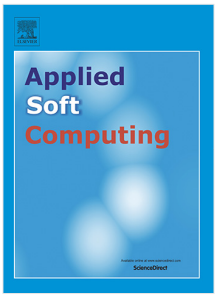
Journal Pre-proof



A novel harmony search algorithm and its application to data clustering

Kazem Talaei, Amin Rahati, Lhassane Idoumghar

PII: S1568-4946(20)30213-1
DOI: https://doi.org/10.1016/j.asoc.2020.106273
Reference: ASOC 106273

To appear in: *Applied Soft Computing Journal*

Received date : 18 October 2019
Revised date : 3 March 2020
Accepted date : 2 April 2020

Please cite this article as: K. Talaei, A. Rahati and L. Idoumghar, A novel harmony search algorithm and its application to data clustering, *Applied Soft Computing Journal* (2020), doi: https://doi.org/10.1016/j.asoc.2020.106273.

A novel harmony search algorithm and its application to data clustering

Kazem Talaei[a], Amin Rahati[a], Lhassane Idoumghar[b]
[a]Department of Computer Science, University of Sistan and Baluchestan, Zahedan, Iran[1]
[b]Universite de Haute-Alsace, IRIMIS-UHA, Mulhouse, France.

Abstract: This paper presents a variant of harmony search algorithm (HS), called best-worst-mean harmony search (BWM_HS). The main difference between the proposed algorithm and the canonical HS is that it employs a modified memory consideration procedure to utilize more efficiently the accumulated knowledge and experience in harmony memory (HM). To this aim, the random harmony selection scheme of this procedure is replaced with three novel pitch selection and production rules. These rules use the information of the current best and worst harmonies as well as the mean of all harmonies to guide the search process. To further utilize the valuable information of HM, two new harmonies are generated at each iteration where the better one will compete with the current worst harmony. The mean of all harmonies is always employed to produce a new harmony. On the other hand, each pitch of the second one is obtained by the rules that consider the information of the best and worst harmonies. These rules can present either explorative or exploitative search behaviors at different stages of search. Thus, a probabilistic self-adaptive selection scheme decides to choose between them to properly balance the exploration and exploitation abilities. The general performance of BWM_HS for solving optimization problems is evaluated against CEC 2017 benchmark functions and its results are compared with HS and eight state-of-the-art variants of HS. The comparison indicates that the performance of BWM_HS is better than or equal to the compared algorithms with respect to the accuracy, robustness, and convergence speed criteria. Moreover, the performance of BWM_HS in solving clustering problems is investigated by applying it for clustering several well-known benchmark datasets. The experimental results show that, in general, the BWM_HS outperforms other well-known algorithms in the literature and in particular, it significantly improves the statistical results for one dataset.

Keywords— Harmony search; CEC 2017; Data clustering.

1. Introduction

In a modern society, many complex problems arise in science and engineering domains, such as improving the reliability of complex systems designed for industrial engineering [1], estimation of parameters for mathematical models of biological systems [2] or industrial chemical reactions [3], and prediction of protein structure for drug design [4]. It is possible to convert these problems into numerical optimization problems with some objective functions and then develop some processes, called optimization methods, to find the best solution among the possible solutions for maximizing or minimizing the objective functions [5-7]. Classical mathematical optimization approaches like linear, non-linear, or dynamic

[1] Corresponding author:
Email address: a.rahati@cs.usb.ac.ir (A.Rahati)

programming are exact methods of such processes [8]. However, the capability of these approaches for solving real-world optimization problems is very limited because they are computationally expensive, and in addition, they pose strong presumptions that are hardly met in such problems. As an alternative approach, during the last few decades, meta-heuristic optimization algorithms have come into sight to efficiently tackle the optimization problems in various domains [9-18].

Harmony search (HS) [19] is an example of meta-heuristic algorithms that, due to its simplicity and effectiveness, has been applied in a wide spectrum of real world applications. For instance, a literature review conducted by Manjarres *et al.* [20] based on a reference repository which indexes 160 papers until 2012 reveals that HS has been employed in engineering, management, medical, robotic, control, power, and many cross-application areas. This algorithm imitates the improvisation process of musicians when they are looking for a new harmony. More precisely, HS simulates a candidate solution for a problem as a harmony where each decision variable of the problem is considered as a pitch of a different musical instrument. Then, HS moves candidate solutions in the search space to find the optimal solution through application of some search operators that model the pitch selection and adjustment actions in production of right harmonies. Despite significant success of HS applications, several studies have pointed out that search mechanisms of HS exhibit excellent explorative behavior but poor exploitative ability [21, 22]. This inefficiency is originated from the insufficient utilization of history information and experience gathered during search process in harmony memory (HM). To explain more, the pitch selection of canonical HS is implemented by a memory consideration mechanism that uses the value of decision variables of randomly selected harmonies from the current HM to generate a new harmony. In other words, it does not utilize the valuable information hidden in HM to follow specific search direction in the selection process. Thus, each time that a new harmony is generated, the algorithm is guided to a random new search direction that is more helpful for improving the exploration ability.

Several studies have tried to ameliorate the above-mentioned issue through replacing the random selection scheme in memory consideration with selection methods that utilize the fitness of harmonies while the rest of improvisation process remains untouched [23-25]. On the other hand, many works have designed a modified improvisation process that considers information of different harmonies from HM, either deterministically or probabilistically [22, 23, 26-37].

Indeed, the performance of any population-based optimization algorithm depends on the extent of balance between its exploitation and exploration abilities during the course of search process. A reasonable balance is enhancing the exploration at the early stages of search to help the algorithm to explore the new regions of a large search space quickly. Then, gradually increasing the exploitation ability of algorithm at the later stages to develop the search space in the vicinity of the current good solutions. However, the harmony selection scheme of the aforementioned works does not take into account the stage of search process to find out which ability needs to be promoted and thus to select the appropriate information from HM. Furthermore, their approaches do not benefit from the population level information and only use the information of a few harmonies. Moreover, except few, they often generate a single new harmony in each iteration, which is not efficient in utilizing the information of HM in comparison with producing more new harmonies.

To overcome the above limitations, this paper develops a new variant of HS that is called best-worst-mean harmony search (BWM_HS). The main contributions reflected in the proposed BWM_HS algorithm are as follows:

- The random selection scheme in memory consideration of HS is substituted with three novel pitch selection and production rules that utilize the accumulated knowledge and experience in harmony memory (HM) to guide the search process.
- Two rules employ the individual level information, i.e. the information of current best and worst harmonies in HM. The other rule uses the population level information that is the mean of all harmonies in HM.
- The proposed algorithm generates two new harmonies (instead of one) to increase the utilization of valuable history information of HM. A new harmony is produced by the rules that use the individual level information and another new one is obtained by the rule that shares the population level of information. To update HM, the better new harmony will compete with the current worst harmony of HM to replace it.
- Rules that utilize the individual level information are designed with a bias toward a special search direction during different stages of search process such that they boost either the exploitation or the exploration ability. Then, to properly establish the balance between these search behaviors, a probabilistic self-adaptive selection scheme helps the proposed algorithm to choose between them for generating each pitch of a new harmony.
- Despite all the above-mentioned modifications, the proposed BWM_HS algorithm does not change the computational complexity and the simple framework of canonical HS, and is straightforward to implement, too.

In order to demonstrate the performances of proposed algorithm, it is evaluated against CEC2017 test suite [38] and the obtained results are compared with HS and eight state-of-the-art variants of HS. The comparison indicates that the performance of the proposed algorithm is better than or equal to the considered algorithms in terms of accuracy, convergence speed, and robustness. Moreover, application of the proposed algorithm for solving ten benchmark clustering problems is verified and compared with HS variants and also ten state-of-the-art clustering algorithms. The proposed method has obtained competitive results compared to the other algorithms.

The remainder of this paper is organized as follows. Section 2 begins with describing the standard HS algorithm in Subsection 2.1. Then, a literature review regarding HS improvements is presented in Subsection 2.2. Next, the proposed algorithm is elaborated in Section 3. Thereafter, in Section 4, some experiments are performed on CEC 2017 as numerical optimization problems and also on some benchmark clustering problems; and then a comparison is made between the obtained results by the proposed algorithms with several well-known and state-of-the-art algorithms. Finally, the conclusions are summarized in Section 5.

2. A Review of HS Algorithm

HS is an optimization algorithm that imitates musicians' search for finding a perfect state of a harmony [19]. This algorithm, after initialization of parameters and HM, repeatedly executes a search mechanism to create a new harmony. This search mechanism includes randomly selecting a pitch from HM, randomly selecting a pitch from HM and adjusting it, and randomly generating a pitch. The greedy selection between the new generated harmony and the worst harmony in HM is performed to select the better harmony to enter the next generation. In the following, the search mechanism of the canonical HS is elaborated and then a brief overview of HS improvements is presented.

2.1. HS algorithm

The HS concept is based on natural musical performance process [19]. More precisely, a harmony and its collection of pitches are similar to a candidate solution and its set of decision variables, respectively. Also, measuring the pleasant state of a harmony through aesthetic estimation of audiences corresponds to an objective function. Each attempt by musicians for incremental improvement of a harmony via producing some new pitches corresponds to the application of search operators that changes the value of some decision variables at each iteration of HS. Musicians' memory which is a place for storing good harmonies is similar to a population of solutions that is known as HM in HS algorithm.

HS uses three rules in search for good harmonies and generating new pitches. The first rule randomly selects an existing pitch from HM as a new pitch. The second rule randomly selects an existing pitch from HM and adjusts it using an arbitrary distance bandwidth (BW) that determines its maximum variation. Finally, the third rule randomly generates a new pitch. A probabilistic parameter called harmony memory consideration rate (HMCR) is used to decide either to apply the first two rules or the third rule in generation of a new pitch. Furthermore, given that the first two rules are selected, another probabilistic parameter named pitch adjustment rate (PAR) is used to choose between the first and second rules for pitch generation.

Fig 1 shows the pseudo code of HS algorithm. It is started by specifying the values for control parameters of algorithm (line 1). In the next line, the HM is randomly initialized through random assignment of values to each j-th decision variable of each i-th harmony vector:

$$HM_{ij} = lb_j + rand * (up_j - lb_j) \quad (1)$$

Here, $i \epsilon [1, HMS]$ and $j \epsilon [1, D]$ where HMS is the harmony memory size and D is the dimension of optimization problem, respectively. Also, $rand$ is a uniform random number in [0, 1] and up_j and lb_j are the upper and lower bounds for the j-th decision variable. Lines 4 to 20 represent operations that are iterated in order to find optimal solution. The first operation improvises a new harmony vector (lines 5-13). As stated so far, this operation incorporates three pitch generation rules indicated in lines 7, 9, and 11, respectively, and also two probabilistic decision-making implemented by two *if* statements that use the HMCR and PAR parameters, respectively. The second operation updates HM (lines 14 to 18). In more detail, the fitness value of the new generated harmony is computed and the harmony with the worst fitness value in HM is determined. Thereafter, these two harmonies are compared based on their fitness values; and if the new generated harmony is better than the worst

harmony, then the HM is updated by replacing the worst harmony with the new generated harmony. The algorithm stops after meeting the termination condition which can be reaching a maximum number of iterations specified by *MaxIt* parameter.

Algorithm 1: HS Algorithm.

Set parameters, $MaxIt$ (maximum iteration), Dim(dimension of optimization problem), BW ($bandwidth$), PAR, and HMCR
Randomly initialize Harmony Memory (HM).
Evaluate the fitness value of each harmony vector in HM.
Repeat
  For *j = 1 to Dim* ***do***
    if *rand < HMCR*
      $X_{new,j} = X_{r,j} \quad r \in [1, 2, \dots, HMS]$
      if *rand < PAR*
        $X_{new,j} = X_{new,j} + rand * BW$
      End
    Else
      $X_{new,j} = lb_j + rand * (up_j - lb_j)$
    End
  End
  Evaluate the fitness values of X_{new} and select the worst harmony vector X_{worst} in the current HM
  if $f(X_{new}) < f(X_{worst})$
    $X_{worst} = X_{new}$
    $f(X_{worst}) = f(X_{new})$
  End
  iteration = iteration + 1
Until (iteration = MaxIt).
Algorithm stops and the best solution is obtained.

Fig 1: Pseudo code of HS algorithm

2.2. HS improvements

After introducing HS by Geem in 2001 [19], many variants and diverse applications of this algorithm have been presented in different fields. As mentioned before, a good literature survey on the early works can be found in [20]. Except studies that hybridize HS with other algorithms, modifications that have been made into HS can be divided into two categories according to whether they utilize the information of HM. In this review, modifications in each category are further divided into three groups: first, modifications related to fine-tuning of control parameters; second, modifications related to the search strategies of improvisation process. It is important to note that a variant of HS may use modifications belonging to different groups of both categories. Furthermore, in this review, modifications that hybridize HS with other algorithms are considered to belong to the third category with one group since both hybridized algorithms use the information of HM.

As already mentioned, in the first category that do not consider the information of HM, the first group of modifications is related to fine-tuning some of the key control parameters HMCR, PAR, and BW. For example, in [39-42], some linear or exponential equations are designed that are function of current generation and rely on some constant values such as a fixed minimum and maximum values for the control parameters and/or a constant maximum number of generations. Then, the equations iteratively change the value of control parameters in increasing or decreasing order at each cycle of algorithm. However, determining the constants, especially the minimum and maximum values for the control parameters is a

crucial task. Alatas [43], used different chaotic maps to generate a random number when it is needed by HS algorithm. In addition, he built seven chaotic HS algorithms in those PAR and/or BW which are either set by fixed values or by a specific chaotic map in each iteration. However, the random setting of control parameters PAR and BW may affect the balance between exploration and exploitation abilities. Recently, Luo et al. [44] employed a parameter setting free technique for configuration of HMCR, PAR, and BW parameters. In more detail, HMCR is dynamically valued by a normal random number at each iteration. PAR is determined by a strictly monotonic decreasing function with respect to the iteration number. Finally, a specific function is used to fine-tune the BW parameter so that it can vary from a wider range to a narrow range. In summary, they defined HMCR as a nonlinear function of the dimension of the problem and calculated PAR as a linear decreasing function with respect to the number of iterations.

The second group of the first category includes few studies. Hasan et al. [45] investigated the application of five different mutation schemes instead of random mutation in standard HS during the random consideration process to enhance the explorative behavior.

As an example of the works in the first group of the second category, Enayatifar et al. [30] employed learning automata mechanism. In their approach, a learning automaton with N actions is considered for each control parameter. Given that a value for each parameter is selected from its permitted range of values, then each action corresponds to one of the N equally discretized distance values in such a range for that parameter. The fitness information of the current worst harmony in HM and the new generated harmony are used to decide whether to punish or reward the selected actions. In fact, selected actions are rewarded if they generate new better solutions otherwise, they are punished. Kattan et al. [46] deployed two new quality measures where each one is used for dynamic tuning of the PAR and BW parameters, correspondingly. The first one uses the fitness information of best and worst harmonies. More precisely, it is obtained by the ratio of the current best harmony fitness values to the current worst fitness value. Then, the PAR value is computed based on this ratio during the search process rather than the current iteration count. The second quality measure is the improvisation acceptance rate percentage that indicates the total number of accepted improvisations in the last 100 iterations. The pitch adjustment process in their approach uses the value of this quality measure for dynamic calculation of BW for each component of harmony vectors. In another work, Zhao et al. [26] introduced an updating rule for BW parameter to increase the population diversity and ameliorate the exploitation ability during search process. Their rule utilizes the information of the difference between maximum and minimum value of each decision variable in the current HM to choose a BW value for that decision variable. Das et al. [47] introduced an explorative harmony search algorithm, which uses the information of the standard deviation of each decision variable to compute the value of BW in each iteration for that variable. Luo et al. [44], adaptively calculate BW for each decision variable through a differential term between the values of that decision variable with another random decision variable.

The second group of the second category is related to studies that some, if not all, of the modifications they made in the improvisation process include strategies that utilize the information of HM. As mentioned in Subsection 2.1, the improvisation process generates a new pitch via three rules including, first memory consideration rule, second pitch adjustment rule, and third random generation. In the following, works related to the second category are

reviewed according to the rules they modified. Several studies in literature changed the memory consideration process to improve the performance of HS. For instance, Castelli et al. [33] replaced the random harmony selection of memory consideration procedure with the tournament selection to select two harmonies with good fitness from HM in each iteration. Then, the information of corresponding pitches in these harmonies is combined through a linear recombination operator to produce the new pitch. After, a mutation operator that uses the parameter PAR is applied; hence, the parameter BW is eliminated. Zhao et al. [26] instead of using a corresponding pitch from a random harmony that is selected by the harmony consideration procedure, employed the information of a random decision variable from the best harmony to generate a pitch. However, this approach can break the building structures of solutions. In addition, to further utilize the information of HM, they substitute the parameter BW for each decision variable with the difference between the maximum value and the minimum value of the corresponding variable in the current harmony memory. Al-Betar et al. [24] built several variants of HS via replacing the random selection process in the memory consideration procedure of canonical HS with different selection schemes. In particular, they considered global-best, fitness-proportional, tournament, linear rank and exponential rank selection.

Many works attempted to improve the procedure of pitch adjustment in HS. For example, based on the concept of swarm intelligence [48], Omran and Mahdavi [27] replaced the pitch adjustment operation of HS with the information of a random decision variable from the best harmony. As stated in [28, 30], this approach suffers from serious deficiencies: generating infeasible solutions and breaking the building blocks of candidate solutions. El-Abd [22] proposed a novel pitch selection and generation scheme that enhances the explorative behavior of the algorithm through adjusting a randomly selected pitch according to a fixed BW and a Gaussian distribution. Also, their algorithm improves the exploitative behavior via adjusting the pitch of current best harmony using the fixed BW and a uniform distribution. In [32], authors followed the same approach of El-Abd where the value of BW is dynamically calculated. Yadav et al. [49] modified the pitch adjustment process where the PAR value is dynamically calculated. Furthermore, to adjust the pitch of a randomly selected harmony, its objective function values is compared with the mean of the objective functions value of the whole HM. Then, harmonies that their objective function values are smaller than the mean of the objective functions of the whole HM are adjusted using two strategies that are selected probabilistically with equal chance to boost both intensification and diversification abilities. One strategy uses the information of the corresponding pitch from the best harmony and the other one employs a combination of information from the corresponding pitches of both best and worst harmonies. On the contrary, the pitches of harmonies that their objective function values are greater than the mean of the objective functions of the whole HM are adjusted by the information of a random pitch from the best harmony. Inspired by a mutation strategy of differential evolution (DE) [50, 51] , i.e. “DE/Best/1”, Guo et al. [52] modified the pitch adjustment mechanism to benefit from the information of global best harmony to improvise a new harmony. In another research, Guo et al. [36] employed the modified the pitch adjustment mechanism introduced by Zou et al. [35] to improvise a new harmony and thereby improve the exploitation ability. Given this new harmony, they also create its corresponding general opposition based solution [53]. By generating more than one new harmony, their approach enhances the exploration ability. Finally, the better harmony between these two new harmonies competes with the worst harmony. Wang et al. [54] modified the pitch adjustment

of HS by combining two search strategies that utilize the information of the best harmony and a random harmony, respectively. To clarify, the first one enhances the exploitation ability by guiding the new improvised harmony towards the best harmony. The second one improves the exploitation ability by guiding the new improvised harmony towards a random harmony.

