complex problems in a moderate amount of time. A metaheuristic-based algorithm tries to iteratively enhance the solution by exploring the problem search space. Such navigation is guided by an awareness about the problem with the anticipation of reaching a global optimum [20, 64].

Mainly, a metaheuristic is divided into *local search-based algorithms* and *population-based algorithms*. Population-based techniques firstly produce a group of randomly generated solutions. After that, the method combines the features from different solutions with the hope of generating better solutions [6, 11]. The population-based algorithms can concurrently look over large segments of the search space. Nevertheless, such algorithms may be unable to discover the local optima in each segment. As a result, the algorithm may not attain the optimal solution. Several population-based algorithms have been proposed and used to solve real-world problems such as genetic algorithm, bee colony optimization, and particle swarm optimization [45, 52, 62, 69, 70, 82].

Local search-based techniques initially begin with an individual solution which is repetitively modified using a neighborhood method. The algorithm stops when it reaches a local optimal solution. The search space can be decomposed into a group of smaller areas. Local search-based techniques work differently than population-based techniques, as they examine the search area intensely to obtain the local optima. Nevertheless, these techniques do not broadly investigate the search space.

Hill Climbing (HC) is considered the simplest local-search based technique. It is commonly used to boost the ability of population-based techniques in discovering the local optima in the currently searched segment [12]. Several techniques are introduced as an improvement of the HC to diminish the problem of being trapped in a local optima [20]. These techniques include Simulated Annealing (SA) [48], Tabu Search (TS) [37], Greedy Randomized Adaptive Search Procedure (GRASP) [29], Variable Neighborhood Search (VNS) [42], and Iterated Local Search (ILS) [50].

β-hill climbing is a recently introduced local search algorithm [7]. The algorithm has been used successfully to solve different optimization problems such as economic dispatch [8], text clustering [4], signal denoising [17, 18], multiple-reservoir scheduling [16], and feature selection [3]. Two operators are used in β-hill climbing to iteratively improve the solution which are neighborhood navigation ($\mathcal{N}$-operator) and β-operator. The $\mathcal{N}$-operator is used to reach randomly into a neighboring value. Then, some of the present solution values are selected or we choose a random value within the range using the β-operator. The β parameter is used to indicate the chances of using this operator.

Harmony Search Algorithm (HSA) [33] is a population-based algorithm which imitates the improvisation process of musicians. HSA has been applied to solving different optimization problems such as patient admission scheduling [24], face recognition [61], and many others [1, 5, 27, 31, 47, 67, 68, 76]. HSA begins by assigning values to harmonies stored in Harmony Memory (HM) within the range of the decision variables. After that, three operators are used to update the HM with a newly generated harmony. These operators are memory consideration, pitch adjustment, and random consideration. The memory consideration picks the value for the decision variable from the current values in HM. The pitch adjustment changes a current value in HM hoping to locally improve the solution. The random consideration selects a value randomly within the range of the decision variables to expand the solutions. Once the new harmony is created, it is compared to the worst harmony in the HM and replaces it if it is better. These steps are re-iterated until a stop criterion is met.

The structure and how the operators work in the HSA is altered over time to improve the algorithm performance when solving different problems [15, 26, 32, 71]. This is done in different ways such as tuning HSA selection in memory consideration [2, 10, 25, 54], improving the neighborhood in pitch adjustment [22, 43], self adaptive choosing of the algorithm parameters [13, 36, 56], using a different harmony memory structure [9, 55], and hybridizing the algorithm with other optimization methods [60, 74, 77].

Recently, several researchers propose variations of HSA to improve its performance. Multi-population-based harmony search algorithm that saves the best solutions in an external archive was developed [72]. In order to improve the diversity of the population, it is divided into sub-populations. Each sub-population is used to introduce a new harmony which replaces the worst solution. A modified harmony search with intersect mutation operator was introduced by Yi et al. [79]. The harmony memory is divided into best and worst

portions which are then used to develop the new harmony. Adaptive harmony search with best-based search strategy is proposed by [40] to improve algorithm search efficiency. The algorithm uses the best solutions in the pitch adjustment stage to generate the new solutions. In addition, the HMCR and PAR are tuned automatically during the search process based on the newly generated solutions. Another enhancement was introduced by Wang et al. [75] with an improved differential harmony search algorithm. The algorithm uses two differential evolution mutation operators to improve the algorithm exploitation. In addition, the pitch adjustment step uses the best solution to generate the new solution. Lastly, selective refining harmony search is introduced by Shabani et al. [63]. This algorithm takes the decision variable into consideration when developing the new harmony by changing only the variables with unwanted values from harmonies.

A significant shortcoming of the HSA is the inadequate exploitation when trying to solve complex problems [51]. Additionally, the algorithm needs setting the algorithm operators experimentally to find the suitable values that can improve the algorithm exploration and exploitation [34]. In order to improve the search ability of HSA this paper introduces a new HSA called best polynomial harmony search algorithm with best β–hill climbing with two variations. The first one is called PHSβ–HC and the second one is called Imp. PHSβ–HC. The proposed algorithm relies on replacing the random selection in memory consideration with roulette wheel and tournament selection as they provided better performance [10]. In addition, the original pitch adjustment and random consideration are replaced with a modified version of highly disruptive polynomial mutation presented in Hasan et al. [43]. The obtained new solution is then iteratively improved using a modified version of β-hill climbing as a local search. Such modification can help to maintain the diversity and enhance the convergence speed of the modified HS algorithm. The algorithm is evaluated using standard benchmark functions. The results demonstrate that the proposed algorithm outperforms other versions of HSA. In addition, the new algorithm exceeds the performance of other evolutionary algorithms for some of the benchmark functions.

The remainder of this paper is organized as follows: Section 2 gives an introduction on HSA. A description of the proposed method is presented in Section 3. Analysis and discussion of the algorithm evaluation are found in Section 4. Lastly, the paper concludes with possible future directions in Section 5.

2 The harmony search algorithm

Harmony Search algorithm (HSA) imitates how musicians improve their music iteratively [35]. During this process, each musician plays on an instrument to achieve the best harmony. In HSA, each decision variable represents a musician and the algorithm generates a value in order to find a global optimal solution.

The HSA consists of the following main steps [35]:

Step 1. Initialization of the problem and algorithm parameters:
The optimization problem can be modeled as a function f to be minimized or maximized as follows:

$f(\vec{x}), x_i \in X_i, i = 1, \ldots, N.$

$X_i \in [L_{x_i}, U_{x_i}]$ where L_{x_i} and U_{x_i} are the lower and upper bounds for the decision variable x_i.

Note that N is the decision variables count; $\vec{x}$ represents the solution vector which consists of decision variables x_i; and, X_i symbolizes each decision variable x_i possible values. Furthermore, during this step the following HSA parameters are specified: Harmony Memory Size (HMS) (or the population size); Harmony Memory Considering Rate (HMCR) which indicates if the decision variable value will be picked from the Harmony Memory (HM); Pitch Adjusting Rate (PAR) which determines if decision variable will be modified to a neighboring value; distance bandwidth (BW) specifies how far is the change in the pitch adjustment operator; and, the stop condition (i.e., Number of Improvisations (NI)).

Step 2. Initialization of the harmony memory:
In step 2, the solution variables (x_i) are initialized randomly with feasible values picked from the interval between the upper and lower bounds $[L_{x_i}, U_{x_i}]$. The HM is filled as follows:

$x_i^j = LB_i + (UB_i - LB_i) \times U(0, 1)$,

$\forall i = 1, 2, \ldots, N$ and $\forall j = 1, 2, \ldots,$ HMS, and $U(0, 1)$ generate a uniform random number between 0 and 1.