Some researcher has tried to modify both the memory consideration and the pitch adjustment rules. For example, Ouyang et al. [55] presented an improved HS (LHS) that modifies the harmony memory consideration by generating two new harmonies at each cycle of proposed algorithm. To this aim, a new harmony is generated as it does in canonical HS in the improvisation process. After, they use the concept of opposition-based learning to produce the second new one. Finally, the better new harmony competes with the worst harmony to update HM. In addition, they replaced the pitch adjusting mechanism by two search operators. One operator implements the best harmony guided search strategy via utilizing the information of the best harmony. Another operator does the worst harmony guided search by using the information of the worst harmony. To balance between exploration and exploitation abilities, the selection between these two operators is decided by a self-adaptive global pitch adjusting mechanism.

Some works in the literature changed both the memory consideration and the random generation rules. Ouyang et al. [56] introduced two search mechanisms. The first mechanism, i.e. global learning, is a combination of learning operation of teaching-learning-based optimization [57] with the concept of swarm intelligence to promote the search capability of memory consideration phase. The second mechanism is a modification of a random selection, which benefits from both the initial search domain and current reliable variable space in HM to enhance the capability of exploration. These mechanisms are employed in different iteration modes of iterative processes of HS and those harmonies are evolved in each iteration.

Many works are reported in the literature that break down the original framework of canonical HS to improve its performance. Zou et al., [35] proposed a modified improvisation step where a new harmony is improvised based on a position updating formula that utilizes the information of the best and worst harmonies. In fact, in each iteration of this algorithm, the position updating formula improves the quality of the worst harmony of memory by moving it toward the best harmony. Although they employed a genetic mutation operator in the modified improvisation step to avoid local optima still the position updating formula can cause the problem of premature convergence. To overcome this issue, Ouyang et al. [21] proposed a new stochastic position updating mechanism that instead of only using the best harmony, it also gives a chance to other harmonies in harmony memory to improvise a new harmony in each iteration. Al-Betar et al. [58] borrowed the idea of structured population from other structured evolutionary algorithms such as cellular genetic algorithm [59] and used it to maintain a high-level of diversity in HM. In fact, structure methods define a neighborhood structure between individuals. Inspired by this idea, they modified improvisation step of HS so that it interacts only with neighborhoods of the specific harmony. In another work, Al-Betar et al. [60] incorporated the island model as the most popular structured population mechanism to improve the diversity of HM. This model divides harmonies into several groups that are called islands. Then harmonies in each island are independently evolved using the HS operators. Immigration between the islands is also performed based on a random ring topology. Yi et al. [61] presented a modified HS

algorithm, using a novel mutation operator and a cellular local search. In their approach, all harmonies are divided into a better part and a worse part according to their fitness. Thereafter, mutation and crossover operations are applied to generate new harmony vectors. Moreover, a cellular local search is employed to boost the exploration ability at the early stages and enhance exploitation ability at the final stages. Recently, Sarkhel et al. [62] employed an enhanced opposition based learning called piecewise opposition-based learning algorithm. In this algorithm, instead of computing and updating all the components of a counter estimate of a harmony at once, each component is computed separately and is added to the corresponding component of piecewise opposition harmony if its addition improves the fitness value of resulted counter estimate harmony.

Finally, the third category is related to hybrid algorithms. The reason for hybridization of algorithms is that each algorithm has its own weakness and strength; therefore, integration of algorithms that inherits their advantages in a complementary way can lead to an effective method. For example, Wang et al. [63] combined the pitch adjustment operation of HS with the bat algorithm. In order to increase the intensification and diversification of HS algorithm, Layeb [64] proposed a new hybrid algorithm that incorporates some features from quantum computing in the standard HS, such as quantum representation scheme and some quantum inspired operators. Wang et al. [65] hybridized HS with cuckoo search (CS) algorithm through utilization of the pitch adjustment operator of HS as a mutation operator in the cuckoo updating process of CS to obtain an algorithm with proper convergence speed. Ouyang et al. [66] presented a hybrid harmony search particle optimization with global dimension selection to augment the exploration and exploitation abilities. Abualigah et al. [67] proposed a hybrid of krill herd (KH) algorithm with the HS algorithm, namely Harmony-KHA that incorporates the improvisation process of HS into KH to ameliorate the global search capability. Recently, Assad et al. [68] built a hybrid of HS with simulated annealing (SA) algorithm so that suboptimal harmonies are also accepted with a probability determined by a parameter named temperature. The initial value of this parameter is high at the early stages of the search process to promote the exploitative behavior; and it linearly decreases to enhance the explorative ability in the subsequent iterations. Wang et al. [69] utilized two mutation operators, namely DE/best/1/bin and DE/rand/1/bin, for DE algorithm to promote local search capability of HS. Furthermore, they used the information of best harmony in a modified pitch adjustment process to accelerate the convergence speed. In addition, their algorithm benefits from dynamic adjustment of parameters to balance between exploitation and exploration behaviors during the search process. Tuo et al. [70] developed a new HS algorithm for solving large-scale optimization problems through integrating teaching-learning strategy of teaching-learning optimization into HS, dynamically adjusting the control parameters, and employing a mutation operator in the improvisation process. Shabani et al. [34] hybridized the canonical HS with a refinement procedure. This procedure is placed after the update step of HS. To refine some harmonies in harmony memory, first the procedure selects a specific number of harmonies for refinement via the tournament selection scheme. Next, a subset X^{best} of decision variables of best harmony is selected. Then, to refine a selected harmony, each decision variable of this harmony is replaced by a decision variable in X^{best} to find a variable of X^{best} that can provide maximum change in the fitness of harmony. This approach may produce infeasible solutions and can break the building structure of good solutions. They also hybridized the refinement procedure with GHS [27] to further utilize the information of best harmony.

Review of studies that utilize the information of HM to modify the improvisation process can be summarized as follows. The underlying framework of the canonical HS is destroyed in most of these HS variants. To clarify, the memory consideration procedure of the canonical HS is responsible for using the history information of HM to guide the search. Furthermore, as mentioned before, the strong exploration ability of HS is due to the memory consideration strategy that selects random harmonies. Therefore, any modification that intends to better utilize the information of HM and provide a proper balance between exploration and exploitation abilities should be implemented in the memory consideration procedure; otherwise, it breaks down the original HS framework. On the other hand, works that did modify the memory consideration procedure do not consider the stage of search process to decide how to select information from HM to provide a proper balance between exploration and exploitation abilities. Furthermore, a few works have attempted to generate more than one new harmony at each iteration which either does not use the information of HM for this aim [53, 55] or misused the information of global best harmony in HM [27, 34]. Finally, the information of whole harmonies has never been used for generation of new harmonies that can guide the search direction.

3. Proposed algorithm

As explained in Subsection 2.1, in each iteration of the standard HS, a new harmony is produced for updating the HM where pitches of this harmony are generated by applying one of the three following rules: randomly selecting a pitch from HM, randomly selecting a pitch from HM and adjusting it, randomly generating a pitch. Thus, the valuable information and experience already collected during search process and stored in HM is mainly utilized by the first rule. In fact, this rule implements the memory consideration process of canonical HS. However, this process only keeps the information of some randomly selected harmonies for producing the subsequent generation of harmonies. Therefore, valuable direction information hidden in HM that are beneficial for guiding the search are neglected by this process. Thus, new harmonies guide the search process in random new search directions that is excellent for improving the exploration ability. As a result, the canonical HS presents strong exploration ability but poor exploitation capability.

To address the issue of imbalance between exploration and exploitation abilities, this paper proposes to replace the memory consideration process (or the first rule) of canonical HS by three novel pitch selection and generation rules. These rules are called *mean*, *best*, and *worst* rules. The *mean* rule provides a way to benefit from the population level information. For this purpose, it calculates a mean harmony from the mean values of pitches in HM so it has the information of the center of all harmonies. On the other hand, the *best* and *worst* rules benefit from the individual level information because they employ the information of current best and worst harmonies in HM. The best and worst harmonies are those that have the best and worst fitness values, respectively. The information of these two harmonies can be used as appropriate representatives of the accumulated knowledge and experience so far to guide the search toward specific search directions. The proposed algorithm employs a probabilistic self-adaptive selection scheme to choose between these two rules to produce each pitch of a

new harmony. The aim of the probabilistic self-adaptive selection scheme is to provide a proper balance during different stages of the search process.

In contrast to canonical HS that produces one new harmony in each iteration, the proposed BWM_HS algorithm uses the three rules to produce two new harmonies at each iteration. Indeed, generating two new harmonies instead of one, increases the utilization of valuable information stored in HM. To explain more, the first new harmony is generated by employing the *mean* rule. On the other hand, the second new harmony is produced through a probabilistic selection and application of the *best* or the *worst* rules where a probabilistic self-adaptive mechanism decides either to select and apply the *best* or the *worst* rule for calculating each pitch.

The first three following subsections explain the *best*, *worst*, and *mean* rules for pitch selection and generation, respectively. Then, Subsection 3.4 introduces the probabilistic self-adaptive selection mechanism. Finally, the pseudo code of BWM_HS is elaborated in Subsection 3.5.

3.1. Best rule

So far, inspired by particle swarm optimization (PSO) algorithm [71] as a well-known swarm intelligent algorithm, many variants of HS [27-29, 34-37, 44, 52, 54, 55, 72, 73] took advantage of the information of global best harmony to generate new harmonies. These works have either used the direct information of the global best harmony or have combined it with other harmonies from HM. Furthermore, they employed this information in different steps of improvisation process. In this study, the following formula is proposed as the *best* rule for the BWM_HS algorithm:

$$X_{new,j} = X_{Best,j} + rand \times (X_{r,j} - X_{Best,j}) \qquad (2)$$

where $X_{new,j}$ is the *j*-th component of the new harmony,$X_{Best,j}$ denotes the *j*-th component of best harmony in HM, and $X_{r,j}$ represents the j-th component of a random harmony in HM. This rule searches a region of search space that lies between the current best harmony and other harmonies in HM. In more detail, at the beginning of the search that exploration is more desired than exploitation, this region is large enough due to the diversity of HM. However, as the search process proceeds, the size of this region reduces and so this rule increases the exploitation capability.

3.2. Worst rule

Harmonies generated by the *best* rule have a tendency to move toward the current best harmony regardless of their actual quality of fitness value. Thus, updating HM with such new harmonies will reduce the diversity of HM and consequently will arise the issue of premature convergence. The *worst* rule intends to avoid this issue by utilizing the information of worst harmony in generation of pitches for new harmonies. It is defined as follows:

$$X_{new,j} = X_{Worst,j} + rand \times (X_{r,j} - X_{Worst,j}) \quad (3)$$

where $X_{new,j}$ is the j-th component of the new harmony, $X_{Worst,j}$ denotes the j-th component of worst harmony in HM, and $X_{r,j}$ represents the j-th component of a random harmony in HM. As explained so far, each iteration of the modified improvisation step generates two new harmonies where one of them is obtained by a probabilistic selection and application of the *best* and *worst* rules. In other words, pitches of such a new harmony are produced by both rules. Moreover, as the following subsection will demonstrate, the probabilistic selection mechanism is self-adaptive so that it gives the *best* and *worst* rules a higher chance for selection at the early stage and at the final stages, correspondingly. Thus, the *worst* rule keeps sufficient diversity of harmonies in HM as the search process progresses and avoids premature convergence problem especially at the late stages of optimization.

3.3. Mean rule

The aforementioned *best* and *worst* rules use the individual level information that are the information of the current best and worst harmonies in HM. In contrary, the aim of *mean* rule is to provide a way to benefit from the population level information or the whole history information stored in HM. For this purpose, this rule calculates the mean of each decision variable of HM to obtain the center of all harmonies:

$$X_{new,j} = X_{r,j} \pm rand \times (Mean_j - X_{r,j}) \quad (4)$$

where $X_{new,j}$ is the j-th component of the new harmony,$Mean_j$denotes the mean value of the j-th component of all harmonies in HM, and $X_{r,j}$ represents the j-th component of a random harmony in HM. At the early stages of the search process, harmonies are some randomly distributed points in the search space. Thus, they are very spread out from the mean harmony and from each other and hence this rule results in an explorative behavior that searches around the current harmonies with large steps. However, as the search process proceeds, the harmonies are getting closer to the best harmony and so to each other; consequently, this rule realizes an exploitive behavior that searches for a new harmony in the vicinity of the current best harmony with small steps.

3.4. Self-adaptive selection mechanism

Several recent studies have argued that proper balance between exploitation and exploration behaviors is a very important feature of an effective and efficient meta-heuristic optimization algorithm [74, 75]. To achieve the balance, this study uses a parameter called *Etha* to probabilistically choose between *best* and *worst* rules in generation of a pitch for new harmonies. As Eq. 5 indicates *Etha* decreases in a descending order when the number of iterations increases:

$$Etha = 0.9 \times \exp(\log(1/9) \times (it/MaxIt)) \quad (5)$$

where *it* and *MaxIt* parameters respectively determine the current number of iterations and maximum number of iterations. Fig.2 indicates that in the early stages of search the *Etha* parameter gives more chance to the *best* rule (Eq. 2) than the *worst* rule (Eq. 3) for selection and vice versa for the final stages of the search. Thus, as explained in Subsection 3.2, the search behavior of BWM_HS will utilize the knowledge and experience accumulated so far during search in the best harmony and other harmonies to search around the best harmony with large steps. Although the search step becomes smaller and smaller as the search process proceeds but the *Etha* decreases too and consequently the probability of application *worst* rule (Eq. 3) increases that prevents the early convergence issue by increasing diversity in HM.

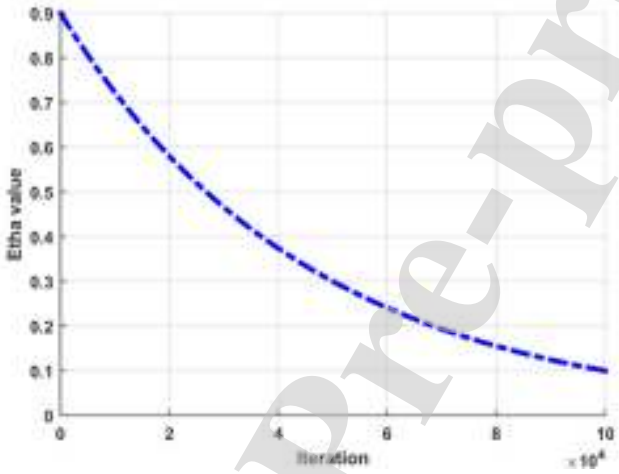


Fig 2: The variation of *Etha* parameter during the search process

So far, the self-adaptive selection mechanism and the three new pitch selection and production rules namely, *best*, *worst*, and *mean* rules were defined in this section. In the following, a rational explanation about the operation of these three rules is presented.

The improvisation process of BWM_HS produces two new pitches in each iteration. One new pitch $X^1_{new,j}$ is generated using the *mean* rule (Eq. 4) and another one $X^2_{new,j}$ is obtained by the application of either the *best* rule (Eq. 2) or the *worst* rule (Eq. 3). Assume that $X_{Best,j}$, $X_{Worst,j}$, $Mean_j$, and $X_{r,j}$ are the information of the best, worst, mean, and random harmonies utilized in these rules. Note that random harmonies used in these rules are different but for simplicity of explanation, they are considered similar. Without losing of generality, Fig. 3 can be used to evaluate the mechanism of these rules. In this figure, the location of $X_{Worst,j}$, $X_{Best,j}$, $Mean_j$, and $X_{r,j}$ are considered at A, B, C, and E, respectively. Furthermore, $WStep_j = | x_{r,j} - x_{worst,j} |$, $BStep_j = |x_{best,j} - x_{r,j}|$, and $MStep_j = | Mean_j - x_{r,j} |$ are differential terms in these rules that are defined here as worst, best, and mean adaptive steps, respectively. In the early stages of search process, all harmonies are widely distributed in the search space and so most adaptive steps are large. Hence, $X^1_{new,j}$ is generated in the wide region between C and D that help to explore the center of all harmonies. Similarly, $X^2_{new,j}$ is produced in the wide region either between A and E or E and B. According to the self-adaptive selection mechanism, the value of *Etha* should be large at the early stage of search and consequently the probability of selecting the best adaptive step is considerably higher than selecting the worst adaptive step. Therefore, the current best solution is explored quickly. As the search process proceeds, all adaptive steps as well as *Etha* decrease gradually

so that at the very middle stage *Etha* is 0.5. This means that both $BStep_j$ and $WStep_j$ have equal chance for selection. The best adaptive step guides the search direction toward exploiting in the vicinity of current best solution. On the other hand, the mean and worst adaptive steps help the algorithm to escape from local minimum.
At the final stages, C and E get very close to each other and also to the B because most harmonies have moved toward the global best harmony. This means that the adaptive $MStep_j$ is small and the *mean* rule will perform a local search in the vicinity of the global best harmony that helps to improve exploitation ability and the accuracy of proposed algorithm. On the other hand, according to the self-adaptive selection mechanism, the *Etha* is very small and the probability of selecting worst adaptive step is considerably higher than the best adaptive step that assists the algorithm to escape from the local optima.

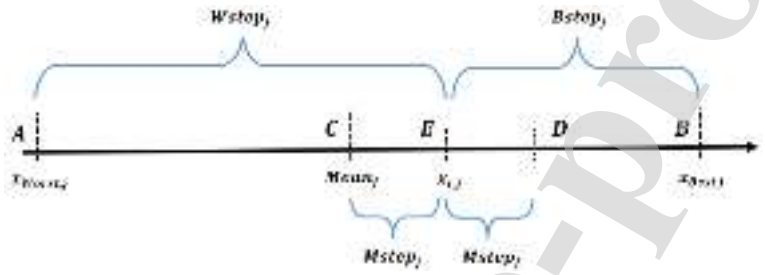


Fig 3: Schematic diagram of pitch production by the *best*, *worst*, and *mean* rules

3.5. Pseudo code of proposed algorithm

Fig. 4 shows the general trend of the proposed algorithm which can be summarized in the following steps. Line 1 defines and sets the initial values for the control parameters. Then, HM is randomly initialized and the fitness value of all harmonies is calculated (line 2-3). Next, *PAR* and *BW* are updated. Although the BWM_HS sets a fixed value for HMCR parameter (line 1) but it dynamically calculates the value of PAR and BW parameters (line 5) using a method proposed by Mahdavi et al. [42], as follows:

$$PAR(it) = PAR_{min} + \frac{(PAR_{max} - PAR_{min})}{MaxIt} \times it \quad (6)$$

$$BW(it) = BW_{max} \times \exp(c.it) \quad (7)$$

$$c = \frac{\ln\left(\frac{BW_{min}}{BW_{max}}\right)}{MaxIt}$$

where PAR_{min} and PAR_{max} are the minimum and the maximum adjusting rate, and BW_{min} and BW_{max} are the minimum and the maximum bandwidth, respectively. Furthermore $MaxIt$ is the maximum number of iteration and it is the current iteration. According to equations (6) and (7), PAR linearly increases and BW exponentially decreases through the search process, correspondingly. The large value of BW at the early iterations of optimization process increases the diversity of harmonies. On the contrary, the small value of BW with the large value of PAR enhances the local search and exploitation ability. The *Etha* parameter is calculated by *Eq. (5)* in line 6. The improvisation process is described in lines 7-22. As aforementioned, each iteration of the improvisation process produces two new harmonies which are called X^1_{new} and X^2_{new}. More precisely, lines 9-14 show that with the rate of HMCR the first new harmony X^1_{new} is generated by the *mean* rule *(Eq.(4))* and the second new harmony X^2_{new} is produced via probabilistic applications of the *best* rule *(Eq.(2))* and the

worst rule *(Eq.(3))* where the selection of these rules depends on the probabilistic parameter *Etha*. Then, lines 15-17 perform pitch adjustment for X^1_{new} and X^2_{new} using a method introduced by El-Abd [22]:

$$X^1_{new,j} = X^1_{new,j} + Gaussian(0,1) * BW \quad (8)$$

$$X^2_{new,j} = X^2_{new,j} + Gaussian(0,1) * BW \quad (9)$$

The *Gaussian(0,1)* in Eq. (8) and Eq. (9) is a Gaussian random number with a mean of zero and a standard deviation of one. The Gaussian distribution is used in this work because it provides a wider area of search than the uniform distribution around the pitch. Therefore, at the final stages of search that the PAR has a high value and the HM may have lost diversity, continues perturbation around the pitches with Gaussian distribution can reduce the possibility of getting trapped in local optimal. Next, lines 18-21 present the random production of two new harmonies. Finally, lines 23-28 exhibit that fitness for the two new harmonies is calculated and so the better new harmony is determined and will replace the worst harmony in HM if it is better than the worst harmony, too. When the stopping criterion of BWM_HS is met the best harmony of HM is presented as the final solution (line 31). A flowchart depicting these steps of the proposed BWM_HS algorithm is presented in Fig. 8 (Appendix A).

Algorithm 2. BWM_HS Algorithm.

1: *Set parameters* $MaxIt$, *Dim*, BW_{max}, BW_{min}, PAR_{max}, PAR_{min}, *and HMCR*
2: *Randomly initialize HM*
3: *Evaluate the fitness value of all harmonies in HM.*
4: ***Repeat***
5: *Update the PAR and BW parameters using Eq.(6) and Eq.(7)*
6: *Compute Etha values using Eq.(5)*
7: ***For*** *j = 1 to Dim* ***do***
8: ***if*** *rand* $<$ *HMCR*
9: *Compute the j-th component of* X^1_{new} *via the mean rule (Eq.(4))*
10: ***if*** *rand* $<$ *Etha*
11: *Compute the j-th component of* X^2_{new} *via the best rule (Eq.(2))*
12: ***Else***
13: *Compute the j-th component of* X^2_{new} *via the worst rule (Eq.(3))*
14: ***End***
15: ***if*** *rand* $<$ *PAR*
16: *Pitch adjustment of* X^1_{new} *and* X^2_{new} *using Eq. (8) and Eq. (9)*
17: ***End***
18: ***Else***
19: $X^1_{new,j} = lb_j + rand * (up_j - lb_j)$
20: $X^2_{new,j} = lb_j + rand * (up_j - lb_j)$
21: ***End***
22: ***End***
23: *Evaluate the fitness values of* X^1_{new} *and* X^2_{new} *and represent the best vector between them with* X_{new}
24: *Find the current worst harmony* X_{worst} *in the HM*
25: ***if*** $f(X_{new}) < f(X_{worst})$
26: $X_{worst} = X_{new}$
27: $f(X_{worst}) = f(X_{new})$
28: ***End***
29; *iteration = iteration + 1*
30: ***Until (iteration = MaxIt)***
31: ***Algorithm stops and the best harmony is represented***

Fig 4: Pseudo code of proposed BWM_HS algorithm

4. Experimental results and discussions

This section extensively evaluates the performance of BWM_HS algorithm. To this end, first in Subsection 4.1 some experiments are conducted on several well-known numerical optimization problems. Then, in Subsection 4.2, the application of BWM_HS on ten well-known datasets of clustering problems is investigated.

The BWM_HS algorithm is implemented via MATLAB 2015a environment using windows 10 operating system. All experiments are performed on a computer with a 2.19 GHz Intel(R), Xeon(R), 4 processor, and 16GB of RAM.

4.1. Experiment on numerical optimization problems

In the following, first, Subsection 4.1.1 describes the benchmark functions of CEC 2017 test suite as the considered numerical optimization problems. Then, Subsection 4.1.2 presents the experimental setup of BWM_HS in all conducted experiments. Next, the effect of the three novel pitch selection and production rules for the performance of BWM_HS is investigated in Subsection 4.1.3. Thereafter, Subsection 4.1.4 evaluating the effect of HMS on the performance of proposed algorithm. Subsection 4.1.5 introduces several state-of-the-art HS variants for the purpose of performance comparison with the BWM_HS and the next subsection explains the experimental setup for these algorithms. Subsection 4.1.7 starts with presenting and discussing the obtained results for the algorithms and ends with evaluating their convergence graphs. Thereafter, the time complexity of the proposed BWM_HS algorithm is compared with the other variants of HS in Subsection 4.1.8 and in the final subsection, the computational complexity of the BWM_HS algorithm is evaluated.

4.1.1. Benchmark functions of CEC 2017

This test collection contains 30 numerical minimization test functions that are divided into the four groups. First group includes three unimodal functions (*f1-f3*), second group contains seven simple multimodal functions (*f4-f10*), third group has ten hybrid functions (*f11-f20*), and finally the fourth comprises ten composition functions (*f21-f30*). These functions and their characteristics are summarized in Table 1. For more detailed information regarding these functions, please see [38].

Table 1.
Benchmark functions of CEC2017.

Function Type	NO.	Function Name	Optimal value
Unimodal	f_1	Shifted and Rotated Bent Cigar Function	100
	f_2	Shifted and Rotated Sum of Different Power Function	200
	f_3	Shifted and Rotated Zakharov Function	300

Simple Multimodal	f_4	Shifted and Rotated Rosenbrock's Function	400
	f_5	Shifted and Rotated Rastrigin's Function	500
	f_6	Shifted and Rotated Expanded Scaffer's F6 Function	600
	f_7	Shifted and Rotated Lunacek Bi_Rastrigin	700
	f_8	Shifted and Rotated Non-Continuous Rastrigin's Function	800
	f_9	Shifted and Rotated Levy Function	900
	f_{10}	Shifted and Rotated Schwefel's Function	1000
Hybrid	f_{11}	Hybrid Function 1 (N=3)	1100
	f_{12}	Hybrid Function 2 (N=3)	1200
	f_{13}	Hybrid Function 3 (N=3)	1300
	f_{14}	Hybrid Function 4 (N=4)	1400
	f_{15}	Hybrid Function 5 (N=4)	1500
	f_{16}	Hybrid Function 6 (N=4)	1600
	f_{17}	Hybrid Function 6 (N=5)	1700
	f_{18}	Hybrid Function 6 (N=5)	1800
	f_{19}	Hybrid Function 6 (N=5)	1900
	f_{20}	Hybrid Function 6 (N=6)	2000
Composition	f_{21}	Composition Function 1 (N=3)	2100
	f_{22}	Composition Function 2 (N=3)	2200
	f_{23}	Composition Function 3 (N=4)	2300
	f_{24}	Composition Function 4 (N=4)	2400
	f_{25}	Composition Function 5 (N=5)	2500
	f_{26}	Composition Function 6 (N=5)	2600
	f_{27}	Composition Function 7 (N=6)	2700
	f_{28}	Composition Function 8 (N=6)	2800
	f_{29}	Composition Function 9 (N=3)	2900
	f_{30}	Composition Function 10 (N=3)	3000

4.1.2. Experimental setup for the proposed BWM_HS

In all experiments, 51 independent runs of the BWM_HS algorithm are performed on the benchmark functions of CEC 2017 test suite with 10 dimensions (10D), 30 dimensions (30D), and 50 dimensions (50D) where the search range for all test functions is [-100,100]. According to the benchmark rules, the maximum number of function evaluations is set to 10,000×D and the obtained error values smaller than 10^{-8} are taken as zero. The values for the HMCR, PAR_{min}, and PAR_{max} are 0.9, 0.01, and 0.99, correspondingly. Finally, BW_{min} is 0.0001 and BW_{max} is computed as (*up-lb*)/20.

4.1.3. The effect of three pitch production rules

This section investigates the effect of three new pitch selection and production rules introduced in the proposed algorithm. To this end, six modified versions of the BWM_HS are

created through considering different combination of these three pitch production rules. These versions are called B_HS, M_HS, W_HS, BW_HS, MB_HS, and MW_HS. In more detail, the B_HS, M_HS, and W_HS generate only one new harmony at each iteration through employing the *best* rule *(*Eq. (2)*)*, the *worst* rule (Eq. (3)), and the *mean* rule *(*Eq. (4)*)*, correspondingly. The BW_HS employs the *best and* the *worst* rules to generate one new harmony at each iteration where these two rules are probabilistically selected for calculating each pitch via the introduced probabilistic selection scheme in Subsection 3.4. On the other hand, the MB_HS produces two new harmonies at each iteration via the *mean* and *best* rules. MW_HS algorithm is similar to MB_HS version, but employs the *worst* rule instead of the *best* rule.

Parameter settings for all these variants are similar to the parameter configuration of BWM_HS as described in Subsection 4.1.2. An experiment with 51 independent runs of BWM_HS and its variants are conducted on CEC 2017 test suite functions. Total sum of best (TSB) results of experiments in all dimension are demonstrated in Table 2. The supplementary details associated with Table 2 exhibited in Table. 11 (Appendix B).

Table 2:
TSB for 51 runs of BWM_HS and its six different versions on 10D, 30D, and 50D benchmarks of CEC 2017

	B_HS	M_HS	W_HS	BW_HS	MB_HS	MW_HS	BWM_HS
TSB on mean	4	17	11	3	2	22	38
TSB on std	9	22	21	2	7	21	17
TSB on both	3	9	7	1	1	13	16

Each row of Table 2 indicates the TSB results achieved by algorithms for each and both of mean and standard deviation criteria, respectively. According to the TSB for the mean criterion, the BWM_HS has the best results for 38 functions. More precisely, according to Table 11 reported in Appendix B, the BWM_HS has obtained the best mean results for 10D of *f1* and *f3* of unimodal functions; *f4* and *f9* of multimodal functions; *f12*, *f13*, *f14,* and *f15* of hybrid function; and all composition functions except *f23* and *f27*. Similarly, for 30D functions, the BWM_HS has outperformed others for all unimodal functions; *f6* of multimodal functions; *f13*, *f14*, *f16*, *f18,* and *f20* of hybrid functions; and *f27*, *f28*, *f29,* and *f30* of composition functions. For 50D functions, the BWM_HS has yielded the best mean results for *f1* of unimodal functions; *f4* of multimodal functions; *f12*, *f15*, *f17,* and *f19* of hybrid functions; and *f25* and *f30* of composition functions. In order to evaluate the stability of algorithms, according to the TSB for standard deviation criterion, M_HS, W_HS, MW_HS and BWM_HS has outperformed other considered algorithms by obtaining the TSB for 22, 21, 21, and 17 functions, respectively. However, by taking into account both stability and accuracy criteria, as the last row of Table2 presents, the BWM_HS and M_HS simultaneously obtained better mean and standard deviation values on 16 and 9 functions, respectively. So, it can be argued that the BWM_HS obtained the first rank of performance according to both stability and accuracy criteria and other modified algorithms have the second to sixth rank in the following order: MW_HS, M_HS, W_HS, B_HS, BW_HS, and finally MB_HS. To explain more, the worst performance belongs to MB_HS because its

search behavior is completely biased toward exploitation by using both the *best* and *mean* rules and consequently is trapped in local minimums. Similarly, B_HS suffers from weak exploration and early convergence by only employing the *best* rule. In contrast, the BW_HS has the stagnation issue due to inefficient use of information stored in population. The W_HS has the late convergence problem because of neglecting the promising search areas. The M_HS properly uses the information of population but has the imbalance issue in exploitation and exploration behavior during the search process. The MW_HS has alleviated the imbalance issue in the late stage of the search process but the problem still exists in the early stages. Thus, it can be claimed that any of the three proposed pitch production rules plays an important role in the performance of BWM_HS so that eliminating any rule will negatively affect the efficiency of the whole algorithm.

Some convergence graphs corresponding to the logarithmic scale of the mean error values for some 10D and 30D functions have been given in Fig. 9 (Appendix B)

4.1.4. Evaluating the effect of HMS

An important criterion for designing a new meta-heuristic algorithm is convergence speed especially if it is supposed to be applied on complex and real-world problems. The HMS is a parameter of HS algorithm that has an inverse relation with its performance since increasing the HMS will decrease its convergence speed. Thus, any improved version of HS is expected to represent a better performance with small HMS than with large HMS. Therefore, this section investigates the performance of BWM_HS algorithm with different HMS values, including5, 10, 20, 40, 80, 150, and 300 on 30D benchmarks of CEC2017. Other parameters setting is similar to those presented in Subsection 4.1.2. The results of 51 independent runs in terms of mean and standard deviation values of error values are demonstrated in Table 3. For the sake of clarity, the best results are shown in boldface in this table.

Table 3:
The mean and standard deviation results for 51 runs of BWM_HS with different values of HMS on 30D benchmarks of CEC 2017

	5		**10**		**20**		**40**		**80**		**150**		**300**	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f1	4.63E+03	6.03E+03	3.63E+03	3.95E+03	4.69E+03	5.23E+03	3.42E+03	4.10E+03	3.56E+03	3.76E+03	3.36E+03	4.41E+03	**2.70E+03**	2.65E+03
f2	8.04E-06	7.07E-06	8.39E-06	7.25E-06	**4.93E-06**	3.59E-06	3.15E-01	2.25E+00	1.61E+02	5.60E+02	1.41E+08	9.30E+08	4.64E+10	9.87E+03
f3	**1.11E-07**	6.52E-08	1.16E-07	8.09E-08	2.73E-07	1.23E-06	1.34E+03	1.26E+03	1.04E+04	5.59E+03	1.51E+04	6.54E+03	1.72E+04	5.73E+03
f4	**6.70E+01**	2.72E+01	8.44E+01	2.04E+01	8.90E+01	1.62E+01	9.16E+01	1.43E+01	9.55E+01	2.08E+01	9.62E+01	2.08E+01	1.03E+02	1.97E+01
f5	4.92E+01	1.56E+01	4.37E+01	1.37E+01	3.31E+01	8.02E+00	2.79E+01	8.38E+00	**2.33E+01**	5.57E+00	2.38E+01	6.88E+00	2.44E+01	6.20E+00
f6	1.07E-05	3.29E-05	1.11E-06	3.45E-06	5.52E-07	2.35E-07	**3.70E-07**	8.22E-08	3.91E-07	7.78E-08	7.05E-07	1.41E-07	2.47E-06	7.34E-07
f7	6.84E+01	9.79E+00	5.79E+01	7.15E+00	5.01E+01	5.68E+00	**4.72E+01**	4.38E+00	4.75E+01	2.06E+01	5.09E+01	3.31E+01	7.21E+01	5.04E+01
f8	5.19E+01	1.25E+01	4.61E+01	1.26E+01	3.35E+01	8.68E+00	2.80E+01	7.84E+00	2.90E+01	6.87E+00	**2.33E+01**	5.16E+00	2.63E+01	6.54E+00
f9	1.01E+01	7.18E+01	1.07E-02	6.46E-02	8.91E-03	6.36E-02	**6.19E-11**	2.31E-11	1.17E-10	5.34E-11	2.89E-10	1.40E-10	8.60E-10	3.91E-10
f10	2.04E+03	4.44E+02	**1.85E+03**	4.55E+02	1.93E+03	5.01E+02	1.92E+03	4.46E+02	1.86E+03	4.68E+02	1.88E+03	6.62E+02	2.17E+03	1.34E+03
f11	9.47E+01	4.05E+01	9.76E+01	3.48E+01	9.03E+01	4.28E+01	7.51E+01	3.03E+01	8.05E+01	3.41E+01	**7.45E+01**	3.00E+01	7.54E+01	3.10E+01
f12	**4.73E+05**	4.66E+05	7.93E+05	7.51E+05	7.28E+05	7.03E+05	1.00E+06	7.43E+05	9.91E+05	6.80E+05	9.19E+05	5.97E+05	9.58E+05	6.05E+05
f13	1.78E+04	1.63E+04	1.82E+04	2.00E+04	1.98E+04	1.59E+04	1.21E+04	1.27E+04	**1.09E+04**	1.20E+04	1.17E+04	1.07E+04	1.14E+04	8.98E+03
f14	**3.65E+03**	3.22E+03	5.33E+03	4.39E+03	9.62E+03	8.75E+03	1.12E+04	1.19E+04	1.23E+04	1.65E+04	1.55E+04	1.84E+04	2.18E+04	2.39E+04
f15	7.89E+03	9.58E+03	7.18E+03	8.32E+03	5.24E+03	4.86E+03	5.45E+03	6.00E+03	4.22E+03	5.79E+03	**3.53E+03**	5.57E+03	3.85E+03	4.44E+03
f16	5.18E+02	2.44E+02	5.40E+02	1.84E+02	4.91E+02	2.11E+02	4.88E+02	2.13E+02	**4.47E+02**	2.00E+02	4.69E+02	1.72E+02	4.94E+02	1.86E+02
f17	2.53E+02	1.18E+02	2.34E+02	9.57E+01	**1.82E+02**	9.69E+01	1.65E+02	8.74E+01	1.59E+02	1.06E+02	1.28E+02	8.22E+01	1.20E+02	9.87E+01
f18	**1.22E+05**	6.21E+04	1.57E+05	1.50E+05	1.87E+05	1.24E+05	1.50E+05	9.39E+04	3.00E+05	2.82E+05	4.41E+05	4.84E+05	4.60E+05	5.47E+05
f19	8.55E+03	1.24E+04	9.72E+03	1.38E+04	9.29E+03	1.07E+04	7.23E+03	9.23E+03	6.38E+03	9.18E+03	7.31E+03	9.32E+03	**4.89E+03**	4.90E+03
f20	2.72E+02	9.81E+01	2.57E+02	8.44E+01	2.15E+02	7.49E+01	1.94E+02	8.98E+01	1.79E+02	7.90E+01	**1.79E+02**	6.03E+01	1.92E+02	8.26E+01
f21	2.58E+02	1.40E+01	2.42E+02	1.14E+01	2.33E+02	9.18E+00	2.28E+02	6.95E+00	2.27E+02	7.46E+00	2.25E+02	7.61E+00	**2.24E+02**	7.61E+00
f22	1.69E+03	1.16E+03	1.52E+03	1.16E+03	1.12E+03	1.00E+03	1.14E+03	1.08E+03	6.81E+02	9.04E+02	5.86E+02	1.07E+03	**5.16E+02**	7.81E+02
f23	3.97E+02	4.51E+01	3.94E+02	1.27E+01	3.83E+02	9.46E+00	3.79E+02	9.35E+00	3.77E+02	8.95E+00	**3.74E+02**	8.35E+00	3.76E+02	9.80E+00

f24	4.76E+02	1.52E+01	4.62E+02	1.45E+01	4.55E+02	1.05E+01	4.47E+02	6.27E+00	4.45E+02	9.11E+00	**4.42E+02**	9.85E+00	4.45E+02	8.76E+00
f25	**3.87E+02**	2.71E+00	**3.87E+02**	1.58E+00	**3.87E+02**	1.57E+00	**3.87E+02**	8.08E-01	**3.87E+02**	2.05E+00	3.88E+02	1.50E+00	3.88E+02	3.66E+00
f26	1.23E+03	6.43E+02	1.23E+03	4.81E+02	1.15E+03	3.92E+02	**9.68E+02**	5.02E+02	1.01E+03	4.59E+02	1.05E+03	4.54E+02	1.14E+03	5.47E+02
f27	**5.04E+02**	1.19E+01	5.07E+02	9.14E+00	5.06E+02	1.07E+01	5.06E+02	8.62E+00	5.06E+02	7.76E+00	5.10E+02	6.36E+00	5.11E+02	7.97E+00
f28	**3.40E+02**	5.73E+01	3.61E+02	6.24E+01	3.49E+02	5.60E+01	3.68E+02	4.32E+01	3.88E+02	4.12E+01	4.02E+02	1.70E+01	4.07E+02	1.24E+01
f29	5.99E+02	1.03E+02	6.20E+02	1.33E+02	5.69E+02	8.06E+01	5.63E+02	9.10E+01	5.48E+02	1.08E+02	5.40E+02	7.06E+01	**5.17E+02**	6.41E+01
f30	1.07E+04	4.81E+03	1.08E+04	6.02E+03	1.02E+04	5.12E+03	1.05E+04	9.37E+03	8.32E+03	3.76E+03	8.14E+03	8.55E+03	**6.70E+03**	2.37E+03
Total	**8**		2		3		5		4		6		6	

The number of best performances gained by different HMS values are presented in the last row of Table 3. As can be seen, the BWM_HS algorithm with HMS = 5 outperforms the others on the majority of benchmark functions (*f3*, *f4*, *f12*, *f14*, *f18*, *f25*, *f27*, and *f28*). For simplicity of comparison, the change of mean error value with respect to the variation of HMS values for some functions are exhibited in Fig. 10 (Appendix C). Furthermore, convergence graphs corresponding to the logarithmic scale of the best values obtained by BWM_HS with different HMS values for some functions are presented in Fig. 11 (Appendix C). All of these results, confirm the superiority of BWM_HS with HMS = 5 compared with other configurations of HMS parameter.