Step 3. Improvisation of a new harmony :
A new harmony vector $\vec{x'} = (x'_1, x'_2, x'_3, \ldots, x'_N)$ is improvised using three rules: (1) memory consideration, (2) random selection, and (3) pitch adjustment.

1. ***Memory consideration***. The memory consideration considers the value of the decision variable x'_i by picking randomly the relevant value from the HM $\{x_i^1, x_i^2, \ldots, x_i^{HMS}\}$. The selection of the values from HM will be on a frequent rate of HMCR.
2. ***Pitch adjustment***. In the memory consideration, a neighboring rule is applied based on the probability of PAR where PAR $\in (0, 1)$. The picked value in memory consideration for the decision variable x'_i is adjusted to its neighboring value as follows: $x'_i = x'_i \pm U(0, 1) \times \text{BW}$.
3. ***Random consideration***. The decision variables that do not take values from the memory consideration will pick a feasible value randomly with a probability of (1-HMCR).

Step 4. Harmony memory update:
In this step, the newly generated harmony vector in step 3 is evaluated to decide if it will be included in HM. The objective function value for the vector $\vec{x'} = (x'_1, x'_2, x'_3, \ldots, x'_N)$ is compared against the worst harmony in the HM,. Then, it will replace the existing worst harmony in HM if it is better.

Step 5. Repeat until the stopping condition is met:
The algorithm is terminated if it reaches the maximum number of improvisation (NI). Otherwise, steps 3 and 4 are repeated.

3 Proposed Method: best polynomial harmony search algorithm with best β-hill climbing

3.1 Best β-hill climbing

In this section, the β-hill climbing optimizer [7] and its modified version are explained. β-hill climbing starts with a set of solutions randomly produced $\mathbf{X} = (X_1, \ldots, X_n)$. Note that n is the number of decision variables where its selected value is feasible (i.e., within the variable range). The generated initial solution will be iteratively modified in the hope of obtaining the optimal solution. The β-hill climbing optimizer relies on using two operators to undergo updating the solution which are: $\mathcal{N}$-operator and β-operator.

The $\mathcal{N}$-operator is used for exploitation in the algorithm by applying a small change in the current solution. The operator is used to update the variable X_i, $i \in [1, 2, \ldots, n]$ for a randomly picked value from the solution as follows:

$$X'_i = X_i \mp U(0, 1) \times \mathcal{N} \qquad \exists i \in [1, n].$$

On the other hand, the β-operator is responsible for exploration by applying a uniform mutation to the current solution. The decision variable is selected to be changed using the β parameter as follows:

$$X_i'' \leftarrow \begin{cases} X_k & r \le \beta \\ X_i' & otherwise. \end{cases}$$

Note that β is how often the uniform mutation is applied. The r operator is a random number between 0 and 1. The pseudo code of the β-hill climbing is presented in algorithm 1.

Algorithm 1 Original β–hill climbing pseudo-code

$x_i = LB_i + (UB_i - LB_i) \times U(0, 1), \forall i = 1, 2, 3,$ and 4 {The initial solution x}
Calculate $f(x)$
$itr = 0$
while ($itr \le$ Max_Itr) **do**
  $x' = improve(\mathcal{N}(x))$
  for $i = 1, \cdots, N$ **do**
    if $(r \le \beta)$ **then**
      $x_i' = LB_i + (UB_i - LB_i) \times U(0, 1)$
    end if $\{r \in [0, 1]\}$
  end for
  if $f(x') \le f(x)$ **then**
    $x = x'$
  end if
  $itr = itr + 1$
end while

The proposed best β-hill climbing algorithm is shown in algorithm 2. We add the tournament selection [11, 38] in addition to the random selection in the β-operator to improve the algorithm exploration. Furthermore, the $\mathcal{N}$-operator is replaced with information from the global best to improvise the exploitation using the following modified equation adapted from [23, 41]:

$$x_i' = x_{Best,j} + U(0, 1) \times (x_{r_2,j}/v - x_{r_3,j}/v)$$

where r_1 and r_2 are two distinct integers from the range of available solutions, and v is the picked value in the β-operator.

3.2 Selection in memory consideration of harmony search

The selection method used in HSA has an effect on improving the diversity of the generated new harmonies [10] and as a result, affects the ability of the algorithm to escape from premature convergence. The selection schemes can be categorized into static and dynamic [19]. The dynamic selection changes the chance of picking a solution at each iteration [10] such as roulette wheel selection. On the other hand, the static selection chance of picking a solution stays fixed during searching such as tournament selection.

The proportionate (*or* Roulette wheel) selection scheme has been used in genetic algorithms [46]. This selection method relies on using the fitness value of any solution and compared against other solutions fitness values. The steps of this selection scheme are presented in algorithm 3.

As proposed in Albetar et al. [10], the tournament selection method picks randomly k solutions from the HM and after that chooses the best fitness solution. Algorithm 4 shows the procedure of tournament selection.

A comparison of six different selection methods for HSA in Albetar et al. [10] recommends the use of roulette wheel and tournament selection as they outperform other methods in most of the evaluated func-

Algorithm 2 The proposed best β–hill climbing pseudo-code

$itr = 0$
while ($itr \le$ Max_Itr) **do**
  i=random number from the range of available solutions
  if ($r \le SelectionRate$) **then**
    x'_i = Tournament Selection
  else
    $x'_i = LB_i + (UB_i - LB_i) \times U(0, 1)$
  end if
  $v = x'_i$
  for $j = 1, \cdots, N$ **do**
    if ($r \le \beta$) **then**
      $x'_j = LB_j + (UB_j - LB_j) \times U(0, 1)$
      Pick two distinct integers from the range of available solutions r_1 and r_2
      $x'_i = x_{Best,j} + U(0, 1) \times (x_{r_2,j}/v - x_{r_3,j}/v)$
    end if $\{ r \in [0, 1]\}$
  end for
  if $f(x') \le f(x)$ **then**
    $x = x'$
  end if
  $itr = itr + 1$
end while

Algorithm 3 Pseudocode for the roulette wheel selection method

Set $r \sim U(0, 1)$.
Set $found = False$
Set $sum\_prob = 0$.
Set $k = 0$.
while ($i \le$ HMS) AND NOT($found$) **do**
  $sum\_prob = sum\_prob + p_i$
  if ($sum\_prob \ge r$) **then**
    $k = i$
    $found = True$
  end if
  $i = i + 1$
end while
Return(k)

Algorithm 4 Pseudo code for the tournament selection method

Set t=tournament size
Set T= randomly picked t solutions from the HM
Return index of the best solution from T

tions. The proposed HSA replaces the random memory consideration in the original HSA with roulette wheel or tournament selection based on a specific probability.

3.3 Polynomial mutation

One of the Genetic Algorithms (GAs) operators is mutation which is used to modify some features of the newly generated chromosome [65]. The aim of applying such operation is to diversify the newly created solution to escape from being stuck in local optima. Several mutation schemes are presented in Hasan et al. [43] to alter the original HSA. The evaluation results when comparing six different mutation schemes prove that *highly disruptive polynomial* mutation (shown in algorithm 5) gives better results in most of the cases. Our proposed HSA applies a modified version of polynomial mutation (shown in algorithm 6) in both random consideration and pitch adjustment operators. We call this version best polynomial mutation.

Algorithm 5 Highly disruptive polynomial mutation

$i \leftarrow$from the range of available solutions
$r \leftarrow U[0, 1]$
if $(r \le P_m)$ **then**
$\delta_1 \leftarrow \frac{x_i - LB_i}{UB_i - LB_i}$
$\delta_2 \leftarrow \frac{UB_i - x_i}{UB_i - LB_i}$
$r \leftarrow U[0, 1]$
$$\delta_q \leftarrow \begin{cases} [(2r) + (1 - 2r) * (1 - \delta_1)^{\eta_m + 1}]^{\frac{1}{\eta_m + 1}} - 1 & \text{if } r \le 0.5 \\ 1 - [2(1 - r) + 2(r - 0.5) * (1 - \delta_2)^{\eta_m + 1}]^{\frac{1}{\eta_m + 1}} & \text{otherwise} \end{cases}$$
$x_i \leftarrow x_i + \delta_q.(UB_i - LB_i)$
end if

Note that n is the number of decision variables, P_m is the mutation probability and η_m is the distribution index presented as a non-negative value. Each decision variables x_i has an upper bound UB_i and a lower bound LB_i.

Algorithm 6 The proposed best polynomial mutation

$i \leftarrow$ taken from the range of available solutions
$r \leftarrow U[0, 1]$
if $(r \le P_m)$ **then**
$\delta_1 \leftarrow \frac{x_i - LB_i}{UB_i - LB_i}$
$\delta_2 \leftarrow \frac{UB_i - x_i}{UB_i - LB_i}$
$r \leftarrow U[0, 1]$
$$\delta_q \leftarrow \begin{cases} [(2r) + (1 - 2r) * (1 - \delta_1)^{\eta_m + 1}]^{\frac{1}{\eta_m + 1}} - 1 & \text{if } r \le 0.5 \\ 1 - [2(1 - r) + 2(r - 0.5) * (1 - \delta_2)^{\eta_m + 1}]^{\frac{1}{\eta_m + 1}} & \text{otherwise} \end{cases}$$
$y = (LB_i + (UB_i - LB_i) \times U(0, 1))/x_i$
$x_i' \leftarrow x_i + \delta_q.y$
end if

3.4 Best polynomial harmony search algorithm with best β-hill climbing

In order to improve the exploitation process of HSA, we embed the proposed best β-hill climbing as a local search method to dig deeper into the search space. The best β-hill climbing will be used after we generate the new harmony and it will iteratively work to improve it. In addition, we replace the techniques used in the three operators of the original HSA with efficient techniques that will help in a better exploration of the search space. The random selection in the memory consideration operator is replaced with roulette wheel

Algorithm 7 The proposed PHSβ–HC algorithm

Set HMCR, PAR, NI, HMS, SelectionRate, β–SelectionRate, β–Rate, β, β–NI.
$x_i^j = LB_i + (UB_i - LB_i) \times U(0,1), \forall i = 1, 2, \ldots, N$ and $\forall j = 1, 2, \ldots,$ HMS {generate HM solutions}
Calculate($f(\boldsymbol{x}^j)$), $\forall j = (1, 2, \ldots,$ HMS)
$itr = 0$
while ($itr \leq$ NI) **do**
  for $i = 1, \cdots, N$ **do**
    if ($U(0,1) \leq$ HMCR) **then**
      if ($U(0,1) \leq$ SelectionRate) **then**
        x_i' = roulette wheel selection
      else
        x_i' = tournament selection
      end if
      if ($U(0,1) \leq$ PAR) **then**
        x_i' = best polynomial mutation { pitch adjustment }
      end if
    else
      x_i'=best polynomial mutation { random consideration }
    end if
  end for
  The proposed best β–hill climbing shown in algorithm 2 improvise the generated new harmony x'
  if ($f(\boldsymbol{x}') < f(\boldsymbol{x}^{\text{worst}})$) **then**
    Include $\boldsymbol{x}'$ to the **HM**.
    Exclude $\boldsymbol{x}^{\text{worst}}$ from **HM**.
  end if
  $itr = itr + 1$
end while

or tournament selection schemes shown previously in algorithm 3 and 4. The random consideration and the pitch adjustment use the proposed best polynomial mutation presented in algorithm 6. The proposed best polynomial harmony search algorithm with best β–hill climbing (PHSβ–HC) steps are shown in algorithm 7. The steps of the algorithm are presented in Figure 1.

After that, two new concepts are introduced to enhance the algorithm:

- Pick HMCR and PAR randomly from the range of values suggested in the literature [10, 54].
- In the β–HC algorithm, we improvise the best solution found so far, not the newly generated harmony vector.

The updated algorithm is called improved PHSβ–HC algorithm (Imp. PHSβ–HC) and is shown in algorithm 8. The steps of the algorithm are presented in Figure 2.

4 Experimental Results

In this section, the proposed PHSβ–HC algorithm is experimentally evaluated on 13 classic benchmark functions shown in Table 1 [14, 39, 41, 78]. These functions cover a wide range of characteristics (i.e., unimodal, multimodal, separable, and non-Separable) as shown in Table 1. The experiments were executed on an Intel CORE i7 vPro 7th Gen with 16 GB of RAM where the proposed algorithm is coded using MATLAB R2016a under Windows 10.

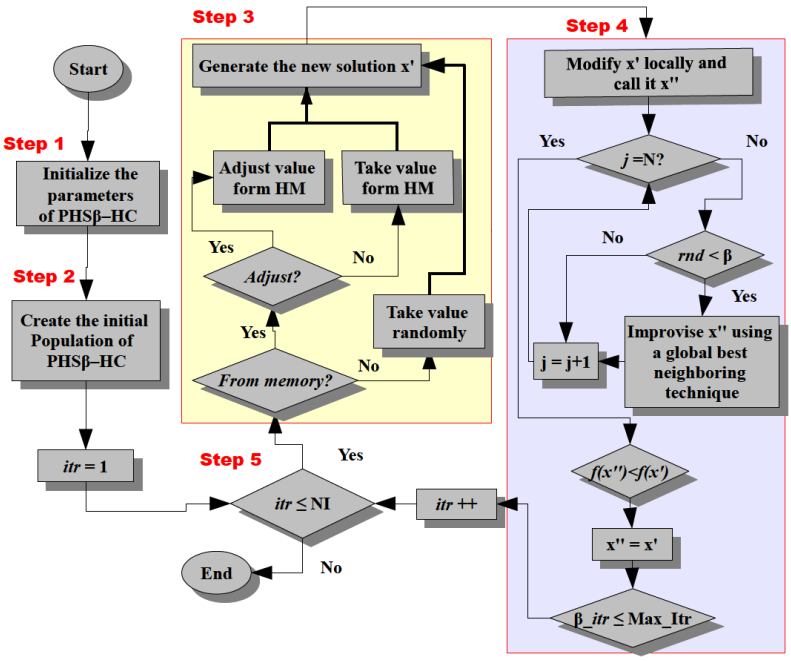

Figure 1: The flowchart of the proposed PHSβ–HC.