4.1.5. Comparing the BWM_HS with HS variants

In this subsection, the performance of BWM_HS algorithm is compared with HS and eight recent variants of HS, including IHS [42], GHS[27], NGHS[35], SGHS[28], IGHS[22], GDHS[40], INGHS[21], and LHS[55].

4.1.6. Experimental setup for the compared HS variants

The parameters setting for the compared HS algorithms are extracted from the main sources related to them. Since several parameters and their settings in the compared algorithms are similar to the parameters of BWM_HS, so their settings can be found in Subsection 4.1.2. The HMS is 5 for all algorithms. The BW parameter is 0.01 for HS. The PAR is 0.03 for HS and SGHS. p_m parameter is 0.005 for NGHS and INGHS.

4.1.7. Experimental results on CEC 2017

An experiment constitutes of 51 independent runs of the BWM_HS, HS, and eight recent variants of HS mentioned in Subsection 4.1.5 is conducted for optimization of CEC 2017 benchmark functions. Tables 4 to 6 exhibit the results in terms of best, worst, mean, and standard deviation of error values obtained in all runs for 10D, 30D, and 50D of test functions, respectively. For the sake of clarity, the best results are shown in boldface in these tables. To indicate the significant difference among the performance of compared algorithms, Wilcoxon signed-rank test with significance level of 0.05 [76], is carried out on the mean results to perform a pair-wise comparison between the BWM_HS and other algorithms. The significance level indicator obtained by applying this test for each function is shown in these

tables with symbols '-', '+', and '=' that denotes the performance of the BWM_HS is worse than, better than, and similar to compared algorithm, respectively.

Table 4.
Results obtained for 51 independent runs of BWM_HS and eight recent variants of HS on 10D benchmarks of CEC2017.

No.	Algorithms	HS	IHS	GHS	NGHS	SGHS	IGHS	GDHS	INGHS	LHS	BWM_HS
	Unimodal										
f1	**Best**	7.869980e-01	1.648727e+01	5.873037e+05	1.641556e+05	4.023751e+08	**2.565695e-01**	3.612370e+01	2.239822e+02	4.276003e+00	5.565038e-01
	Worst	1.236168e+04	1.217240e+04	4.924865e+07	7.077183e+06	1.929224e+09	1.150846e+04	1.108628e+04	1.363949e+04	**8.017705e+03**	1.222942e+04
	Mean	3.510039e+03	4.098013e+03	9.138119e+06	1.653972e+06	1.021657e+09	3.010851e+03	2.230604e+03	5.134612e+03	2.054044e+03	**1.968016e+03**
	S.D	3.910953e+03	3.398666e+03	8.990707e+06	1.569262e+06	4.386615e+08	3.082084e+03	2.580019e+03	3.705001e+03	**2.010974e+03**	2.381597e+03
	Sign	=	+	+	+	+	=	=	+	=	
f2	**Best**	4.972222e-06	1.917370e-06	4.385454e+02	5.242990e+01	2.811071e+06	1.203952e-06	2.977476e-07	9.883005e-06	1.029363e-04	**3.076804e-08**
	Worst	5.324544e+02	3.887983e-04	1.996779e+06	1.115163e+08	1.131933e+09	6.720580e-05	**3.570716e-05**	8.777944e-03	2.622103e+01	2.350995e-04
	Mean	1.739406e+01	9.381433e-05	1.231318e+05	1.073169e+07	1.197720e+08	2.172051e-05	**1.028762e-05**	1.631586e-03	5.184911e-01	1.826849e-05
	S.D	7.933397e+01	9.706422e-05	3.090010e+05	2.355119e+07	1.787954e+08	1.580720e-05	**9.518666e-06**	2.128302e-03	3.671059e+00	3.997627e-05
	Sign	+	+	+	+	+	+	=	+	+	
f3	**Best**	1.141148e-05	**0.000000**	7.101264e+01	1.011180e+04	2.636111e+03	1.659703e-08	**0.000000**	1.447864e-04	1.219989e-01	**0.000000**
	Worst	2.543780e+03	**0.000000**	3.591467e+03	9.494388e+04	1.006427e+04	6.664351e-08	**0.000000**	2.379219e+01	2.557341e+03	**0.000000**
	Mean	3.670676e+02	**0.000000**	6.241080e+02	3.726044e+04	6.056116e+03	4.216116e-08	**0.000000**	1.078980e+00	1.668513e+02	**0.000000**
	S.D	4.591584e+02	**0.000000**	6.433706e+02	2.057986e+04	2.126746e+03	1.022540e-08	**0.000000**	4.171280e+00	4.810493e+02	**0.000000**
	Sign	+	=	+	+	+	+	-	+	+	
	Multimodal										
f4	**Best**	6.944250e-04	**5.710939e-06**	1.283933e+00	4.276931e-01	2.810135e+01	7.629659e-04	2.720493e-05	7.168883e-03	5.818823e-02	1.378827e-02
	Worst	7.919452e+01	7.137591e+01	1.437352e+02	1.674733e+02	1.846751e+02	6.708556e+01	4.737111e+00	6.503087e+00	7.879084e+01	**8.703892e-02**
	Mean	8.084789e+00	3.115428e+00	1.889917e+01	3.650556e+01	8.437276e+01	2.765947e+00	1.581461e-01	3.650604e+00	1.268365e+01	**6.303845e-02**
	S.D	1.391288e+01	9.811844e+00	2.890120e+01	4.492499e+01	3.541969e+01	1.297671e+01	6.938994e-01	2.485232e+00	2.134873e+01	**1.355577e-02**
	Sign	+	+	+	+	+	-	-	+	+	
f5	**Best**	4.974795e+00	2.984877e+00	7.152028e+00	6.546139e+00	3.912558e+01	8.954626e+00	3.292501e+00	5.969851e+00	5.969754e+00	**9.949591e-01**
	Worst	**1.890420e+01**	2.188906e+01	3.404692e+01	3.957488e+01	6.729341e+01	5.173769e+01	5.273259e+01	3.880395e+01	4.783333e+01	2.686383e+01
	Mean	1.129571e+01	1.160058e+01	1.543070e+01	1.889060e+01	5.468012e+01	2.367765e+01	1.781052e+01	1.995822e+01	1.678061e+01	**1.089236e+01**
	S.D	**3.646606e+00**	4.167861e+00	4.656200e+00	8.336649e+00	5.820873e+00	9.664739e+00	8.013143e+00	7.763779e+00	8.421084e+00	5.118904e+00
	Sign	=	=	+	+	+	+	+	+	+	
f6	**Best**	1.397412e-02	2.801781e-05	5.582077e-01	2.437713e-01	9.292785e+00	7.212700e-05	8.214665e-07	6.845464e-03	2.339849e-05	**5.377797e-08**
	Worst	3.013461e-01	1.072847e-03	3.208253e+00	2.272700e+00	3.370689e+01	1.220104e+00	1.366793e+01	4.387283e+00	1.468476e-01	**2.881011e-05**
	Mean	9.274549e-02	5.950880e-05	1.897024e+00	8.599394e-01	2.048915e+01	5.784915e-02	5.530487e-01	5.072855e-01	1.007465e-02	**1.788848e-06**
	S.D	5.533955e-02	1.450745e-04	5.848137e-01	4.438429e-01	5.698486e+00	1.826345e-01	2.140713e+00	9.773269e-01	2.436953e-02	**5.712656e-06**
	Sign	+	+	+	+	+	+	+	+	+	
f7	**Best**	7.946126e+00	7.596049e+00	2.374659e+01	1.932598e+01	6.626412e+01	1.641405e+01	1.057855e+01	1.705733e+01	1.502690e+01	**4.257606e+00**
	Worst	4.041459e+01	3.892518e+01	4.955328e+01	8.343332e+01	1.359165e+02	5.505019e+01	4.881127e+01	7.284514e+01	5.334438e+01	**2.802264e+01**
	Mean	2.598648e+01	2.612301e+01	3.466521e+01	3.452207e+01	9.787031e+01	3.060421e+01	2.705423e+01	4.124960e+01	3.133877e+01	**1.633615e+01**
	S.D	7.866585e+00	7.178911e+00	6.176040e+00	1.075409e+01	1.844236e+01	8.895446e+00	7.275566e+00	1.300597e+01	1.020596e+01	**3.795262e+00**
	Sign	+	+	+	+	+	+	+	+	+	
f8	**Best**	3.979836e+00	5.001402e+00	6.275163e+00	6.006639e+00	3.072783e+01	7.959667e+00	3.979836e+00	1.989920e+00	4.974795e+00	**1.989918e+00**
	Worst	**2.188906e+01**	3.482349e+01	2.787790e+01	4.617030e+01	6.977333e+01	4.875282e+01	3.581845e+01	3.283355e+01	4.377805e+01	2.387899e+01
	Mean	1.262441e+01	1.366598e+01	1.423824e+01	2.184852e+01	5.200889e+01	2.686670e+01	1.709998e+01	1.673450e+01	2.067992e+01	**1.026173e+01**
	S.D	**4.349917e+00**	5.354034e+00	5.053904e+00	7.816582e+00	7.541640e+00	9.576070e+00	7.251484e+00	7.190574e+00	8.343426e+00	5.171826e+00
	Sign	+	+	+	+	+	+	+	+	+	
f9	**Best**	1.407443e-08	**0.000000**	2.059293e+00	4.731854e-01	9.484413e+01	**0.000000**	**0.000000**	2.731854e-07	5.269483e-04	**0.000000**
	Worst	3.854495e+01	5.181821e+01	3.738175e+01	9.546755e+01	5.802942e+02	8.871784e+02	9.379478e+01	9.549845e+02	1.872784e+02	**0.000000**
	Mean	4.109657e+00	3.040345e+00	1.268540e+01	2.133258e+01	3.130293e+02	7.009952e+01	1.156374e+01	2.450535e+01	1.652648e+01	**0.000000**
	S.D	6.362313e+00	8.418491e+00	8.515457e+00	2.177184e+01	1.182765e+02	1.366945e+02	2.202181e+01	1.336544e+02	3.053618e+01	**0.000000**
	Sign	+	+	+	+	+	+	+	+	+	
f10	**Best**	3.682340e+01	**3.539870e+00**	4.923163e+01	1.416214e+02	1.132744e+03	1.304552e+02	3.664778e+00	1.255982e+02	1.369729e+02	1.253307e+02
	Worst	**5.959264e+02**	6.016523e+02	1.032726e+03	1.268318e+03	1.654018e+03	1.330507e+03	1.017471e+03	1.117393e+03	1.252957e+03	1.008754e+03
	Mean	**2.932318e+02**	3.103113e+02	4.354318e+02	6.879814e+02	1.394814e+03	6.944520e+02	5.347808e+02	5.448331e+02	6.579269e+02	5.210147e+02
	S.D	1.334247e+02	1.405516e+02	1.787510e+02	2.359405e+02	**1.176917e+02**	2.489044e+02	2.223285e+02	2.283328e+02	2.257632e+02	1.800041e+02
	Sign	-	-	-	+	+	+	=	=	+	
	Hybrid										
f11	**Best**	1.347442e+00	2.287957e+00	8.971400e+00	1.272163e+01	2.622758e+01	3.275735e+00	**1.113974e+00**	3.172600e+00	3.232759e+00	2.326204e+00
	Worst	2.211959e+02	**1.571708e+01**	3.595498e+02	1.514281e+04	2.050088e+02	3.616247e+01	2.322871e+01	4.458598e+01	1.785784e+02	3.391463e+01
	Mean	1.206740e+01	**8.685203e+00**	7.432346e+01	3.671772e+03	7.811046e+01	1.815045e+01	1.043317e+01	1.568513e+01	1.441046e+01	1.047121e+01
	S.D	3.015371e+01	**4.022975e+00**	8.585035e+01	4.600654e+03	3.421881e+01	7.834167e+00	5.394715e+00	1.041086e+01	2.412973e+01	6.087139e+00
	Sign	=	=	+	+	+	+	=	+	=	