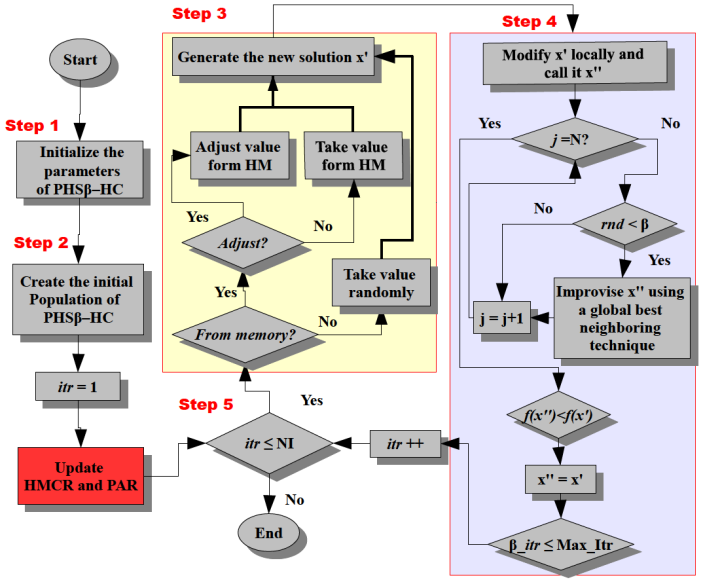

Figure 2: The flowchart of the proposed Imp. PHSβ–HC.

The dimensionality of all the test functions are set to D=30 which is similar to what has been done by the comparative methods. An extensive testing of different parameter settings is performed to study their effect on the convergence behaviour of the proposed methods. These parameters are set according to intensive investigation with guidance from literature [6, 44, 54]. The parameter settings for the proposed PHSβ–HC algorithm are as follows: HMS=100, HMCR=0.9, PAR=0.3, NI=50,000, SelectionRate=0.7, β–Rate=0.8, β–SelectionRate=0.8, β=0.3, β–NI=200, and $\eta_m = 10$. On the other hand, The parameter settings for the proposed improved PHSβ–HC algorithm are the same except HMCR=($U(0.9, 1)$, PAR=($U(0.1, 0.3)$.

The two proposed algorithms (PHSβ– HC and Imp. PHSβ–HC) are compared against the algorithms presented in Table 2 in terms of mean and standard deviation. The mean and the standard deviation for the tested

Algorithm 8 The proposed Imp. PHSβ–HC algorithm

Set HMCR, PAR, NI, HMS, SelectionRate, β–SelectionRate, β–Rate, β, β–NI.
$x_i^j = LB_i + (UB_i - LB_i) \times U(0,1), \forall i = 1, 2, \ldots, N$ and $\forall j = 1, 2, \ldots,$ HMS {generate HM solutions}
Calculate($f(\boldsymbol{x}^j)$), $\forall j = (1, 2, \ldots,$ HMS)
$itr = 0$
while ($itr \leq$ NI) **do**
  HMCR = $U(0.9, 1)$
  PAR = $U(0.1, 0.3)$
  for $i = 1, \cdots, N$ **do**
    if ($U(0,1) \leq$ HMCR) **then**
      if ($U(0,1) \leq$ SelectionRate) **then**
        x_i' = roulette wheel selection
      else
        x_i' = tournament selection
      end if
      if ($U(0,1) \leq$ PAR) **then**
        x_i' = best polynomial mutation { pitch adjustment }
      end if
    else
      x_i'=best polynomial mutation { random consideration }
    end if
  end for
  if ($f(\boldsymbol{x}') < f(\boldsymbol{x}^{\text{worst}})$) **then**
    Include $\boldsymbol{x}'$ to the **HM**.
    Exclude $\boldsymbol{x}^{\text{worst}}$ from **HM**.
  end if
  The proposed best β–hill climbing shown in algorithm 2 improvise the best solution found so far
  $itr = itr + 1$
end while

benchmark functions using is shown in Table 5. The best algorithm with the best result for each function is shown in bold text font. IGHS produces the best results for the five functions $f1, f3, f4, f6$, and $f10$. The

Table 1: The 13 classical test functions, Characteristics, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable

Function	Name	Characteristics	Range	f_{min}
$f1$	Sphere problem	US	$[-100, 100]^D$	0
$f2$	Schwefel's problem 2.22	UN	$[-10, 10]^D$	0
$f3$	Schwefel's problem 1.2	UN	$[-100, 100]^D$	0
$f4$	Schwefel's problem 2.21	US	$[-100, 100]^D$	0
$f5$	Rosenbrock's function	UN	$[-30, 30]^D$	0
$f6$	Step function	US	$[-100, 100]^D$	0
$f7$	Quatric function with noise	US	$[-1.28, 1.28]^D$	0
$f8$	Rastrigin's function	MS	$[-5.12, 5.12]^D$	0
$f9$	Ackley's function	MN	$[-32, 32]^D$	0
$f10$	Griewank function	MN	$[-600, 600]^D$	0
$f11$	Penalized function 1	MN	$[-50, 50]^D$	0
$f12$	Penalized function 2	MN	$[-50, 50]^D$	0
$f13$	Alpine function	MS	$[-10, 10]^D$	0

IGHS algorithm uses improved global best technique which picks the best individuals of the members in the generated solutions, which makes a balance between exploration and exploitation when solving *(unimodal, separable)* and *(unimodal, non-separable)* functions.

On the other hand, the proposed Imp. PHSβ–HC generates the best solution for the four functions $f2, f5, f6$, and $f7$. The algorithm escapes the problem of the local optimum. It is remarkable that the performance of Imp. PHSβ–HC obtains the best results on four functions which are *(unimodal , non-separable)* and *(unimodal, separable)*. This is because the proposed algorithm is a hybrid technique that uses a population-based technique empowered by a local search method.

In addition, the proposed algorithm PHSβ–HC generates the best results for the three functions $f11, f12$, and $f13$. This algorithm explores the search space deeply and diversifies the selected individuals in each iteration. As a result the algorithm obtains the best results for *(multi-modal , non-separable)* and *(multi-modal, separable)* functions. From the results, we can conclude that the two proposed algorithms are competitive when compared against the original HS and the other versions of HS in terms of obtaining the best solution for the different function.

The proposed algorithms provide a tradeoff between exploration and exploitation. The algorithms are capable of finding better solutions by adding diversity to the population which allows more exploration of the search space. Furthermore, the local search used allows more focused exploitation to find better solutions. The proposed Imp. PHSβ–HC algorithm has the ability to get out of a local minimum and can be efficiently used for unimodal, non-separable and unimodal, separable functions. On the other hand, the proposed PHSβ– HC algorithm is quite successful in optimizing multimodal, non-separable and multimodal, separable functions.

In order to compare the performance of the proposed algorithms (i.e., PHSβ–HC and Imp. PHSβ–HC) with multiple HS algorithms on all the benchmark function, the Friedman test is applied. Table 3 presents the average rankings of the two proposed algorithms against five other HS algorithms. The results show that the two proposed algorithms ranked 2nd and 3rd. On the other hand, IGHS obtains the best ranking.

Two-tailed t-test at a 0.05 significance level is applied to see if there is significant evidence that the mean of the proposed method is different than other methods means. The obtained results show that the proposed method PHSβ– HC mean is different than the methods HS, NGHS, and OHS. In addition, the proposed method Imp. PHSβ–HC mean is different than the methods HS, ACHS, GHS, NGHS, OHS, and IGHS.

4.1 Comparisons with other evolutionary algorithms (EAs)

The two proposed algorithms are compared with similar EAs including particle swarm optimization (PSO) and differential evolution (DE) algorithms presented in Table 4. Table 6 compares between the two proposed algorithms (PHSβ–HC and Imp. PHSβ–HC) and four versions of PSO. The four versions of PSO are HPSO-TVAC, CLPSO, FPSO, and OLPSO-G. The results are shown in Table 6 show that the proposed PHSβ–HC algorithm generates the best result for the functions $f6, f11$, and $f12$. In addition, the proposed Imp. PHSβ–HC algorithm is able to overcome the other algorithms for functions $f6$ and $f8$. The CLPSO algorithm obtains the best result for four functions $f5, f6, f7$, and $f10$.

Table 2: Key to comparative methods for the first group experiment with multiple HS algorithms.

No.	Abbr	Name	Citation
1	HS	Harmony Search algorithm	[35]
2	ACHS	HS with the adaptive control parameters strategy	[73]
3	GHS	Global best HS	[54]
4	OHS	HS with opposition-based learning	[30]
5	IGHS	Improved global-best HS	[28]

Table 3: Average rankings of HS, ACHS, GHS, OHS, IGHS, PHSβ–HC, Imp. PHSβ–HC on the 13 test functions gained by the Friedman test

Algorithm	Mean Rank
HS	5.81
ACHS	5.42
GHS	4.58
OHS	5.73
IGHS	3.27
PHSβ–HC	4.73
Imp. PHSβ–HC	3.81

Table 4: Key to comparative particle swarm optimization (PSO) and differential evolution (DE) algorithms.

No.	Abbr	Name	Citation
1	HPSO-TVAC	Self-organizing hierarchical particle swarm optimizer	[59]
2	CLPSO	Comprehensive learning particle swarm optimizer	[49]
3	FPSO	Fully informed particle swarm	[53]
4	OLPSO-G	Orthogonal learning particle swarm optimization	[80]
5	jDE	Self adaptive differential evolution	[21]
6	JADE	Adaptive differential evolution with optional external archive	[81]
7	SaDE	Differential evolution algorithm with strategy adaptation	[58]

In addition, the two proposed algorithms are compared with four variations of DE which are DE [66], jDE, JADE, and SaDE. It is clear from Table 7 that the proposed PHSβ–HC algorithm produces the best results for the functions $f6, f11$, and $f12$. Furthermore, the proposed Imp. PHSβ–HC algorithm outperforms the other algorithms for functions $f6$ and $f8$. The JADE algorithm outperform other algorithms in the five functions $f1, f2, f3, f7$, and $f9$.

Two-tailed t-test at a 0.05 significance level is applied to see if there is significant evidence that the mean of the proposed method is different than other methods means. The results show that the proposed method PHSβ– HC mean is different than the methods HPSOTVAC, CLPSO, FPSO, and OLPSOG. Furthermore, the proposed method Imp. PHSβ–HC mean is different than the methods HPSOTVAC, CLPSO, and OLPSOG. On the other hand, the proposed methods PHSβ– HC and Imp. PHSβ–HC mean is **not** different than the methods DE, jDE, JADE, and SaDE.

Table 8 present a summarization of the best obtained results between the comparative methods. The proposed methods are shown in bold font.

5 Conclusion and Future Work

This paper presented two new variations of harmony search with local search and updated harmony selection consideration, pitch adjustment and random consideration. The harmony selection utilized roulette wheel and tournament selection. The pitch adjustment and random consideration used a modified version of polynomial mutation called best polynomial mutation. These changes were introduced to improve the exploration of the algorithm. A recently proposed local search method called β–HC is used to improve the exploitation of the algorithm. The first proposed algorithm is called PHSβ–HC. In this algorithm, the HMCR and PAR are fixed and the local search is applied to the generated new harmony to improve it. The second proposed algorithm is called Imp. PHSβ–HC. This algorithm uses a randomly picked HMCR and PAR from an identified range based on the literature suggestion. The local search is applied to the best solution currently in the harmony memory.

Table 5: Experimental results of HS, ACHS, GHS, OHS, IGHS, PHSβ–HC , and Imp. PHSβ–HC over 30 independent runs on the 15 test functions

	HS		**ACHS**		**GHS**		**OHS**		**IGHS**		**PHSβ–HC**		**Imp. PHSβ–HC**	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
$f1$	2.81E−04	1.25E−05	5.95E−05	2.10E−06	1.00E−05	0.00E+00	2.93E−04	3.67E−05	**9.65E−08**	8.76E−09	1.29E−01	4.27E−02	7.16301E−07	1.03435E−06
$f2$	6.40E−02	6.82E−04	1.98E−02	3.27E−03	7.28E−02	0.00E+00	6.16E−02	1.76E−03	1.29E−03	9.28E−05	1.97E+00	1.90E−01	**9.67193E−05**	7.64385E−05
$f3$	8.30E+02	4.46E+02	4.32E+02	4.91E+01	5.83E+00	2.80E+00	1.87E+02	4.43E+01	**4.33E−07**	1.46E−07	4.23E−01	1.03E−01	3.85E+02	3.10E+02
$f4$	7.09E−01	1.77E−01	6.29E−01	1.99E−01	2.77E+00	1.23E+00	8.82E−01	2.30E−01	**1.42E−04**	1.25E−05	3.29E−01	3.11E−02	3.24E−02	1.50E−02
$f5$	3.29E+01	2.88E+01	3.38E+01	2.80E+01	4.97E+01	1.28E+01	3.10E+01	3.16E+01	6.08E+01	2.80E+01	3.07E+01	3.41E+01	**1.61E+01**	1.67E+01
$f6$	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
$f7$	6.01E−03	9.74E−04	4.21E−03	1.21E−03	**1.71E−03**	1.38E−03	5.37E−03	4.13E−04	2.24E−03	6.52E−04	5.40E+00	1.43E+00	**1.99E−02**	6.23E−03
$f8$	4.47E−05	4.92E−03	1.11E−02	2.87E−03	8.63E−03	0.00E+00	5.11E−02	7.26E−03	1.77E−05	3.53E−06	6.50E+00	7.66E−01	**2.07893E−07**	2.78854E−07
$f9$	**4.47E−05**	9.94E−05	5.94E−03	3.63E−04	2.09E−02	0.00E+00	1.27E−02	8.64E−04	2.21E−04	1.27E−06	4.64E−01	9.81E−02	1.16E−04	1.03E−04
$f10$	9.88E−03	8.05E−03	3.26E−02	3.31E−02	1.02E−01	0.00E+00	3.68E−02	1.40E−02	**3.29E−03**	4.65E−03	3.00E−02	2.90E−02	2.08E−02	2.45E−02
$f11$	2.12E−06	1.68E−07	4.81E−07	7.96E−08	8.03E−07	0.00E+00	2.05E−06	8.13E−08+	9.48E−10	1.03E−10+	**1.57054E−32**	5.5674E-48	2.79E−02	1.54E−02
$f12$	3.17E−05	2.55E−06	3.67E−03	5.18E−03	1.13E−05	0.00E+00	3.55E−05	2.02E−06	1.50E−08	2.73E−09	**1.34978E−32**	5.5674E-48	2.94E−09	4.74355E−09
$f13$	7.70E−03	1.10E−03	1.21E−02	3.37E−03	0.00E+00	0.00E+00	6.82E−03	1.37E−03	1.54E−04	5.57E−06	**1.42749E−13**	4.28212E−13	1.31E−05	1.02552E−05

Table 6: Comparisons between PHSβ–HC, Imp. PHSβ–HC, and PSO algorithms on 30-dimensional test functions