f12	**Best**	**8.415072e+01**	4.064014e+02	1.882498e+04	3.410007e+04	4.114759e+05	3.440772e+02	3.655637e+02	1.835314e+03	8.982508e+03	3.336243e+02
	Worst	2.032793e+05	5.007817e+04	1.006830e+06	2.004131e+07	7.633192e+07	5.974353e+04	**4.848489e+04**	4.950047e+04	8.798468e+06	5.052152e+04
	Mean	2.656945e+04	1.268079e+04	1.880388e+05	4.413719e+06	1.943807e+07	1.577628e+04	1.472790e+04	1.566717e+04	1.500142e+06	**8.805041e+03**
	S.D	4.005595e+04	1.389870e+04	1.711160e+05	4.392951e+06	1.628423e+07	1.267922e+04	1.356999e+04	1.149498e+04	2.112859e+06	**1.031375e+04**
	Sign	+	=	+	+	+	+	+	+	+	
f13	**Best**	**7.576291e+00**	9.319124e+01	7.951392e+01	1.491991e+03	1.945157e+02	3.406264e+01	3.031745e+01	1.880987e+01	1.770223e+01	5.755605e+01
	Worst	3.034963e+04	3.094027e+04	5.016183e+04	7.079676e+05	3.174849e+04	2.771571e+04	2.877644e+04	3.134452e+04	3.124926e+04	**2.673318e+04**
	Mean	1.212937e+04	1.337586e+04	1.223967e+04	9.729687e+04	9.924388e+03	9.140446e+03	9.984996e+03	7.817193e+03	1.058585e+04	**7.593353e+03**
	S.D	9.303175e+03	1.109883e+04	1.144052e+04	1.520964e+05	9.653958e+03	9.247348e+03	8.206813e+03	8.467577e+03	9.093977e+03	**6.408238e+03**
	Sign	+	+	=	+	=	=	=	=	=	
f14	**Best**	6.961303e+00	2.017157e+01	1.591526e+01	4.549269e+01	2.587662e+01	5.260831e+00	7.641207e+00	1.176823e+01	**2.006736e+00**	2.984909e+00
	Worst	2.367848e+04	2.454340e+04	1.062454e+04	3.585412e+04	**6.653583e+02**	8.233082e+02	9.452118e+03	3.702020e+03	1.549900e+04	2.538942e+03
	Mean	4.839696e+03	4.939014e+03	9.662131e+02	9.425729e+03	**9.954312e+01**	2.777381e+02	2.203366e+03	2.509886e+02	2.783519e+03	2.840139e+02
	S.D	6.092895e+03	6.595500e+03	1.777469e+03	9.719199e+03	**1.243285e+02**	3.211765e+02	2.632851e+03	8.677276e+02	3.563161e+03	4.730585e+02
	Sign	+	+	+	+	=	=	+	=	+	
f15	**Best**	2.362479e+00	4.995001e+00	6.268417e+01	3.059824e+02	2.843988e+01	2.928987e+00	5.178687e+00	1.286161e+00	7.467409e+00	3.693794e+00
	Worst	2.976979e+04	2.763056e+04	7.413656e+03	1.311172e+05	**2.226454e+03**	3.687927e+03	1.580716e+04	2.730113e+03	2.777639e+04	5.235767e+03
	Mean	5.954015e+03	3.744469e+03	1.458547e+03	1.903567e+04	6.094275e+02	4.914274e+02	2.916917e+03	**2.901107e+02**	4.989933e+03	5.616528e+02
	S.D	8.069506e+03	5.649089e+03	1.604238e+03	2.706073e+04	**5.923967e+02**	9.630818e+02	4.153403e+03	7.372694e+02	6.074062e+03	1.120848e+03
	Sign	+	+	+	+	+	=	+	=	+	
f16	**Best**	**2.541329e-01**	3.164234e-01	1.639615e+00	1.845154e+00	1.352810e+01	1.830981e+00	8.142376e-01	5.358528e-01	7.842936e-01	7.463902e-01
	Worst	3.345075e+02	2.852645e+02	3.580886e+02	6.150163e+02	**2.583912e+02**	6.093790e+02	5.786441e+02	4.703827e+02	5.915584e+02	4.364763e+02
	Mean	1.257393e+02	1.070517e+02	1.405679e+02	2.358085e+02	**1.007282e+02**	3.011782e+02	2.437574e+02	1.492556e+02	2.077398e+02	1.456230e+02
	S.D	9.668993e+01	9.435593e+01	9.113570e+01	1.629663e+02	**6.584349e+01**	1.536741e+02	1.479785e+02	1.354875e+02	1.426790e+02	1.272904e+02
	Sign	=	=	=	+	=	+	+	=	=	
f17	**Best**	**3.320030e-01**	3.911377e-01	2.978491e+00	3.201369e+00	1.311780e+01	1.416783e+00	1.172981e+00	1.367622e+00	9.951069e-01	4.021547e-01
	Worst	6.031056e+01	6.028100e+01	**4.463773e+01**	2.501341e+02	9.876378e+01	1.752124e+02	2.020685e+02	1.797565e+02	2.181574e+02	2.490454e+02
	Mean	1.221460e+01	**1.029418e+01**	1.629891e+01	6.605568e+01	5.434306e+01	6.366542e+01	3.700199e+01	4.357473e+01	3.748082e+01	4.566916e+01
	S.D	1.455293e+01	1.198147e+01	**9.439348e+00**	6.447679e+01	1.644955e+01	5.310691e+01	4.621642e+01	4.268406e+01	4.554293e+01	5.111614e+01
	Sign	-	-	-	+	+	+	=	=	=	
f18	**Best**	2.272647e+02	6.951372e+01	3.029703e+02	4.153996e+03	7.474113e+02	2.339963e+02	7.188422e+02	1.564564e+02	**5.309198e+01**	2.653506e+02
	Worst	**2.754066e+04**	3.860597e+04	3.934172e+04	1.797709e+05	4.808587e+05	4.181582e+04	4.003101e+04	3.323429e+04	3.860454e+04	4.367422e+04
	Mean	7.867905e+03	9.384904e+03	**6.834995e+03**	3.463413e+04	4.288083e+04	1.306725e+04	1.054927e+04	9.882279e+03	8.273557e+03	1.238224e+04
	S.D	**7.187901e+03**	9.036664e+03	7.345534e+03	3.645903e+04	7.480058e+04	1.188432e+04	8.907767e+03	9.249265e+03	8.882218e+03	1.102783e+04
	Sign	-	=	-	+	+	=	=	=	-	
f19	**Best**	4.642944e+00	1.293349e+01	1.048657e+02	1.189632e+02	2.259318e+01	1.243516e+00	2.704403e+00	1.394865e+00	**1.055701e+00**	1.155281e+00
	Worst	3.059745e+04	2.833526e+04	2.324152e+04	2.168882e+05	8.745641e+03	4.630171e+03	1.344110e+04	**1.323775e+02**	2.676996e+04	1.173028e+04
	Mean	6.319353e+03	7.108882e+03	5.163962e+03	2.045474e+04	9.538275e+02	6.030789e+02	2.701266e+03	**1.844286e+01**	7.252991e+03	1.408474e+03
	S.D	8.290822e+03	7.895838e+03	5.807591e+03	3.871574e+04	1.643024e+03	1.066022e+03	3.814725e+03	**2.171638e+01**	7.663953e+03	2.625247e+03
	Sign	+	+	+	+	=	=	+	-	+	
f20	**Best**	5.166554e-04	3.698797e-04	4.272918e+00	3.581341e+00	2.465218e+01	1.348301e+00	3.381970e-01	7.205935e-04	6.243557e-01	**3.805491e-05**
	Worst	1.714840e+01	**1.447564e+01**	3.080782e+01	4.010812e+01	8.648894e+01	3.825578e+01	1.597790e+02	7.364907e+01	3.646542e+01	3.746039e+01
	Mean	**4.335820e+00**	5.116853e+00	1.276152e+01	1.782113e+01	3.889704e+01	1.771273e+01	3.881888e+01	2.319750e+01	1.007943e+01	1.184889e+01
	S.D	**5.576009e+00**	5.644992e+00	5.618087e+00	1.119001e+01	1.182253e+01	9.924064e+00	5.466651e+01	1.867874e+01	9.800866e+00	9.608733e+00
	Sign	-	-	=	+	+	+	=	+	=	
Composition											
f21	**Best**	2.067944e+02	**1.000000e+02**	1.043504e+02	1.154528e+02	1.104733e+02	**1.000000e+02**	**1.000000e+02**	1.000003e+02	1.005987e+02	**1.000000e+02**
	Worst	2.342539e+02	2.428625e+02	2.323032e+02	2.673848e+02	**1.429466e+02**	2.907360e+02	2.513960e+02	2.493767e+02	2.589281e+02	2.268882e+02
	Mean	2.204059e+02	2.221464e+02	1.607315e+02	2.236009e+02	**1.224739e+02**	2.285995e+02	2.220663e+02	2.136245e+02	1.796870e+02	2.037605e+02
	S.D	**6.826088e+00**	1.928818e+01	3.710807e+01	3.640977e+01	7.154389e+00	4.565144e+01	1.947819e+01	3.910208e+01	6.449375e+01	3.494594e+01
	Sign	+	+	-	+	-	+	+	+	=	
f22	**Best**	2.176933e+01	1.004507e+02	6.691402e+01	3.112040e+01	6.198010e+01	5.031448e+01	3.891707e+01	**1.556050e+01**	3.727728e+01	1.003975e+02
	Worst	1.297648e+03	1.213393e+03	**2.505180e+02**	1.594494e+03	3.876698e+02	1.645969e+03	1.207363e+03	1.311257e+03	1.006951e+03	1.028743e+03
	Mean	2.351471e+02	2.478073e+02	**1.142807e+02**	4.489051e+02	1.491667e+02	5.155292e+02	2.449528e+02	4.106884e+02	1.182894e+02	1.908696e+02
	S.D	2.861509e+02	2.814107e+02	**2.313316e+01**	5.040070e+02	5.355814e+01	5.310350e+02	3.274635e+02	4.085998e+02	1.279177e+02	2.200245e+02
	Sign	+	+	+	+	+	+	=	+	+	
f23	**Best**	3.083269e+02	3.076699e+02	3.088413e+02	3.085941e+02	3.342033e+02	3.217319e+02	3.061775e+02	**9.625253e-04**	3.082473e+02	3.056937e+02
	Worst	**3.345458e+02**	4.304798e+02	3.399421e+02	3.718088e+02	3.807420e+02	4.573325e+02	3.709680e+02	4.996744e+02	3.541623e+02	3.708412e+02
	Mean	3.203277e+02	3.234833e+02	3.201275e+02	3.273862e+02	3.584957e+02	3.459125e+02	3.276237e+02	3.361594e+02	3.263352e+02	**3.164990e+02**
	S.D	**6.508120e+00**	1.760892e+01	7.205896e+00	1.144811e+01	1.002045e+01	2.353420e+01	1.440758e+01	6.028015e+01	1.097577e+01	9.864801e+00
	Sign	+	+	+	+	+	+	+	+	+	
f24	**Best**	3.376661e+02	1.000001e+02	1.073175e+02	1.043237e+02	1.653296e+02	1.000003e+02	1.000000e+02	1.000007e+02	1.000000e+02	**1.198313e-05**
	Worst	3.925708e+02	3.982222e+02	3.819759e+02	4.439687e+02	**3.037314e+02**	4.838104e+02	4.144178e+02	4.315432e+02	4.316600e+02	3.583709e+02
	Mean	3.608635e+02	3.529243e+02	3.307888e+02	3.441853e+02	**2.283091e+02**	3.573751e+02	3.371882e+02	3.436669e+02	3.365685e+02	3.102579e+02
	S.D	1.229401e+01	4.433812e+01	5.440979e+01	7.428948e+01	**3.024435e+01**	8.833427e+01	8.080053e+01	7.197477e+01	9.215116e+01	8.257980e+01
	Sign	+	+	+	+	-	+	+	+	+	
f25	**Best**	3.000019e+02	3.977429e+02	3.994313e+02	1.185821e+02	4.491497e+02	**1.000265e+02**	3.977429e+02	1.008467e+02	2.000056e+02	1.000282e+02
	Worst	4.711312e+02	4.712434e+02	4.752317e+02	5.258750e+02	5.477118e+02	5.241938e+02	5.241724e+02	4.516909e+02	4.552735e+02	**4.459093e+02**
	Mean	4.335897e+02	4.279410e+02	4.415062e+02	4.323535e+02	4.946944e+02	4.157021e+02	4.296184e+02	4.147006e+02	4.267289e+02	**4.092888e+02**
	S.D	2.902392e+01	2.503457e+01	**2.007651e+01**	5.138722e+01	2.197782e+01	6.436069e+01	2.578747e+01	6.785871e+01	4.365632e+01	8.018491e+01
	Sign	+	+	+	+	+	=	=	+	+	
f26	**Best**	2.000007e+02	2.000009e+02	5.383076e+01	2.186716e+02	4.178263e+02	1.783303e-03	2.000000e+02	2.000147e+02	**8.774045e-07**	4.938015e-06
	Worst	1.596291e+03	1.551530e+03	9.809798e+02	1.755940e+03	**6.887522e+02**	1.966142e+03	1.879450e+03	1.685680e+03	1.817107e+03	1.478170e+03
	Mean	6.463739e+02	7.970879e+02	**4.124443e+02**	1.010524e+03	5.306655e+02	8.509547e+02	7.449923e+02	7.252227e+02	5.675303e+02	4.776900e+02
	S.D	4.777589e+02	5.263505e+02	1.788626e+02	5.892121e+02	**6.635872e+01**	6.005920e+02	5.555540e+02	4.807898e+02	4.552821e+02	4.647395e+02
	Sign	+	+	+	+	+	+	+	+	+	

f27	**Best**	3.896647e+02	**3.885711e+02**	3.932700e+02	3.938429e+02	4.061458e+02	3.897481e+02	3.893081e+02	3.889780e+02	3.925177e+02	3.893081e+02
	Worst	4.795512e+02	4.779539e+02	**4.163890e+02**	5.085085e+02	4.355955e+02	4.907679e+02	5.367236e+02	5.006186e+02	4.830652e+02	4.841565e+02
	Mean	4.053792e+02	4.035581e+02	4.007185e+02	4.382607e+02	4.191231e+02	4.148954e+02	4.152634e+02	4.143950e+02	4.164982e+02	**4.005426e+02**
	S.D	2.272626e+01	1.745157e+01	**4.947542e+00**	3.288837e+01	6.501279e+00	2.158262e+01	3.110578e+01	2.671315e+01	1.877153e+01	1.889401e+01
	Sign	+	+	+	+	+	+	+	+	+	
f28	**Best**	3.000002e+02	3.000001e+02	3.073059e+02	3.035733e+02	4.026805e+02	**3.186813e-03**	3.000000e+02	3.000064e+02	3.000000e+02	3.000000e+02
	Worst	6.118218e+02	6.118218e+02	**5.340349e+02**	6.503661e+02	5.551039e+02	9.361800e+02	6.191938e+02	6.118218e+02	6.464807e+02	9.318129e+02
	Mean	4.619498e+02	4.338356e+02	**4.255395e+02**	5.525978e+02	4.620535e+02	4.997942e+02	4.955555e+02	5.137916e+02	4.987382e+02	4.844802e+02
	S.D	1.199558e+02	1.246972e+02	4.349595e+01	1.067475e+02	**3.391257e+01**	1.792245e+02	1.452375e+02	1.232419e+02	1.349693e+02	1.601692e+02
	Sign	=	=	=	+	=	=	=	=	=	
f29	**Best**	2.380839e+02	**2.344467e+02**	2.541729e+02	2.458806e+02	3.176633e+02	2.408711e+02	2.359359e+02	2.555260e+02	2.487351e+02	2.390142e+02
	Worst	3.868111e+02	4.192845e+02	**3.311887e+02**	5.676344e+02	4.292719e+02	5.726521e+02	4.623724e+02	4.126028e+02	4.953399e+02	4.508391e+02
	Mean	2.764155e+02	**2.745949e+02**	2.863860e+02	3.671054e+02	3.673052e+02	3.421441e+02	3.031212e+02	3.128958e+02	3.497894e+02	3.037306e+02
	S.D	3.703342e+01	3.398194e+01	**2.118599e+01**	7.454593e+01	2.670549e+01	6.534007e+01	4.866282e+01	4.605503e+01	6.475238e+01	5.104503e+01
	Sign	-	-	=	+	+	+	=	=	+	
f 30	**Best**	3.454951e+03	2.019294e+03	1.096664e+04	2.212695e+03	1.162666e+05	2.218289e+03	**1.009529e+03**	2.508465e+03	1.744197e+04	2.768918e+03
	Worst	1.237956e+06	1.381897e+06	1.131803e+06	4.652546e+06	3.922495e+06	1.242301e+06	1.243666e+06	8.177**270e+05**	1.669593e+06	9.062202e+05
	Mean	1.388747e+05	1.517530e+05	3.790866e+05	1.370650e+06	1.532950e+06	3.118512e+05	1.379480e+05	1.381611e+05	4.567301e+05	**1.148024e+05**
	S.D	**2.411477e+05**	3.374226e+05	3.255099e+05	1.211658e+06	9.374478e+05	4.369668e+05	3.640801e+05	2.961280e+05	5.202039e+05	2.832147e+05
	Sign	+	+	+	+	+	=	=	=	+	

Sign	BWM_HS VS.	HS	IHS	GHS	NGHS	SGHS	IGHS	GDHS	INGHS	LHS
+/=/-		20/5/5	19/7/4	21/5/4	30/0/0	23/5/2	20/9/1	15/13/2	19/10/1	21/8/1

Table 5.
Results obtained for 51 independent runs of BWM_HS and eight recent variants of HS on 30D benchmarks of CEC2017.

No.	Algorithms	HS	IHS	GHS	NGHS	SGHS	IGHS	GDHS	INGHS	LHS	BWM_HS
Unimodal											
f1	**Best**	4.236809e+01	1.219356e+00	1.013624e+09	1.546400e+05	2.229795e+10	1.719633e+01	9.571351e+00	1.401856e+06	1.176299e+00	**1.137640e+00**
	Worst	1.954060e+04	1.978398e+04	2.702650e+09	1.250115e+06	4.787620e+10	**1.652060e+04**	1.902295e+04	2.503806e+08	2.009841e+04	2.058761e+04
	Mean	6.448399e+03	5.336775e+03	1.661671e+09	4.715276e+05	3.495969e+10	**4.329417e+03**	5.131995e+03	3.077070e+07	5.816497e+03	4.513913e+03
	S.D	6.678902e+03	6.342381e+03	3.438598e+08	2.405515e+05	5.531439e+09	4.201**424e+03**	5.399440e+03	4.614425e+07	6.481670e+03	5.430401e+03
	Sign	=	=	+	+	+	=	=	+	=	
f 2	**Best**	4.443557e+06	3.291280e+03	3.969758e+22	2.650437e+09	1.587902e+34	6.876615e-06	1.965472e-06	2.021600e+12	6.754219e-03	**8.991043e-08**
	Worst	1.157376e+14	1.524729e+12	2.255818e+29	1.500268e+17	4.973477e+39	6.679027e-05	6.799511e-05	7.692013e+17	1.459263e+11	**2.706832e-05**
	Mean	8.598341e+12	1.258220e+11	1.097444e+28	3.949950e+15	2.475883e+38	3.226231e-05	1.562992e-05	8.375166e+16	6.454045e+09	**8.042088e-06**
	S.D	2.116097e+13	3.662799e+11	3.716127e+28	2.201569e+16	7.624945e+38	1.659165e-05	1.070917e-05	1.648545e+17	2.431189e+10	**7.066708e-06**
	Sign	+	+	+	+	+	+	+	+	+	
f3	**Best**	2.327006e+03	2.319748e+03	1.557264e+04	6.699253e+04	6.237485e+04	4.558976e-07	3.182930e-07	1.315985e+03	1.142432e+03	**2.483074e-08**
	Worst	1.262510e+04	1.338621e+04	5.815884e+04	2.590003e+05	1.179389e+05	1.132988e-06	1.444978e-06	1.911015e+04	1.994965e+04	**4.108418e-07**
	Mean	6.709435e+03	5.393217e+03	3.931177e+04	1.565862e+05	9.687237e+04	8.238018e-07	7.235261e-07	6.958634e+03	8.441260e+03	**1.108154e-07**
	S.D	2.397073e+03	2.310029e+03	7.825217e+03	4.147428e+04	1.182681e+04	1.541259e-07	2.619424e-07	4.290256e+03	3.847008e+03	**6.523261e-08**
	Sign	+	+	+	+	+	+	+	+	+	
Multimodal											
f4	**Best**	3.868611e+00	4.460026e+00	2.059191e+02	3.212553e+01	3.134827e+03	6.408004e-04	1.877868e-05	6.932709e+01	4.015356e+00	**2.523960e-04**
	Worst	1.747443e+02	1.450612e+02	4.945932e+02	2.431692e+02	7.607896e+03	1.077416e+02	8.771925e+01	1.350156e+02	1.334073e+02	**8.729289e+01**
	Mean	9.951820e+01	9.000422e+01	3.276374e+02	1.108127e+02	5.188293e+03	4.968612e+01	**2.556090e+01**	9.716265e+01	8.882779e+01	6.695919e+01
	S.D	2.784939e+01	2.720407e+01	5.934668e+01	3.768907e+01	1.051710e+03	3.719221e+01	3.297872e+01	**2.003480e+01**	2.628050e+01	2.719478e+01
	Sign	+	+	+	+	+	=	-	+	+	
f 5	**Best**	3.681347e+01	2.785884e+01	1.294784e+02	6.107235e+01	3.243005e+02	6.765709e+01	2.990242e+01	6.892744e+01	5.870249e+01	**2.586891e+01**
	Worst	1.183996e+02	1.152570e+02	2.261582e+02	1.692992e+02	4.205599e+02	2.440489e+02	1.164097e+02	2.452542e+02	1.731212e+02	**9.452041e+01**
	Mean	6.885706e+01	7.328210e+01	1.763465e+02	1.076656e+02	3.709806e+02	1.300970e+02	6.528494e+01	1.367651e+02	1.052443e+02	**4.917634e+01**
	S.D	1.956553e+01	1.635412e+01	2.349805e+01	2.791390e+01	2.445217e+01	3.838038e+01	1.780578e+01	3.598260e+01	2.660624e+01	**1.560829e+01**
	Sign	+	+	+	+	+	+	+	+	+	
f6	**Best**	1.554385e-01	3.350526e-04	8.459178e+00	1.061187e-01	5.756514e+01	1.095879e-02	4.243312e-05	3.757038e+00	9.708082e-05	**4.097135e-07**
	Worst	4.344553e-01	1.753995e-01	1.809129e+01	4.762075e-01	8.163935e+01	4.381987e-01	1.073383e-02	3.173744e+01	1.437361e-02	**2.075697e-04**
	Mean	2.817909e-01	2.876857e-02	1.272469e+01	2.550378e-01	6.850655e+01	1.293353e-01	3.824002e-04	1.408876e+01	1.858704e-03	**1.073743e-05**
	S.D	7.500920e-02	3.592118e-02	2.506558e+00	8.807362e-02	5.387848e+00	1.042491e-01	1.493890e-03	6.099292e+00	2.921927e-03	**3.292264e-05**
	Sign	+	+	+	+	+	+	+	+	+	
f7	**Best**	8.185264e+01	6.532883e+01	2.023314e+02	9.582492e+01	6.416589e+02	9.141931e+01	6.492893e+01	1.546309e+02	7.022009e+01	**5.359835e+01**
	Worst	1.957348e+02	1.921422e+02	3.468748e+02	2.082774e+02	1.296262e+03	2.479651e+02	1.327640e+02	3.460756e+02	1.858349e+02	**9.229885e+01**
	Mean	1.257658e+02	1.201791e+02	2.743743e+02	1.339770e+02	8.961189e+02	1.335671e+02	9.152127e+01	2.665942e+02	1.336654e+02	**6.843604e+01**
	S.D	2.610724e+01	2.442725e+01	2.675929e+01	2.250302e+01	1.489414e+02	3.191490e+01	1.723808e+01	4.827418e+01	2.200325e+01	**9.789323e+00**
	Sign	+	+	+	+	+	+	+	+	+	
f8	**Best**	4.299226e+01	3.984450e+01	1.110935e+02	6.993945e+01	2.870734e+02	8.158648e+01	4.377816e+01	6.441288e+01	5.273276e+01	**3.084369e+01**
	Worst	1.333241e+02	1.164096e+02	2.211931e+02	1.784807e+02	3.727821e+02	2.178946e+02	1.104401e+02	2.137776e+02	1.741166e+02	**8.855112e+01**
	Mean	7.390628e+01	7.418079e+01	1.628021e+02	1.134132e+02	3.368227e+02	1.378779e+02	7.448250e+01	1.162727e+02	1.069075e+02	**5.190076e+01**
	S.D	1.752415e+01	1.611934e+01	2.352678e+01	2.170933e+01	2.072289e+01	3.177498e+01	1.635167e+01	3.295288e+01	2.560726e+01	**1.253997e+01**
	Sign	+	+	+	+	+	+	+	+	+	
f9	**Best**	1.893523e+01	2.578365e+01	3.218906e+02	1.379376e+02	7.527901e+03	5.963932e+02	1.438138e-09	7.545731e+02	1.315682e+02	**1.273293e-11**
	Worst	1.381133e+03	1.246741e+03	2.499167e+03	4.823662e+03	1.428079e+04	8.306213e+03	8.228657e+02	7.946513e+03	3.608643e+03	**5.130854e+02**
	Mean	3.230810e+02	3.012317e+02	1.285284e+03	1.608842e+03	1.050190e+04	3.012875e+03	1.123111e+02	3.669041e+03	1.251654e+03	**1.014769e+01**
	S.D	2.691886e+02	2.228410e+02	4.531574e+02	9.734456e+02	1.547383e+03	1.700487e+03	1.410477e+02	1.784742e+03	9.352343e+02	**7.183408e+01**

	Sign	+	+	+	+	+	+	+	+	+	
f10	**Best**	1.386010e+03	1.174008e+03	4.354226e+03	1.839152e+03	6.013958e+03	2.106283e+03	1.230489e+03	1.663656e+03	1.525264e+03	**1.074806e+03**
	Worst	3.382533e+03	3.632996e+03	6.814467e+03	4.072850e+03	7.704309e+03	4.454816e+03	3.443315e+03	6.815034e+03	3.410701e+03	**2.781166e+03**
	Mean	2.191271e+03	2.276985e+03	5.628381e+03	2.750209e+03	7.188523e+03	3.007024e+03	2.255618e+03	3.713033e+03	2.618339e+03	**2.036321e+03**
	S.D	4.696116e+02	4.904001e+02	6.279806e+02	5.223243e+02	**3.431511e+02**	5.915340e+02	4.889527e+02	1.440964e+03	4.204094e+02	4.435341e+02
	Sign	=	+	+	+	+	+	+	+	+	
	Hybrid										
f11	**Best**	2.757983e+01	**1.765748e+01**	2.313523e+02	4.430669e+02	1.777635e+03	2.497933e+01	2.037896e+01	5.078190e+01	2.605155e+01	2.730753e+01
	Worst	4.077907e+02	**1.158222e+02**	1.137636e+03	2.283046e+04	6.162842e+03	2.030228e+02	1.256467e+02	2.197828e+02	1.455240e+02	2.073235e+02
	Mean	1.254202e+02	**5.574552e+01**	5.179095e+02	6.809226e+03	3.721254e+03	1.069801e+02	6.053198e+01	1.198489e+02	7.296190e+01	9.470615e+01
	S.D	6.752067e+01	3.296005e+01	2.243406e+02	5.558441e+03	9.181103e+02	4.569505e+01	3.180521e+01	3.501086e+01	**2.853538e+01**	4.052064e+01
	Sign	+	-	+	+	+	=	-	+	-	
f12	**Best**	1.559278e+05	2.477886e+05	1.128065e+07	4.481614e+05	1.802906e+09	3.995566e+04	**1.400254e+04**	6.160288e+04	9.158521e+04	3.408666e+04
	Worst	9.941211e+06	5.176129e+06	1.041065e+08	1.800405e+07	5.088342e+09	1.444322e+06	**1.318150e+06**	5.684238e+06	5.738161e+06	2.182133e+06
	Mean	2.551935e+06	1.553026e+06	4.055255e+07	5.916245e+06	3.245245e+09	**3.168162e+05**	4.034503e+05	1.790434e+06	1.896584e+06	4.733276e+05
	S.D	2.006031e+06	1.183266e+06	1.704197e+07	3.992994e+06	7.345285e+08	**2.377493e+05**	3.284422e+05	1.238139e+06	1.429446e+06	4.659493e+05
	Sign	+	+	+	+	+	=	=	+	+	
f13	**Best**	1.418254e+02	1.456690e+02	1.560403e+05	6.356403e+04	1.416437e+08	1.243393e+02	2.089892e+02	2.792524e+02	**9.538797e+01**	3.338342e+02
	Worst	6.120333e+04	6.116425e+04	7.835101e+06	4.571085e+06	1.497623e+09	**6.082990e+04**	6.097981e+04	6.969292e+04	6.562905e+04	6.108563e+04
	Mean	**1.716745e+04**	2.140201e+04	9.311834e+05	4.001253e+05	6.239103e+08	2.098894e+04	2.023596e+04	2.372591e+04	1.820556e+04	1.777048e+04
	S.D	1.838118e+04	2.265145e+04	1.296248e+06	6.906053e+05	3.230489e+08	2.057152e+04	1.844461e+04	2.036846e+04	2.086745e+04	**1.628363e+04**
	Sign	=	=	+	+	+	=	=	=	=	
f14	**Best**	2.648620e+03	4.446684e+02	3.397732e+04	5.234875e+04	1.659095e+04	**9.468360e+01**	9.886223e+01	5.431395e+03	1.186095e+04	2.247579e+02
	Worst	1.151265e+06	6.887341e+04	1.013981e+06	1.379717e+07	8.897066e+05	1.236011e+04	**8.297826e+03**	1.352503e+05	2.506250e+06	1.446644e+04
	Mean	1.755387e+05	2.832742e+04	3.815771e+05	3.308079e+06	3.172265e+05	2.639183e+03	**2.078369e+03**	2.868012e+04	6.828898e+05	3.651831e+03
	S.D	2.019404e+05	2.155142e+04	2.386883e+05	3.637179e+06	2.071323e+05	2.746748e+03	2.244527e+03	2.024329e+04	6.012361e+05	3.218541e+03
	Sign	+	+	+	+	+	-	-	+	+	
f15	**Best**	4.501798e+01	**2.334649e+01**	2.572939e+03	2.529903e+03	5.663558e+05	1.172214e+02	3.285391e+01	6.903861e+01	3.601390e+01	4.392985e+01
	Worst	4.127221e+04	4.125195e+04	1.844747e+05	5.573334e+05	7.687850e+07	4.057567e+04	4.082433e+04	4.146944e+04	**4.021769e+04**	4.037165e+04
	Mean	9.376910e+03	9.075286e+03	5.845141e+04	1.158983e+05	1.543916e+07	1.062104e+04	9.083066e+03	9.423522e+03	**6.167206e+03**	7.892827e+03
	S.D	1.042728e+04	9.705512e+03	3.377444e+04	1.106819e+05	1.983477e+07	1.019031e+04	1.065486e+04	1.210352e+04	**8.443040e+03**	9.582214e+03
	Sign	=	=	+	+	+	=	=	=	=	
f16	**Best**	2.593453e+02	1.675640e+02	3.861448e+02	7.500021e+02	1.782076e+03	5.413977e+02	3.033302e+02	2.805306e+02	5.758154e+02	**4.437523e+01**
	Worst	1.436216e+03	1.682241e+03	1.570833e+03	1.886634e+03	2.859840e+03	1.581702e+03	1.435282e+03	1.437780e+03	1.836778e+03	**1.243529e+03**
	Mean	9.306407e+02	9.661679e+02	1.124965e+03	1.286422e+03	2.460140e+03	9.531679e+02	9.806120e+02	8.110248e+02	1.251662e+03	**5.176936e+02**
	S.D	2.765961e+02	2.976304e+02	2.811679e+02	2.940553e+02	**2.257846e+02**	2.551172e+02	2.802868e+02	2.977058e+02	3.157133e+02	2.444367e+02
	Sign	+	+	+	+	+	+	+	+	+	
f17	**Best**	**3.349389e+01**	4.990722e+01	8.146179e+01	7.015078e+01	7.000758e+02	2.498572e+02	1.822846e+02	5.564503e+01	8.214475e+01	7.601903e+01
	Worst	8.074291e+02	7.573494e+02	8.184304e+02	1.071834e+03	1.364629e+03	1.069805e+03	8.932697e+02	8.676128e+02	1.038280e+03	**6.261483e+02**
	Mean	4.041487e+02	4.442122e+02	4.336549e+02	6.372835e+02	1.073661e+03	6.530212e+02	5.037432e+02	3.896303e+02	5.833362e+02	**2.527556e+02**
	S.D	2.099313e+02	1.720169e+02	1.764034e+02	2.577616e+02	1.513972e+02	2.009261e+02	1.816092e+02	2.009518e+02	2.064126e+02	**1.175577e+02**
	Sign	+	+	+	+	+	+	+	+	+	
f18	**Best**	2.519171e+04	1.447570e+04	1.701401e+05	1.859514e+05	1.106079e+06	2.081589e+04	**1.418522e+04**	4.193492e+04	3.308784e+04	3.088925e+04
	Worst	1.769495e+06	1.122938e+06	3.769122e+06	1.423423e+07	1.361860e+07	**2.297708e+05**	4.101145e+05	1.334922e+06	1.111735e+07	2.736128e+05
	Mean	2.968182e+05	1.966361e+05	1.200623e+06	3.363963e+06	7.771951e+06	**1.147283e+05**	1.324211e+05	3.114889e+05	1.644392e+06	1.216823e+05
	S.D	3.511651e+05	2.267360e+05	8.109171e+05	3.360989e+06	2.829843e+06	**5.541521e+04**	8.594907e+04	2.662198e+05	2.064417e+06	6.212788e+04
	Sign	+	=	+	+	+	=	=	+	+	
f19	**Best**	2.325888e+01	**1.220782e+01**	1.912576e+04	2.052424e+03	2.448382e+06	2.416815e+01	5.768410e+01	1.047427e+02	3.204384e+01	1.985905e+01
	Worst	4.822431e+04	**3.886758e+04**	3.954014e+05	4.559988e+05	2.142363e+08	5.347899e+04	4.735995e+04	1.025910e+05	4.591135e+04	4.920867e+04
	Mean	9.803636e+03	1.002244e+04	1.296321e+05	8.950129e+04	4.672978e+07	1.026675e+04	1.090113e+04	1.488600e+04	**7.651925e+03**	8.549565e+03
	S.D	1.232546e+04	**1.002267e+04**	9.168949e+04	8.557438e+04	5.072928e+07	1.272294e+04	1.237261e+04	2.175421e+04	1.112359e+04	1.235625e+04
	Sign	=	=	+	+	+	=	=	+	=	
f20	**Best**	**1.500582e+01**	4.336638e+01	1.844728e+02	5.249199e+01	4.307136e+02	1.616887e+02	3.835801e+01	6.604171e+01	4.580061e+01	5.189996e+01
	Worst	7.836213e+02	8.206901e+02	7.819776e+02	1.245603e+03	9.725666e+02	1.024268e+03	9.489881e+02	9.584908e+02	9.923323e+02	**5.108359e+02**
	Mean	4.283788e+02	4.040501e+02	4.739274e+02	7.009872e+02	7.633449e+02	6.146498e+02	5.193360e+02	4.137974e+02	5.183916e+02	**2.722181e+02**
	S.D	1.615816e+02	1.689158e+02	1.581386e+02	2.272086e+02	1.044610e+02	1.865788e+02	2.267959e+02	1.772346e+02	2.152265e+02	**9.808947e+01**
	Sign	+	+	+	+	+	+	+	+	+	
	Composition										
f21	**Best**	2.386011e+02	2.457920e+02	3.303537e+02	2.687297e+02	4.936582e+02	2.925349e+02	2.448944e+02	**1.117243e+02**	2.597692e+02	2.353862e+02
	Worst	3.450628e+02	3.135402e+02	4.254983e+02	3.776795e+02	5.817628e+02	4.091117e+02	3.308570e+02	3.837963e+02	3.897915e+02	**2.973876e+02**
	Mean	2.742120e+02	2.798433e+02	3.768156e+02	3.117919e+02	5.454260e+02	3.440128e+02	2.765827e+02	3.010218e+02	3.106135e+02	**2.576968e+02**
	S.D	1.889273e+01	1.673275e+01	2.299325e+01	2.592722e+01	1.799052e+01	3.385022e+01	1.947028e+01	5.245534e+01	2.940658e+01	**1.397393e+01**
	Sign	+	+	+	+	+	+	+	+	+	
f22	**Best**	1.000125e+02	1.000008e+02	3.248536e+02	6.884471e+01	2.771147e+03	1.000015e+02	1.000001e+02	1.115647e+02	1.000003e+02	**1.000000e+02**
	Worst	**3.688284e+03**	4.072407e+03	7.065743e+03	4.464752e+03	5.276137e+03	4.811655e+03	3.845859e+03	7.321033e+03	4.340986e+03	3.798410e+03
	Mean	2.267513e+03	2.202985e+03	3.732092e+03	2.941187e+03	4.046349e+03	3.317582e+03	2.259326e+03	2.615136e+03	1.758871e+03	**1.692878e+03**
	S.D	9.676562e+02	1.185833e+03	2.465440e+03	1.322338e+03	**4.659030e+02**	8.539029e+02	1.229027e+03	2.086716e+03	1.741456e+03	1.162270e+03
	Sign	+	+	+	+	+	+	+	+	=	
f23	**Best**	3.928507e+02	4.047527e+02	4.920643e+02	4.107259e+02	7.471055e+02	4.223618e+02	3.886155e+02	1.771926e+02	3.679767e+02	**1.000001e+02**
	Worst	4.968396e+02	4.792131e+02	5.898904e+02	5.536827e+02	8.985546e+02	5.567154e+02	4.954721e+02	6.167156e+02	5.451907e+02	**4.317421e+02**
	Mean	4.291926e+02	4.372138e+02	5.411037e+02	4.641449e+02	8.315133e+02	4.861027e+02	4.312387e+02	4.910643e+02	4.690260e+02	**3.965493e+02**
	S.D	2.184466e+01	**1.854369e+01**	2.422805e+01	2.939077e+01	3.382096e+01	3.285209e+01	1.990219e+01	7.835414e+01	3.764880e+01	4.507822e+01
	Sign	+	+	+	+	+	+	+	+	+	
f24	**Best**	4.553356e+02	4.787256e+02	6.217055e+02	5.677127e+02	8.052225e+02	5.472772e+02	4.612227e+02	5.020840e+02	5.583417e+02	**4.398118e+02**
	Worst	5.741217e+02	6.147865e+02	6.887189e+02	8.369366e+02	9.937870e+02	9.753651e+02	5.586588e+02	7.010050e+02	9.130056e+02	**5.160277e+02**
	Mean	5.102718e+02	5.084109e+02	6.656945e+02	6.776122e+02	9.005724e+02	6.757087e+02	5.018595e+02	5.826874e+02	7.082190e+02	**4.756321e+02**

	S.D	2.869222e+01	2.592817e+01	**1.443752e+01**	7.339034e+01	4.396126e+01	7.396617e+01	2.273908e+01	5.206270e+01	8.160720e+01	1.516629e+01
	Sign	+	+	+	+	+	+	+	+	+	
	Best	3.848567e+02	3.834842e+02	4.621075e+02	3.846522e+02	1.420473e+03	3.834165e+02	**3.833883e+02**	3.840115e+02	3.838492e+02	3.833991e+02
f 25	**Worst**	4.720075e+02	4.321815e+02	6.646840e+02	4.675801e+02	3.803898e+03	4.440336e+02	**3.869253e+02**	4.483543e+02	4.416807e+02	3.982881e+02
	Mean	4.026339e+02	3.908030e+02	5.389683e+02	4.030424e+02	2.439382e+03	3.876260e+02	**3.862127e+02**	4.056125e+02	3.990629e+02	3.874169e+02
	S.D	2.194895e+01	1.047057e+01	5.088068e+01	1.923970e+01	5.268044e+02	8.477466e+00	**1.306206e+00**	1.696380e+01	1.779362e+01	2.707715e+00
	Sign	+	+	+	+	+	=	=	+	+	
	Best	3.000318e+02	2.000052e+02	1.096000e+03	2.110670e+02	5.046335e+03	2.000093e+02	2.000013e+02	3.555955e+02	2.000025e+02	**2.000001e+02**
f 26	**Worst**	4.068023e+03	**2.635234e+03**	3.629624e+03	4.410210e+03	6.860560e+03	3.773164e+03	2.778501e+03	3.274194e+03	3.914351e+03	2.731061e+03
	Mean	2.005216e+03	1.866901e+03	2.861615e+03	2.218594e+03	6.051304e+03	2.349173e+03	1.958574e+03	2.145201e+03	2.428106e+03	**1.232647e+03**
	S.D	4.544384e+02	5.327811e+02	**4.373042e+02**	8.382594e+02	4.376500e+02	8.679169e+02	5.133970e+02	7.751345e+02	9.038548e+02	6.427768e+02
	Sign	+	+	+	+	+	+	+	+	+	
	Best	5.029758e+02	5.076109e+02	5.350319e+02	5.145060e+02	7.263531e+02	4.984211e+02	4.962646e+02	5.067063e+02	5.109507e+02	**4.739111e+02**
f27	**Worst**	5.724600e+02	5.543319e+02	5.994025e+02	6.146886e+02	1.044976e+03	5.899894e+02	5.821073e+02	5.721983e+02	5.891403e+02	**5.332204e+02**
	Mean	5.276361e+02	5.282389e+02	5.642253e+02	5.496064e+02	9.386018e+02	5.296109e+02	5.275970e+02	5.313873e+02	5.470127e+02	**5.038978e+02**
	S.D	1.340101e+01	**1.093669e+01**	1.503594e+01	2.248195e+01	6.709615e+01	1.874197e+01	1.461526e+01	1.635826e+01	1.604371e+01	1.193585e+01
	Sign	+	+	+	+	+	+	+	+	+	
	Best	3.977142e+02	3.149189e+02	5.121806e+02	3.986998e+02	1.799176e+03	3.000002e+02	**3.000000e+02**	4.062057e+02	3.003368e+02	**3.000000e+02**
f28	**Worst**	4.910140e+02	4.995673e+02	6.973400e+02	4.791157e+02	3.446204e+03	4.618452e+02	4.833378e+02	5.343070e+02	4.631358e+02	**4.601094e+02**
	Mean	4.359225e+02	4.205988e+02	5.960447e+02	4.324410e+02	2.550108e+03	3.483333e+02	3.424633e+02	4.631565e+02	4.066490e+02	**3.399951e+02**
	S.D	2.422186e+01	3.324749e+01	3.919602e+01	**2.164031e+01**	3.288267e+02	6.268375e+01	6.264383e+01	2.578495e+01	3.619500e+01	5.728003e+01
	Sign	+	+	+	+	+	=	=	+	+	
	Best	4.415835e+02	4.189983e+02	7.087867e+02	5.364256e+02	2.021730e+03	4.669635e+02	4.724497e+02	4.574674e+02	5.103231e+02	**4.140687e+02**
f29	**Worst**	1.105138e+03	1.143775e+03	1.488220e+03	1.403680e+03	2.779957e+03	1.339511e+03	1.264984e+03	1.097865e+03	1.409881e+03	**8.885631e+02**
	Mean	7.943226e+02	7.742905e+02	1.026206e+03	9.793466e+02	2.365910e+03	9.243629e+02	8.525943e+02	7.747921e+02	9.234220e+02	**5.987537e+02**
	S.D	1.894831e+02	1.842523e+02	1.890201e+02	2.263557e+02	1.543656e+02	2.380941e+02	1.943077e+02	1.738509e+02	2.352952e+02	**1.027363e+02**
	Sign	+	+	+	+	+	+	+	+	+	
	Best	3.174379e+03	3.147548e+03	2.756983e+05	1.137695e+04	1.691767e+07	3.672919e+03	3.557916e+03	3.366411e+03	**3.108583e+03**	4.694056e+03
f 30	**Worst**	1.861610e+04	2.119805e+04	8.346885e+06	9.489619e+05	2.074709e+08	2.234146e+04	2.412678e+04	8.109407e+04	**1.818093e+04**	2.691829e+04
	Mean	1.046595e+04	9.356128e+03	2.300468e+06	8.901509e+04	7.366312e+07	1.264865e+04	1.000818e+04	1.178947e+04	**7.966663e+03**	1.003525e+04
	S.D	4.461697e+03	4.645673e+03	2.070899e+06	1.464950e+05	3.683418e+07	5.056304e+03	4.437855e+03	1.186142e+04	**4.154114e+03**	4.647984e+03
	Sign	=	=	+	+	+	+	=	=	-	

Sign	BWM_HS VS.	HS	IHS	GHS	NGHS	SGHS	IGHS	GDHS	INGHS	LHS
+/=/-		24/6/0	23/6/1	30/0/0	30/0/0	30/0/0	19/10/1	18/9/3	27/3/0	23/5/2

Table 6.
Results obtained for 51 independent runs of BWM_HS and eight recent variants of HS on 50D benchmarks of CEC2017.