	HPSO-TVAC		**CLPSO**		**FPSO**		**OLPSO-G**		**PHSβ–HC**		**Imp. PHSβ–HC**	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
$f1$	**2.83E−33**	3.19E−33	1.58E−12	7.70E−13	2.40E−16	2.00E−31	4.12E−54	6.34E−54	1.29E−01	4.27E−02	7.16E−07	1.03E−06
$f2$	9.03E−20	9.58E−20	2.51E−08	5.84E−09	1.58E−11	1.03E−22	**9.85E−30**	1.01E−29	1.97E+00	1.90E−01	9.67E−05	7.64E−05
$f5$	2.39E+01	2.65E+01	**1.13E+01**	9.85E+00	2.81E+01	2.31E+02	2.15E+01	2.99E+01	3.07E+01	3.41E+01	1.61E+01	1.67E+01
$f6$	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
$f7$	9.82E−02	3.26E−02	**5.85E−03**	1.11E−03	4.16E−03	2.40E−06	1.16E−02	4.10E−03	5.40E+00	1.43E+00	1.99E−02	6.23E−03
$f8$	9.43E+00	3.48E+00	9.09E−05	1.25E−04	7.38E+01	3.70E+02	1.07E+00	9.92E−01	6.50E+00	7.66E−01	**2.08E−07**	2.79E−07
$f9$	7.29E−14	3.00E−14	3.66E−07	7.57E−08	2.17E−09	1.71E−18	**7.98E−15**	2.03E−15	4.64E−01	9.81E−02	1.16E−04	1.03E−04
$f10$	9.75E−03	8.33E−03	**9.02E−09**	8.57E−09	1.47E−03	1.28E−05	4.83E−03	8.63E−03	3.00E−02	2.90E−02	2.08E−02	2.45E−02
$f11$	2.71E−29	1.88E−28	6.45E−14	3.70E−14	5.51E−18	1.45E−34	1.59E−32	1.03E−33	**1.57E−32**	5.57E-48	2.79E−02	1.54E−02
$f12$	2.79E−28	2.18E−28	1.25E−12	9.45E−12	1.37E−17	3.42E−32	4.39E−04	2.20E−03	**1.35E−32**	5.57E-48	2.94E−09	4.74E−09

Table 7: Comparisons between PHSβ–HC, Imp. PHSβ–HC, and DE algorithms on 30-dimensional test functions

	DE		**jDE**		**JADE**		**SaDE**		**PHSβ–HC**		**Imp. PHSβ–HC**	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
$f1$	9.80E−14	8.40E−14	1.46E−28	1.78E−28	**1.32E−54**	9.22E−54	8.43E−06	3.63E−20	1.29E−01	4.27E−02	7.16E−07	1.03E−06
$f2$	1.60E−09	1.10E−09	9.02E−24	6.01E−24	**3.18E−25**	2.05E−24	3.51E−25	2.74E−25	1.97E+00	1.90E−01	9.67E−05	7.64E−05
$f5$	2.10E+00	1.50E+00	1.30E+01	1.40E+01	**3.20E−01**	1.10E+00	2.10E+01	7.80E+00	3.07E+01	3.41E+01	1.61E+01	1.67E+01
$f6$	4.70E+03	1.10E+03	6.13E+02	1.72E+02	5.62E+00	1.87E+00	5.07E+01	1.34E+01	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
$f7$	4.70E−03	1.20E−03	3.35E−03	8.68E−04	**6.14E−04**	2.55E−04	4.86E−03	5.21E−04	5.40E+00	1.43E+00	1.99E−02	6.23E−03
$f8$	1.80E+02	1.30E+01	3.32E−04	6.39E−04	1.33E−01	9.74E−02	2.43E+00	1.60E+00	6.50E+00	7.66E−01	**2.08E−07**	2.79E−07
$f9$	1.10E−01	3.90E−02	2.37E−04	7.10E−05	**3.35E−09**	2.84E−09	3.81E−06	8.26E−07	4.64E−01	9.81E−02	1.16E−04	1.03E−04
$f10$	2.00E−01	1.10E−01	7.29E−06	1.05E−05	1.57E−08	1.09E−07	**2.52E−09**	1.24E−08	3.00E−02	2.90E−02	2.08E−02	2.45E−02
$f11$	1.20E−02	1.00E−02	7.03E−08	5.74E−08	1.67E−15	1.02E−14	8.25E−12	5.12E−12	**1.57E−32**	5.57E-48	2.79E−02	1.54E−02
$f12$	7.50E−02	3.80E−02	1.80E−05	1.42E−05	1.87E−10	1.09E−09	1.93E−09	1.53E−09	**1.35E−32**	5.57E-48	2.94E−09	4.74E−09
$f13$	2.30E−04	1.70E−04	6.08E−10	8.36E−10	2.78E−05	8.43E−06	2.94E−06	3.47E−06	**1.43E−13**	4.28E−13	1.31E−05	1.03E−05

Table 8: Best obtained results between comparative methods

function	Best result HS	Best result PSO	Best result DE
f1	IGHS	HPSO-TVAC	JADE
f2	**Imp. PHSβ–HC**	OLPSO-G	JADE
f3	IGHS	-	-
f4	IGHS	-	-
f5	**Imp. PHSβ–HC**	CLPSO	JADE
f6	All functions	All functions	**Imp. PHSβ–HC; PHS-β–HC**
f7	GHS; **Imp. PHSβ–HC**	CLPSO	JADE
f8	**Imp. PHSβ–HC**	**Imp. PHSβ–HC**	**Imp. PHSβ–HC**
f9	HS	OLPSO-G	JADE
f10	IGHS	CLPSO	SaDE
f11	**PHS-β–HC**	**PHS-β–HC**	**PHS-β–HC**
f12	**PHS-β–HC**	textbfPHS-β–HC	**PHS-β–HC**
f13	**PHS-β–HC**	-	**PHS-β–HC**

The experimental results demonstrated the competitiveness of the two new proposed algorithms when compared against variations of HS, PSO, and DE. In the future, we will use the two proposed algorithms to solve economic dispatch and job scheduling problems. Another future direction can be evaluating the algorithm performance on the functions with other smaller or larger dimensions.

Acknowledgement: This research project is partially funded by the **Dartmouth College and American University of Kuwait** (Dartmouth-AUK) fellowship program.

References

[1] Iyad Abu Doush. Harmony search with multi-parent crossover for solving ieee-cec2011 competition problems. In *Proceedings of the 19th International Conference on Neural Information Processing - Volume Part IV*, ICONIP'12, pages 108–114, 2012.

[2] Iyad Abu Doush, Faisal Alkhateeb, Eslam Al Maghayreh, Mohammed Azmi Al-Betar, and Basima Hani F. Hasan. Hybridizing harmony search algorithm with multi-parent crossover to solve real world optimization problems. *Int. J. Appl. Metaheuristic Comput.*, 4(3):1–14, July 2013.

[3] Laith Mohammad Abualigah, Ahamad Tajudin Khader, and Mohammed Azmi Al-Betar. β-hill climbing technique for the text document clustering. In *New Trends in Information Technology NTIT2017 Conference, Amman, Jordan*, pages 1–6. IEEE, 2017.

[4] Laith Mohammad Abualigah, Ahamad Tajudin Khadery, Mohammed Azmi Al-Betar, Zaid Abdi Alkareem Alyasseri, Osama Ahmad Alomari, and Essam Said Hanandehk. Feature selection with β-hill climbing search for text clustering application. In *Second Palestinian International Conference on Information and Communication Technology (PICICT 2017), Gaza, Palestine*, pages 1–5. IEEE, 2017.

[5] M. A. Al-Betar and A. T. Khader. A harmony search algorithm for university course timetabling. *Annals of Operation Research*, pages 1–29, 2010. 10.1007/s10479-010-0769-z.