No.	Algorithms	HS	IHS	GHS	NGHS	SGHS	IGHS	GDHS	INGHS	LHS	BWM_HS
	Unimodal										
	Best	7.008629e+06	7.941216e+01	8.077923e+09	1.894807e+10	7.260629e+10	9.412807e-01	**5.088245e-01**	7.405311e+07	2.935512e+01	7.909614e+00
f1	**Worst**	3.548872e+07	4.013174e+05	1.894807e+10	1.102489e+06	1.212448e+11	3.157455e+04	**3.092259e+04**	2.647743e+09	3.179056e+04	3.151857e+04
	Mean	1.740643e+07	1.893881e+04	1.402425e+10	3.949224e+05	1.006051e+11	7.783403e+03	7.074646e+03	3.692506e+08	6.412109e+03	**4.822875e+03**
	S.D	6.507575e+06	5.659768e+04	2.279610e+09	2.022056e+05	1.057667e+10	7.931783e+03	8.404485e+03	4.311917e+08	8.264014e+03	**6.979403e+03**
	Sign	+	+	+	+	+	+	=	+	=	
	Best	4.848892e+24	7.831974e+20	6.029363e+53	1.531069e+11	1.562409e+67	9.984774e-06	6.905668e-06	7.291970e+29	9.471569e+01	**5.781283e-06**
f 2	**Worst**	4.113112e+36	2.297355e+37	9.384740e+61	3.785050e+36	5.210400e+73	7.480888e-05	8.279871e-05	2.434865e+42	1.271884e+30	**6.299082e-05**
	Mean	1.059350e+35	4.508879e+35	4.206742e+60	7.629655e+34	3.409459e+72	4.289620e-05	2.712888e-05	5.989614e+40	2.494062e+28	**2.655128e-05**
	S.D	5.928717e+35	3.216880e+36	1.517361e+61	5.298971e+35	9.846401e+72	1.500406e-05	1.440685e-05	3.455874e+41	1.780991e+29	**1.345615e-05**
	Sign	+	+	+	+	+	+	=	+	+	
	Best	1.324269e+04	1.310361e+04	9.382198e+04	1.070415e+05	1.614250e+05	1.872589e-06	1.449868e-05	1.221134e+04	1.387317e+04	**8.859291e-08**
	Worst	3.320857e+04	3.283148e+04	1.746166e+05	3.462597e+05	2.459090e+05	3.580127e-06	4.725582e-05	3.889528e+04	6.017382e+04	**3.896289e-07**
f3	**Mean**	2.198102e+04	2.236392e+04	1.200990e+05	1.977834e+05	2.122238e+05	2.632798e-06	2.836164e-05	2.210669e+04	3.316668e+04	**2.300498e-07**
	S.D	4.430381e+03	4.067189e+03	1.680446e+04	4.628436e+04	2.062704e+04	3.550205e-07	7.318913e-06	5.516200e+03	1.100105e+04	**6.173167e-08**
	Sign	+	+	+	+	+	+	+	+	+	
	Multimodal										
	Best	8.678231e+01	8.710618e+01	1.208954e+03	2.960795e+01	1.273333e+04	1.949813e+01	**1.317310e-02**	1.128467e+02	1.067192e+01	1.715797e+01
f4	**Worst**	3.330217e+02	3.474533e+02	3.140751e+03	3.176110e+02	2.998552e+04	1.848913e+02	**1.560741e+02**	4.048035e+02	2.215952e+02	1.860652e+02
	Mean	2.159781e+02	2.106780e+02	1.960072e+03	1.604664e+02	2.270968e+04	8.806598e+01	**6.799456e+01**	2.400759e+02	1.289377e+02	7.671882e+01
	S.D	5.364006e+01	6.131371e+01	4.015262e+02	7.244375e+01	3.679831e+03	**5.307734e+01**	5.366952e+01	6.604307e+01	5.630201e+01	5.628236e+01
	Sign	+	+	+	+	+	=	=	+	+	
	Best	7.730708e+01	8.357712e+01	3.901672e+02	1.266002e+02	6.531908e+02	1.611827e+02	6.964705e+01	1.722194e+02	1.233747e+02	**5.969748e+01**
f 5	**Worst**	1.996280e+02	1.874001e+02	5.094112e+02	2.997278e+02	8.048703e+02	3.422637e+02	1.651627e+02	4.671658e+02	3.114196e+02	**1.471399e+02**
	Mean	1.386481e+02	1.359798e+02	4.475694e+02	2.012539e+02	7.446472e+02	2.354300e+02	1.183087e+02	2.979971e+02	2.026664e+02	**1.024784e+02**
	S.D	2.611056e+01	2.687644e+01	2.920225e+01	3.909905e+01	3.168616e+01	4.191250e+01	2.290910e+01	8.176318e+01	4.392988e+01	**1.808918e+01**
	Sign	+	+	+	+	+	+	+	+	+	
	Best	5.904867e-01	5.094122e-01	1.730051e+01	5.798784e-02	7.232980e+01	6.013892e-02	2.547388e-04	1.506301e+01	3.254727e-04	**6.075276e-07**
f6	**Worst**	1.665880e+00	1.951694e+00	3.077542e+01	2.283445e-01	1.029260e+02	3.546947e-01	4.833544e-03	5.418918e+01	1.200918e-02	**9.191797e-04**
	Mean	1.067208e+00	1.032543e+00	2.514118e+01	1.437792e-01	9.126437e+01	1.715180e-01	1.019743e-03	3.502527e+01	3.606161e-03	**7.301065e-05**
	S.D	2.258404e-01	3.186833e-01	3.272292e+00	3.625694e-02	5.200000e+00	6.780048e-02	1.055972e-03	8.281911e+00	2.775102e-03	**1.819539e-04**
	Sign	+	+	+	+	+	+	+	+	+	
	Best	1.847962e+02	1.827265e+02	5.933380e+02	1.780462e+02	1.429247e+03	1.996667e+02	1.397538e+02	3.393923e+02	1.853793e+02	**1.104661e+02**
f7	**Worst**	3.304526e+02	3.096218e+02	7.979829e+02	3.362563e+02	2.747478e+03	4.368479e+02	2.753065e+02	6.971972e+02	3.641493e+02	**1.860314e+02**

	Mean	2.547674e+02	2.373302e+02	6.944302e+02	2.590825e+02	2.094122e+03	2.709794e+02	1.825189e+02	5.601441e+02	2.611366e+02	**1.420127e+02**
	S.D	3.610098e+01	3.017674e+01	4.788891e+01	3.746826e+01	3.517821e+02	4.024547e+01	2.704378e+01	8.393039e+01	4.380263e+01	**1.787046e+01**
	Sign	+	+	+	+	+	+	+	+	+	
	Best	1.004151e+02	8.906387e+01	3.444302e+02	1.424078e+02	6.098655e+02	1.765491e+02	**6.467224e+01**	1.787462e+02	1.373038e+02	8.258147e+01
f8	**Worst**	1.917422e+02	2.069494e+02	5.249117e+02	3.752367e+02	7.899776e+02	3.651468e+02	1.751122e+02	5.477918e+02	3.124199e+02	**1.683602e+02**
	Mean	1.438844e+02	1.320197e+02	4.458717e+02	2.239237e+02	7.395794e+02	2.433918e+02	1.244423e+02	3.438150e+02	2.034367e+02	**1.152306e+02**
	S.D	2.397477e+01	2.787746e+01	3.482481e+01	5.715163e+01	3.729953e+01	4.289407e+01	2.337009e+01	9.408423e+01	3.635656e+01	**2.007441e+01**
	Sign	+	+	+	+	+	+	+	+	+	
	Best	2.336572e+02	3.120209e+02	5.174010e+03	5.207220e+02	3.144287e+04	2.678561e+03	1.342187e+01	1.038940e+04	3.745103e+02	**2.142997e-10**
f9	**Worst**	3.345475e+03	**1.832849e+03**	1.275164e+04	1.384857e+04	4.698141e+04	1.580903e+04	2.325023e+03	3.154616e+04	9.855942e+03	4.127673e+03
	Mean	1.315445e+03	8.776051e+02	8.815951e+03	4.452806e+03	3.895161e+04	8.073416e+03	4.441769e+02	2.011268e+04	3.297636e+03	**2.635973e+02**
	S.D	6.905177e+02	**3.804495e+02**	1.687653e+03	2.769780e+03	3.672784e+03	3.774858e+03	4.115273e+02	5.951120e+03	2.207431e+03	7.706110e+02
	Sign	+	+	+	+	+	+	+	+	+	
	Best	2.912553e+03	2.717763e+03	1.029980e+04	3.537289e+03	1.257634e+04	3.456027e+03	2.631603e+03	4.250715e+03	3.193009e+03	**2.325546e+03**
f10	**Worst**	6.125331e+03	5.482285e+03	1.373141e+04	6.015561e+03	1.399068e+04	6.338975e+03	**5.045947e+03**	1.286391e+04	6.219623e+03	5.245775e+03
	Mean	4.307696e+03	4.119057e+03	1.245370e+04	4.801354e+03	1.346774e+04	4.669245e+03	3.832392e+03	8.960299e+03	4.523957e+03	**3.697111e+03**
	S.D	7.396287e+02	6.605546e+02	6.192701e+02	6.385476e+02	**3.224950e+02**	7.000895e+02	5.849327e+02	2.469903e+03	6.745187e+02	6.761201e+02
	Sign	+	+	+	+	+	+	=	+	+	
	Hybrid										
	Best	1.151876e+02	6.382414e+01	8.509519e+02	2.592730e+02	8.724284e+03	1.257042e+02	**3.793777e+01**	1.448877e+02	9.391451e+01	1.037741e+02
f11	**Worst**	2.161760e+03	7.998893e+02	5.027041e+03	3.345652e+04	2.158252e+04	4.580468e+02	**1.623768e+02**	2.866082e+03	2.144955e+03	2.919676e+02
	Mean	5.558298e+02	2.635293e+02	2.182703e+03	8.898325e+03	1.508319e+04	2.276111e+02	**9.879876e+01**	6.171765e+02	5.421515e+02	1.790622e+02
	S.D	3.812584e+02	1.889629e+02	9.518510e+02	8.596780e+03	2.733634e+03	6.205992e+01	**3.104765e+01**	7.409494e+02	5.095575e+02	5.023918e+01
	Sign	+	=	+	+	+	+	-	+	+	
	Best	1.152095e+06	1.594220e+06	3.820915e+08	2.310711e+06	1.491941e+10	2.236224e+05	2.534516e+05	4.335872e+06	4.061462e+05	**1.718606e+05**
f12	**Worst**	3.248714e+07	2.265535e+07	2.710158e+09	3.625029e+07	3.817920e+10	6.241683e+06	6.151903e+06	1.224283e+08	9.701914e+06	6.444498e+06
	Mean	1.120048e+07	1.018132e+07	1.448702e+09	1.302817e+07	2.760056e+10	2.472226e+06	2.124365e+06	1.957163e+07	3.173653e+06	**1.989677e+06**
	S.D	5.688205e+06	4.231623e+06	4.790339e+08	6.968623e+06	5.584733e+09	1.318768e+06	1.416708e+06	2.133545e+07	1.748987e+06	**1.303682e+06**
	Sign	+	+	+	+	+	=	=	+	+	
	Best	5.166199e+02	2.590908e+02	1.550559e+06	3.293049e+04	4.323961e+09	3.617142e+02	3.477417e+02	1.634145e+03	**1.188685e+02**	2.483246e+02
f13	**Worst**	3.814859e+04	3.781932e+04	1.858598e+07	5.639320e+05	1.205099e+10	3.778613e+04	3.699188e+04	5.911016e+05	**2.146167e+04**	3.773787e+04
	Mean	8.285528e+03	8.055195e+03	7.812768e+06	1.883195e+05	7.518996e+09	1.098198e+04	8.771156e+03	2.751574e+04	**3.741400e+03**	8.329137e+03
	S.D	9.535225e+03	1.000404e+04	3.788975e+06	1.119848e+05	1.949510e+09	1.057295e+04	1.035788e+04	8.812670e+04	**4.510306e+03**	1.049949e+04
	Sign	=	=	+	+	+	=	=	+	-	
	Best	2.953401e+04	5.283291e+03	1.159319e+05	1.203969e+06	8.670363e+05	3.631581e+02	**1.684150e+03**	1.339536e+04	1.478301e+05	2.647205e+03
f14	**Worst**	2.012463e+06	1.754633e+05	8.674284e+06	2.308657e+07	1.105900e+07	4.823978e+04	4.883181e+04	7.125372e+05	2.571932e+06	1.077786e+05
	Mean	4.414018e+05	6.863287e+04	2.928191e+06	6.533564e+06	5.351678e+06	**1.435471e+04**	1.845692e+04	1.693231e+05	8.517712e+05	2.699636e+04
	S.D	4.184548e+05	4.106273e+04	1.978042e+06	4.493210e+06	2.139002e+06	1.272860e+04	**1.181119e+04**	1.642516e+05	5.804747e+05	2.200514e+04
	Sign	+	+	+	+	+	-	=	+	+	
	Best	1.896436e+02	**7.852281e+01**	6.861899e+04	4.874923e+03	3.303880e+08	1.854433e+02	2.764439e+02	4.466713e+02	9.245484e+01	1.636613e+02
f15	**Worst**	3.073889e+04	2.915850e+04	1.272276e+06	2.549424e+05	3.754400e+09	3.067786e+04	3.153763e+04	1.849864e+04	**1.815025e+04**	2.145306e+04
	Mean	9.687987e+03	7.811491e+03	4.053904e+05	9.998907e+04	1.076832e+09	8.624438e+03	8.940652e+03	1.041834e+04	8.036433e+03	**7.269766e+03**
	S.D	7.012601e+03	7.340254e+03	2.108919e+05	5.794566e+04	6.006339e+08	6.637384e+03	8.066370e+03	6.611965e+03	**6.024012e+03**	6.679883e+03
	Sign	=	=	+	+	+	=	=	+	=	
	Best	5.333479e+02	7.051804e+02	2.422672e+03	1.118812e+03	4.497346e+03	5.218559e+02	7.686074e+02	4.596850e+02	1.087819e+03	**3.785607e+02**
f16	**Worst**	2.554934e+03	2.798922e+03	4.147524e+03	3.047514e+03	5.532090e+03	2.840388e+03	2.399371e+03	2.432229e+03	3.137530e+03	**1.982347e+03**
	Mean	1.716341e+03	1.709591e+03	3.513139e+03	2.158850e+03	5.008190e+03	1.689330e+03	1.625911e+03	1.388205e+03	2.168454e+03	**1.008178e+03**
	S.D	4.505903e+02	4.391297e+02	3.509643e+02	4.614926e+02	2.474824e+02	5.053660e+02	4.165062e+02	4.379579e+02	4.953326e+02	**3.311631e+02**
	Sign	+	+	+	+	+	+	+	+	+	
	Best	4.862258e+02	6.008166e+02	1.038018e+03	6.937013e+02	2.856293e+03	3.421994e+02	2.518886e+02	6.032440e+02	6.491108e+02	**2.407448e+02**
f17	**Worst**	1.977242e+03	1.831048e+03	2.448504e+03	2.152452e+03	4.774323e+03	1.997515e+03	1.828233e+03	1.866452e+03	2.144992e+03	**1.447678e+03**
	Mean	1.254918e+03	1.149130e+03	1.772196e+03	1.529458e+03	3.659674e+03	1.231499e+03	1.088634e+03	1.142732e+03	1.377109e+03	**7.522037e+02**
	S.D	3.072762e+02	**2.638950e+02**	3.108343e+02	3.736374e+02	3.849098e+02	3.587862e+02	3.224804e+02	3.005825e+02	3.695310e+02	2.682327e+02
	Sign	+	+	+	+	+	+	+	+	+	
	Best	3.129408e+05	3.689506e+04	1.082260e+06	9.449997e+05	1.568404e+07	**3.106714e+04**	3.470069e+04	2.922410e+05	3.015150e+05	5.269191e+04
f18	**Worst**	9.324134e+06	3.132986e+06	3.102907e+07	4.079554e+07	6.329859e+07	**2.400258e+05**	2.663439e+05	5.142228e+06	7.006765e+06	3.671394e+05
	Mean	3.379459e+06	8.831437e+05	1.091285e+07	1.258659e+07	3.627596e+07	**1.093060e+05**	1.184535e+05	1.452661e+06	1.899737e+06	1.301734e+05
	S.D	2.498744e+06	8.393180e+05	7.180972e+06	8.849593e+06	1.183622e+07	**4.496889e+04**	5.403441e+04	8.714185e+05	1.378975e+06	6.965701e+04
	Sign	+	+	+	+	+	=	=	+	+	
	Best	7.154047e+01	**6.948234e+01**	7.143801e+04	1.023431e+04	1.132755e+08	2.603211e+02	8.637600e+01	1.955396e+02	1.618713e+02	1.385497e+02
f19	**Worst**	**4.127884e+04**	4.252617e+04	5.869416e+05	1.033226e+05	1.362698e+09	4.227728e+04	4.247701e+04	4.265653e+04	4.186403e+04	4.245404e+04
	Mean	1.614784e+04	1.762600e+04	2.376610e+05	3.925340e+04	5.108462e+08	2.211992e+04	1.694282e+04	2.380401e+04	1.529596e+04	**1.359815e+04**
	S.D	1.345323e+04	1.564807e+04	1.207321e+05	2.009514e+04	2.437123e+08	1.324111e+04	1.503278e+04	1.548429e+04	**1.051531e+04**	1.343354e+04
	Sign	=	=	+	+	+	+	=	+	=	
	Best	3.444710e+02	2.555547e+02	6.541464e+02	5.825927e+02	1.564511e+03	3.864433e+02	3.249270e+02	1.970959e+02	4.818954e+02	**1.571507e+02**
f 20	**Worst**	1.584909e+03	1.575388e+03	1.816359e+03	1.962665e+03	2.196288e+03	1.827802e+03	1.587721e+03	1.995473e+03	1.838014e+03	**1.076577e+03**
	Mean	9.977685e+02	9.344746e+02	1.229292e+03	1.312281e+03	1.963548e+03	1.042468e+03	9.465916e+02	7.901374e+02	1.156920e+03	**5.449835e+02**
	S.D	3.080033e+02	3.195002e+02	2.661891e+02	3.139180e+02	**1.397764e+02**	3.103023e+02	2.788830e+02	3.473337e+02	2.813182e+02	2.094613e+02
	Sign	+	+	+	+	+	+	+	+	+	
	Composition										
	Best	2.862204e+02	2.808191e+02	5.627442e+02	3.190110e+02	8.586186e+02	3.788757e+02	2.893953e+02	3.693141e+02	3.224291e+02	**2.611300e+02**
f21	**Worst**	4.300325e+02	3.952399e+02	7.175913e+02	5.637009e+02	1.015386e+03	5.803348e+02	**3.708170e+02**	7.041102e+02	5.099072e+02	3.796513e+02
	Mean	3.463760e+02	3.376087e+02	6.538434e+02	4.186465e+02	9.402263e+02	4.534870e+02	3.274298e+02	5.138540e+02	4.203264e+02	**3.071108e+02**
	S.D	2.745962e+01	2.521537e+01	3.474544e+01	5.156587e+01	3.394726e+01	4.677549e+01	**1.906048e+01**	7.742136e+01	4.202555e+01	2.327844e+01
	Sign	+	+	+	+	+	+	+	+	+	
	Best	3.260900e+03	3.409200e+03	1.122626e+04	4.178887e+03	1.255573e+04	3.857315e+03	3.424716e+03	4.342538e+03	1.000123e+02	**1.000001e+02**