[6] M.A. Al-Betar, I.A. Doush, A.T. Khader, and M.A. Awadallah. Novel selection schemes for harmony search. *Applied Mathematics and Computation*, 218(10):6095–6117, 2011.

[7] Mohammed Azmi Al-Betar. β-hill climbing: an exploratory local search. *Neural Computing and Applications*, pages 1–16, 2016.

[8] Mohammed Azmi Al-Betar, Mohammed A Awadallah, Iyad Abu Doush, Emad Alsukhni, and Habes ALkhraisat. A non-convex economic dispatch problem with valve loading effect using a new modified beta-hill climbing local search algorithm. *Arabian Journal for Science and Engineering*, pages 1–18, 2018.

[9] Mohammed Azmi Al-Betar, Mohammed A Awadallah, Ahamad Tajudin Khader, and Zahraa Adnan Abdalkareem. Island-based harmony search for optimization problems. *Expert Systems with Applications*, 42(4):2026–2035, 2015.

[10] Mohammed Azmi Al-Betar, Iyad Abu Doush, Ahamad Tajudin Khader, and Mohammed A. Awadallah. Novel selection schemes for harmony search. *Applied Mathematics and Computation*, 2011.

[11] Mohammed Azmi Al-Betar, Ahamad Tajudin Khader, Zong Woo Geem, Iyad Abu Doush, and Mohammed A Awadallah. An analysis of selection methods in memory consideration for harmony search. *Applied Mathematics and Computation*, 219(22):10753–10767, 2013.

[12] Mohammed Azmi Al-Betar, Ahamad Tajudin Khader, and Munir Zaman. University course timetabling using a hybrid harmony search metaheuristic algorithm. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(5):664–681, 2012.

[13] Bilal Alatas. Chaotic harmony search algorithms. *Applied Mathematics and Computation*, 216(9):2687–2699, 2010.

[14] M Montaz Ali, Charoenchai Khompatraporn, and Zelda B Zabinsky. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of global optimization*, 31(4):635–672, 2005.

[15] Osama Alia and Rajeswari Mandava. The variants of the harmony search algorithm: an overview. *Artificial Intelligence Review*, 36:49–68, 2011.

[16] Emad Alsukni, Omar Suleiman Arabeyyat, Mohammed A Awadallah, Laaly Alsamarraie, Iyad Abu-Doush, and Mohammed Azmi Al-Betar. Multiple-reservoir scheduling using β-hill climbing algorithm. *Journal of Intelligent Systems*, 2017.

[17] Zaid Abdi Alkareem Alyasseri, Ahamad Tajudin Khader, Mohammed Azmi Al-Betar, and Laith Mohammad Abualigah. Ecg signal denoising using β-hill climbing algorithm and wavelet transform. In *ICIT 2017 The 8th International Conference on Information Technology*, pages 1–7, 2017.

[18] Zaid Abdi Alkareem Alyasseri, Ahamad Tajudin Khader, Mohammed Azmi Al-Betar, and Mohammed A Awadallah. Hybridizing β-hill climbing with wavelet transform for denoising ecg signals. *Information Sciences*, 429:229–246, 2018.

[19] Thomas Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, Oxford, UK, 1996.

[20] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[21] Janez Brest, Sao Greiner, Borko Boskovic, Marjan Mernik, and Viljem Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE transactions on evolutionary computation*, 10(6):646–657, 2006.

[22] Prithwish Chakraborty, Gourab Ghosh Roy, Swagatam Das, Dhaval Jain, and Ajith Abraham. An improved harmony search algorithm with differential mutation operator. *Fundamenta Informaticae*, 95(4):401–426, 2009.

[23] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: a survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, 15(1):4–31, 2011.

[24] Iyad Abu Doush, Mohammed Azmi Al-Betar, Mohammed A Awadallah, Abdelaziz I Hammouri, M Raed, Saba ElMustafa, and Habes ALkhraisat. Harmony search algorithm for patient admission scheduling problem. *Journal of Intelligent Systems*, 2018.

[25] Iyad Abu Doush, Mohammed Azmi Al-Betar, Ahamad Tajudin Khader, Mohammed A Awadallah, and Ashraf Bany Mohammed. Analysis of takeover time and convergence rate for harmony search with novel selection methods. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(4):305–322, 2013.

[26] Iyad Abu Doush, Mohammad Qasem Bataineh, and Mohammed El-Abd. The hybrid framework for multi-objective evolutionary optimization based on harmony search algorithm. In *First International Conference on Real Time Intelligent Systems*, pages 134–142. Springer, 2017.

[27] Iyad Abu Doush, Amal Lutfi Quran, Mohammed Azmi Al-Betar, and Mohammed A Awadallah. Max-sat problem using hybrid harmony search algorithm. *Journal of Intelligent Systems*, 27(4):643–658, 2018.

[28] Mohammed El-Abd. An improved global-best harmony search algorithm. *Applied mathematics and computation*, 222:94–106, 2013.

[29] Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

[30] XZ Gao, X Wang, SJ Ovaska, and K Zenger. A hybrid optimization method of harmony search and opposition-based learning. *Engineering Optimization*, 44(8):895–914, 2012.

[31] Z. W. Geem. Harmony search applications in industry. *Soft Computing Applications in Industry*, 226:117–134, 2008.

[32] Z. W. Geem. State-of-the-art in the structure of harmony search algorithm. In Z.W. Geem, editor, *Recent Advances In Harmony Search Algorithm*, volume 270 of *SCI*, pages 1–10. Springer-Verlag, Berlin, Heidelberg, 2010.

[33] Z. W. Geem, J. H. Kim, and G. V. Loganathan. A New Heuristic Optimization Algorithm: Harmony Search. *Simulation*, 76(2):60–68, 2001.

[34] Zong Woo Geem. Parameter estimation of the nonlinear muskingum model using parameter-setting-free harmony search. *Journal of Hydrologic Engineering*, 16(8):684–688, 2010.

[35] Zong Woo Geem, Joong-Hoon Kim, and G. V. Loganathan. A new heuristic optimization algorithm: Harmony search. *Simulation*, 76(2):60–68, 2001.

[36] Zong Woo Geem and Kwee-Bo Sim. Parameter-setting-free harmony search algorithm. *Applied Mathematics and Computation*, 217(8):3881–3889, 2010.

[37] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

[38] David Goldberg, K. Deb, and B. Korb. Messy genetic algorithms: motivation, analysis, and first results. *Complex Systems*, (3):493–530, 1989.

[39] Zhaolu Guo, Haixia Huang, Huogen Yang, Shenwen Wang, and Hui Wang. An enhanced gravitational search algorithm for global optimisation. *International Journal of Wireless and Mobile Computing*, 9(3):273–280, 2015.

[40] Zhaolu Guo, Huogen Yang, Shenwen Wang, Caiying Zhou, and Xiaosheng Liu. Adaptive harmony search with best-based search strategy. *Soft Computing*, 22(4):1335–1349, Feb 2018.

[41] Zhaolu Guo, Huogen Yang, Shenwen Wang, Caiying Zhou, and Xiaosheng Liu. Adaptive harmony search with best-based search strategy. *Soft Computing*, 22(4):1335–1349, 2018.

[42] Pierre Hansen and Nenad Mladenović. *An Introduction to Variable Neighborhood Search*, pages 433–458. Springer US, Boston, MA, 1999.

[43] Basima Hani F Hasan, Iyad Abu Doush, Eslam Al Maghayreh, Faisal Alkhateeb, and Mohammad Hamdan. Hybridizing harmony search algorithm with different mutation operators for continuous problems. *Applied Mathematics and Computation*, 232:1166–1182, 2014.

[44] Basima Hani F. Hasan, Iyad Abu Doush, Eslam Al Maghayreh, Faisal Alkhateeb, and Mohammad Hamdan. Hybridizing harmony search algorithm with different mutation operators for continuous problems. *Applied Mathematics and Computation*, 232(0):1166 – 1182, 2014.

[45] S Hemamalini and Sishaj P Simon. Artificial bee colony algorithm for economic load dispatch problem with non-smooth cost functions. *Electric Power Components and Systems*, 38(7):786–803, 2010.

[46] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, 1975.

[47] Gordon Ingram and Tonghua Zhang. Overview of applications and developments in the harmony search algorithm. In Zong Woo Geem, editor, *Music-Inspired Harmony Search Algorithm*, volume 191 of *SCI*, pages 15–37. Springer-Verlag, Berlin, Heidelberg, 2009.

[48] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[49] Jing J Liang, A Kai Qin, Ponnuthurai N Suganthan, and S Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE transactions on evolutionary computation*, 10(3):281–295, 2006.

[50] Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle. *Iterated Local Search*, pages 320–353. Springer US, Boston, MA, 2003.

[51] Mehrdad Mahdavi, Mohammad Fesanghary, and E Damangir. An improved harmony search algorithm for solving optimization problems. *Applied mathematics and computation*, 188(2):1567–1579, 2007.

[52] Raino A.E. Makinen, Jacques Periaux, and Jari Toivanen. Multidisciplinary shape optimization in aerodynamics and electromagnetics using genetic algorithms. *International Journal for Numerical Methods in Fluids*, 30(2):149–159, 1999.

[53] Rui Mendes, James Kennedy, and José Neves. The fully informed particle swarm: simpler, maybe better. *IEEE transactions on evolutionary computation*, 8(3):204–210, 2004.

[54] M. G. H. Omran and M. Mahdavi. Global-best harmony search. *Applied Mathematics and Computation*, 198(2):643–656, 2008.

[55] Quan-Ke Pan, PN Suganthan, JJ Liang, and M Fatih Tasgetiren. A local-best harmony search algorithm with dynamic subpopulations. *Engineering Optimization*, 42(2):101–117, 2010.

[56] Quan-Ke Pan, P.N. Suganthan, M. Fatih Tasgetiren, and J.J. Liang. A self-adaptive global best harmony search algorithm for continuous optimization problems. *Applied Mathematics and Computation*, 216(3):830 – 848, 2010.

[57] Zhaoqing Pan, Yun Zhang, and Sam Kwong. Efficient motion and disparity estimation optimization for low complexity multiview video coding. *IEEE Transactions on Broadcasting*, 61(2):166–176, 2015.

[58] A Kai Qin, Vicky Ling Huang, and Ponnuthurai N Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE transactions on Evolutionary Computation*, 13(2):398–417, 2009.

[59] Asanga Ratnaweera, Saman K Halgamuge, and Harry C Watson. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on evolutionary computation*, 8(3):240–255, 2004.

[60] Ali Sadollah, Hassan Sayyaadi, Do Guen Yoo, Ho Min Lee, and Joong Hoon Kim. Mine blast harmony search: A new hybrid optimization method for improving exploration and exploitation capabilities. *Applied Soft Computing*, 68:548 – 564, 2018.

[61] Rana Sawalha and Iyad Abu Doush. Face recognition using harmony search-based selected features. *International Journal of Hybrid Information Technology*, 5(2):1–16, 2012.

[62] A Immanuel Selvakumar and K Thanushkodi. A new particle swarm optimization solution to nonconvex economic dispatch problems. *Power Systems, IEEE Transactions on*, 22(1):42–51, 2007.

[63] Mahdi Shabani, Seyed Abolghasem Mirroshandel, and Hadi Asheri. Selective refining harmony search. *Expert Syst. Appl.*, 81(C):423–443, September 2017.

[64] Kenneth Sörensen. Metaheuristics?the metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18, 2015.

[65] W.M. Spears. *Foundations of Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA, 1993.

[66] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[67] Ata Allah Taleizadeh, Seyed Taghi Akhavan Niaki, and Farnaz Barzinpour. Multiple-buyer multiple-vendor multi-product multi-constraint supply chain problem with stochastic demand and variable lead-time: A harmony search algorithm. *Applied Mathematics and Computation*, 217(22):9234–9253, 2011.

[68] Z Tian, S Li, Y Wang, and X Wang. A network traffic hybrid prediction model optimized by improved harmony search algorithm. *Neural Network World*, 25(6):669, 2015.

[69] Zhongda Tian, Shujiang Li, and Yanhong Wang. Generalized predictive pid control for main steam temperature based on improved pso algorithm. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 21(3):507–517, 2017.

[70] Zhongda Tian, Yi Ren, and Gang Wang. Short-term wind speed prediction based on improved pso algorithm optimized em-elm. *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects*, 41(1):26–46, 2019.

[71] Zhongda Tian and Chao Zhang. An improved harmony search algorithm and its application in function optimization. *Journal of Information Processing Systems*, 14(5), 2018.

[72] Ayad Mashaan Turky and Salwani Abdullah. A multi-population harmony search algorithm with external archive for dynamic optimization problems. *Inf. Sci.*, 272(C):84–95, July 2014.

[73] Chia-Ming Wang and Yin-Fu Huang. Self-adaptive harmony search algorithm for optimization. *Expert Systems with Applications*, 37:2826–2837, 2010.

[74] Gai-Ge Wang, Amir H Gandomi, Xiangjun Zhao, and Hai Cheng Eric Chu. Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Computing*, 20(1):273–285, 2016.

[75] Lin Wang, Huanling Hu, Rui Liu, and Xiaojian Zhou. An improved differential harmony search algorithm for function optimization problems. *Soft Computing*, 2018.

[76] Ling Wang, Quan-Ke Pan, and M. Fatih Tasgetiren. A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem. *Computers & Industrial Engineering*, 61(1):76 – 83, 2011.

[77] Bin Wu, Cunhua Qian, Weihong Ni, and Shuhai Fan. Hybrid harmony search and artificial bee colony algorithm for global optimization problems. *Computers & Mathematics with Applications*, 64(8):2621–2634, 2012.

[78] Xin Yao, Yong Liu, and Guangming Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary computation*, 3(2):82–102, 1999.

[79] Jin Yi, Liang Gao, Xinyu Li, and Jie Gao. An efficient modified harmony search algorithm with intersect mutation operator and cellular local search for continuous function optimization problems. *Applied Intelligence*, 44(3):725–753, Apr 2016.

[80] Zhi-Hui Zhan, Jun Zhang, Yun Li, and Yu-Hui Shi. Orthogonal learning particle swarm optimization. *IEEE transactions on evolutionary computation*, 15(6):832–847, 2011.

[81] Jingqiao Zhang and Arthur C Sanderson. Jade: adaptive differential evolution with optional external archive. *IEEE Transactions on evolutionary computation*, 13(5):945–958, 2009.

[82] Tian Zhongda, Li Shujiang, Wang Yanhong, and Wang Xiangdong. Svm predictive control for calcination zone temperature in lime rotary kiln with improved pso algorithm. *Transactions of the Institute of Measurement and Control*, 40(10):3134–3146, 2018.