f22	**Worst**	6.541800e+03	5.771700e+03	1.408939e+04	7.314757e+03	1.443697e+04	6.967033e+03	**5.765812e+03**	1.287804e+04	7.272517e+03	6.285794e+03
	Mean	4.903524e+03	4.723394e+03	1.303503e+04	5.913439e+03	1.387790e+04	5.551882e+03	4.530975e+03	8.463638e+03	5.302154e+03	**4.485235e+03**
	S.D	7.132740e+02	5.785733e+02	5.726724e+02	7.766283e+02	**3.636594e+02**	7.163132e+02	5.522041e+02	2.546827e+03	1.257763e+03	9.706367e+02
	Sign	+	=	+	+	+	+	=	+	+	
f23	**Best**	5.477300e+02	5.243600e+02	8.471717e+02	5.684729e+02	1.321894e+03	6.148334e+02	4.961987e+02	6.239651e+02	5.841357e+02	**4.635544e+02**
	Worst	7.013200e+02	6.748900e+02	1.031663e+03	8.377716e+02	1.615513e+03	8.025477e+02	6.309152e+02	1.207586e+03	8.187358e+02	**6.114797e+02**
	Mean	5.952324e+02	5.940296e+02	9.423554e+02	6.847272e+02	1.473725e+03	7.060660e+02	5.772673e+02	8.257060e+02	6.756387e+02	**5.400932e+02**
	S.D	2.937571e+01	3.260657e+01	3.494517e+01	6.948984e+01	6.184436e+01	5.325679e+01	3.234864e+01	1.292198e+02	5.282117e+01	**2.749018e+01**
	Sign	+	+	+	+	+	+	+	+	+	
f24	**Best**	5.823200e+02	5.688300e+02	9.439732e+02	8.927205e+02	1.403232e+03	7.086697e+02	5.743593e+02	7.047928e+02	8.119691e+02	**5.621014e+02**
	Worst	9.068100e+02	7.154700e+02	1.063820e+03	1.475524e+03	1.735382e+03	1.203956e+03	**6.892742e+02**	1.173555e+03	1.381630e+03	6.924050e+02
	Mean	6.381275e+02	6.332918e+02	1.014193e+03	1.125155e+03	1.577355e+03	9.695000e+02	6.312178e+02	9.092592e+02	1.093274e+03	**6.175078e+02**
	S.D	6.071173e+01	3.321711e+01	**2.546359e+01**	1.312149e+02	7.523610e+01	1.145065e+02	2.587455e+01	1.372633e+02	1.437862e+02	2.854694e+01
	Sign	=	+	+	+	+	+	+	+	+	
f25	**Best**	5.371700e+02	5.007000e+02	1.349508e+03	5.241232e+02	8.626628e+03	4.593173e+02	**4.284845e+02**	5.636701e+02	4.730604e+02	4.602784e+02
	Worst	6.620700e+02	6.287900e+02	2.326222e+03	6.545589e+02	1.896132e+04	5.960935e+02	6.027544e+02	8.708655e+02	6.096151e+02	**5.688700e+02**
	Mean	5.930867e+02	5.763627e+02	1.763030e+03	5.794503e+02	1.319905e+04	5.180139e+02	5.151524e+02	6.977220e+02	5.565691e+02	**5.002684e+02**
	S.D	3.200490e+01	2.901848e+01	2.419963e+02	**2.747785e+01**	2.270039e+03	4.405639e+01	4.260156e+01	6.335536e+01	3.218948e+01	3.123827e+01
	Sign	+	+	+	+	+	=	=	+	+	
f26	**Best**	4.874400e+02	2.011300e+03	5.818627e+03	3.023061e+02	1.072627e+04	3.000038e+02	2.167051e+03	6.994314e+02	3.000027e+02	**3.000000e+02**
	Worst	4.105600e+03	4.047700e+03	7.292713e+03	5.177676e+03	1.429062e+04	7.691320e+03	3.473743e+03	7.891534e+03	6.370023e+03	**2.904223e+03**
	Mean	2.905724e+03	2.867320e+03	6.423032e+03	3.574115e+03	1.228806e+04	4.084386e+03	2.756338e+03	4.344211e+03	3.806454e+03	**2.149907e+03**
	S.D	5.169083e+02	3.656816e+02	3.526148e+02	1.050630e+03	8.073728e+02	1.100645e+03	**2.993978e+02**	1.746420e+03	1.135092e+03	5.976180e+02
	Sign	+	+	+	+	+	+	+	+	+	
f27	**Best**	5.816485e+02	5.738726e+02	8.715911e+02	6.624295e+02	1.968758e+03	5.372660e+02	5.622132e+02	6.176011e+02	6.553086e+02	**5.066696e+02**
	Worst	8.207768e+02	9.188115e+02	1.263299e+03	1.116668e+03	2.818523e+03	1.011658e+03	8.634563e+02	8.637480e+02	1.077213e+03	**6.712272e+02**
	Mean	6.764429e+02	6.959941e+02	1.070763e+03	8.552416e+02	2.354943e+03	6.885304e+02	6.418420e+02	7.513259e+02	8.233999e+02	**5.702168e+02**
	S.D	6.549233e+01	7.191428e+01	8.570209e+01	1.198805e+02	1.809005e+02	1.023113e+02	5.194843e+01	6.170624e+01	9.819377e+01	**3.205468e+01**
	Sign	+	+	+	+	+	+	+	+	+	
f28	**Best**	4.923430e+02	4.743646e+02	1.299928e+03	4.842341e+02	7.092709e+03	4.588490e+02	4.588498e+02	5.156117e+02	4.588958e+02	**4.588488e+02**
	Worst	6.878640e+02	6.172493e+02	2.622113e+03	6.494183e+02	9.790296e+03	5.195427e+02	6.021574e+02	7.636451e+02	6.029424e+02	**5.145091e+02**
	Mean	5.655934e+02	5.581018e+02	1.766457e+03	5.362084e+02	8.480408e+03	4.945272e+02	4.938980e+02	5.959912e+02	5.101103e+02	**4.846882e+02**
	S.D	3.989019e+01	2.989266e+01	2.571615e+02	3.278119e+01	6.049146e+02	1.949395e+01	2.894505e+01	5.499053e+01	2.523858e+01	2.338845e+01
	Sign	+	+	+	+	+	+	=	+	+	
f29	**Best**	4.654990e+02	4.785390e+02	1.550990e+03	6.819684e+02	4.804336e+03	6.089301e+02	**4.499590e+02**	6.765456e+02	6.627739e+02	4.535416e+02
	Worst	1.483072e+03	1.794724e+03	3.552823e+03	2.375089e+03	9.341729e+03	2.059885e+03	1.756820e+03	2.177538e+03	2.499523e+03	**1.406437e+03**
	Mean	1.014282e+03	1.016306e+03	2.481143e+03	1.443662e+03	6.544050e+03	1.218076e+03	1.047612e+03	1.191573e+03	1.340636e+03	**8.008333e+02**
	S.D	2.116198e+02	2.919426e+02	3.966966e+02	3.668357e+02	8.205849e+02	3.250391e+02	2.972326e+02	2.770662e+02	3.435701e+02	**2.014597e+02**
	Sign	+	+	+	+	+	+	+	+	+	
f30	**Best**	7.091059e+05	6.827765e+05	8.116459e+06	6.585548e+05	7.402209e+08	5.996783e+05	6.703273e+05	6.573740e+05	6.638180e+05	**5.879322e+05**
	Worst	2.225263e+06	2.364901e+06	3.597213e+07	2.258927e+06	2.298379e+09	1.619969e+06	1.631157e+06	2.110431e+06	1.622992e+06	**1.046200e+06**
	Mean	1.187887e+06	1.285498e+06	2.141256e+07	1.075911e+06	1.447117e+09	8.943219e+05	9.335388e+05	1.063003e+06	8.979771e+05	**7.735523e+05**
	S.D	3.640728e+05	4.665598e+05	7.076530e+06	3.182663e+05	3.864873e+08	2.190059e+05	2.492993e+05	2.514587e+05	2.058842e+05	**1.229850e+05**
	Sign	+	+	+	+	+	+	+	+	+	

Sign +/=/-	BWM_HS VS.	HS	IHS	GHS	NGHS	SGHS	IGHS	GDHS	INGHS	LHS
		26/4/0	**25/5/0**	30/0/0	**30/0/0**	**30/0/0**	**23/6/1**	**16/13/1**	**30/0/0**	**26/3/1**

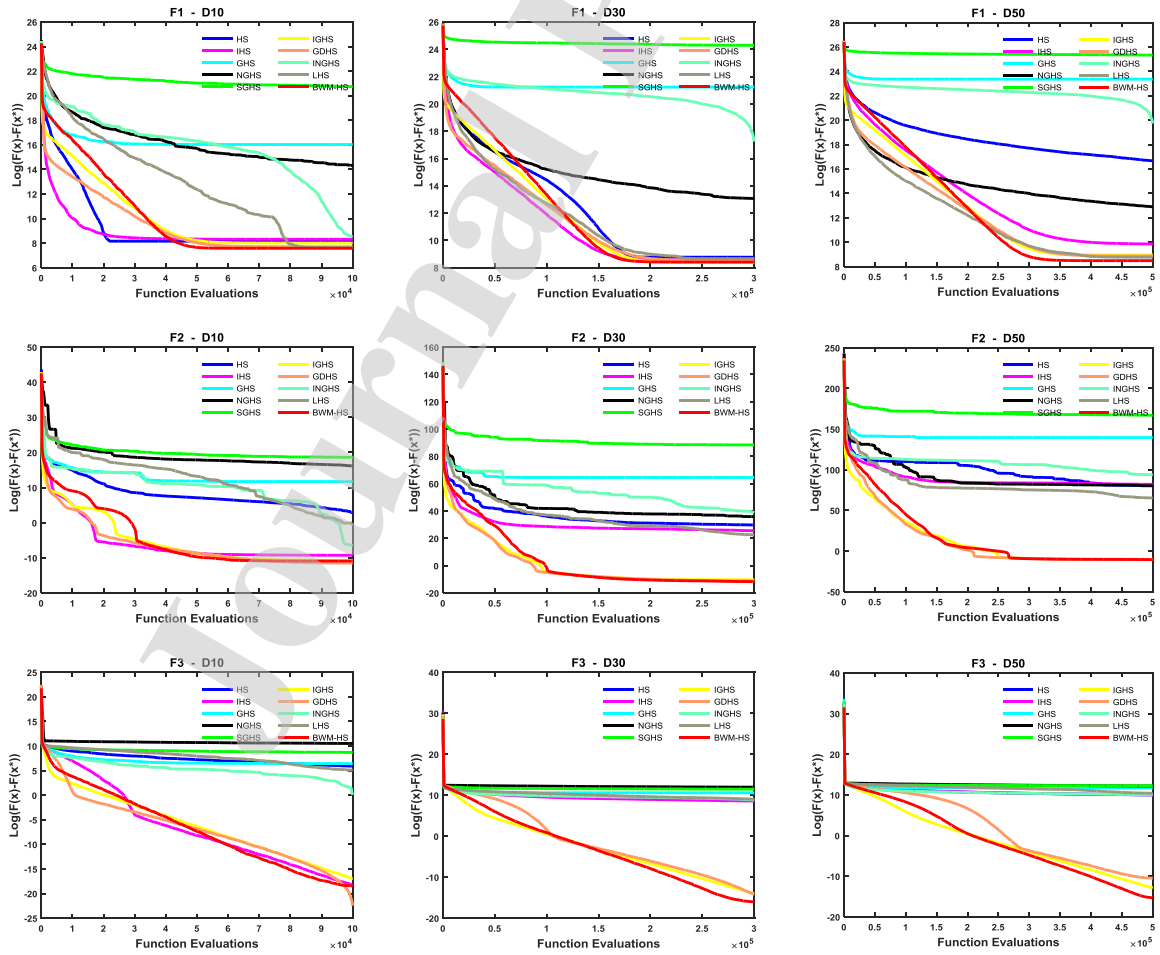
The unimodal functions have no local optima, and only present one global optimum. Indeed, the main purpose of investigating unimodal functions is to examine the exploitation capability of the algorithms. In this case, BWM_HS benefits from four key features. First, it utilizes the valuable information of the best harmony via the *best* rule to guide the search direction toward the promising regions. Second, the mean rule (Eq. (4)) helps the algorithm to exploit the information of whole HM in generation of new harmonies. Third, the large value of PAR with small value of BW at the final stages of search, enhances the local exploitation ability of the algorithm. Fourth, generation of two new harmonies instead of one, enables the algorithm to utilize the information of HM more exhaustively. In regards to above mentioned properties, BWM_HS works well on the exploitation phase and achieves better mean values than other algorithms for *f1* and *f3* on 10D; *f2* and *f3* on 30D; and *f1*, *f2*, and *f3* on 50D.

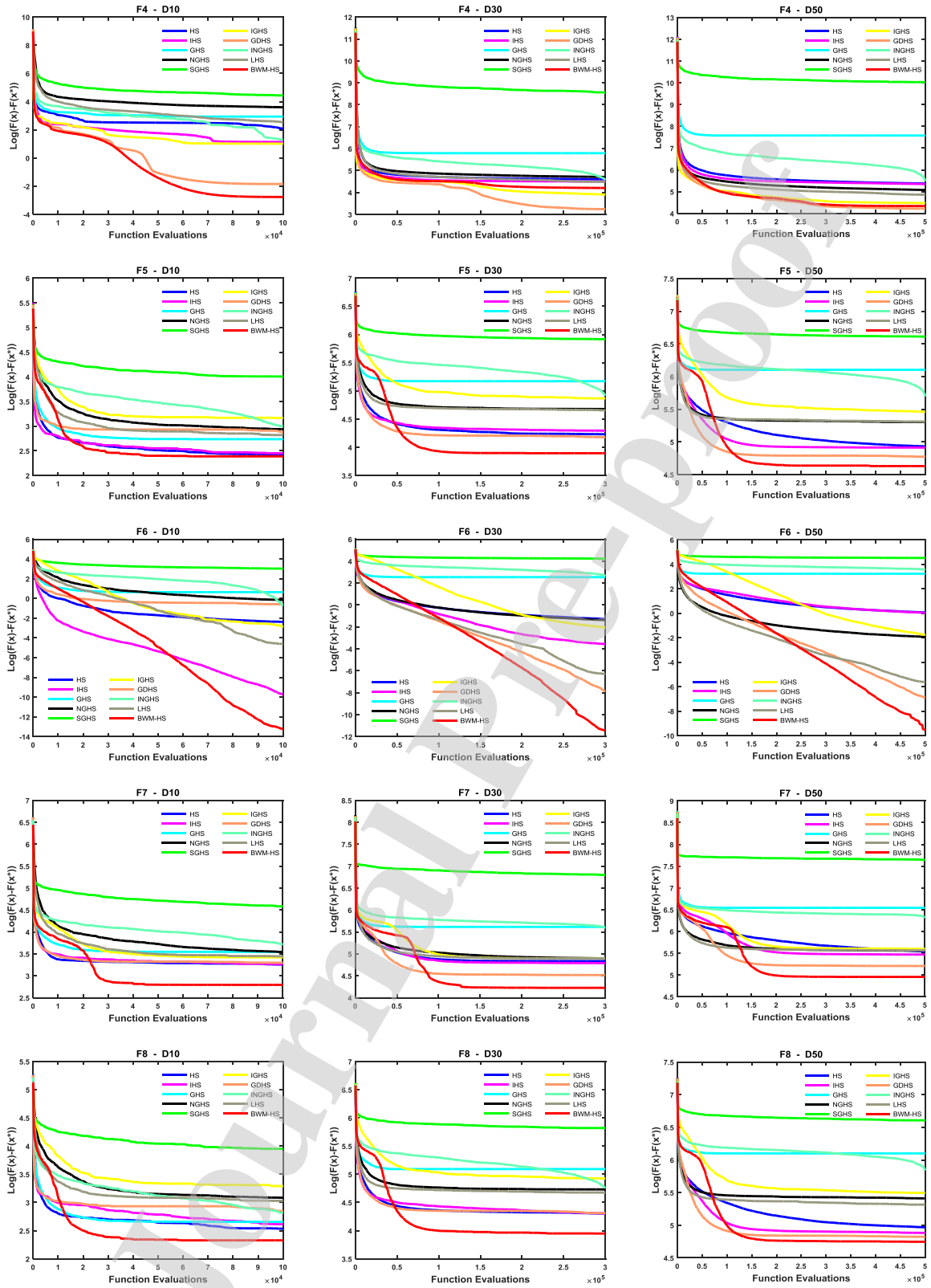
Multimodal functions have a large number of local optima and generally are used to evaluate the exploration ability of search algorithms. According to the mean values reported in these Tables 4-6, the BWM_HS has obtained better results for *f4* to *f9* on 10D and 50D and for *f5* to *f10* on 30D. These excellent results demonstrate the strong exploration ability of BWM_HS algorithm especially at the early stages of the search course. To explain more, at these stages, the *best* rule (Eq. (2)) and the *mean*

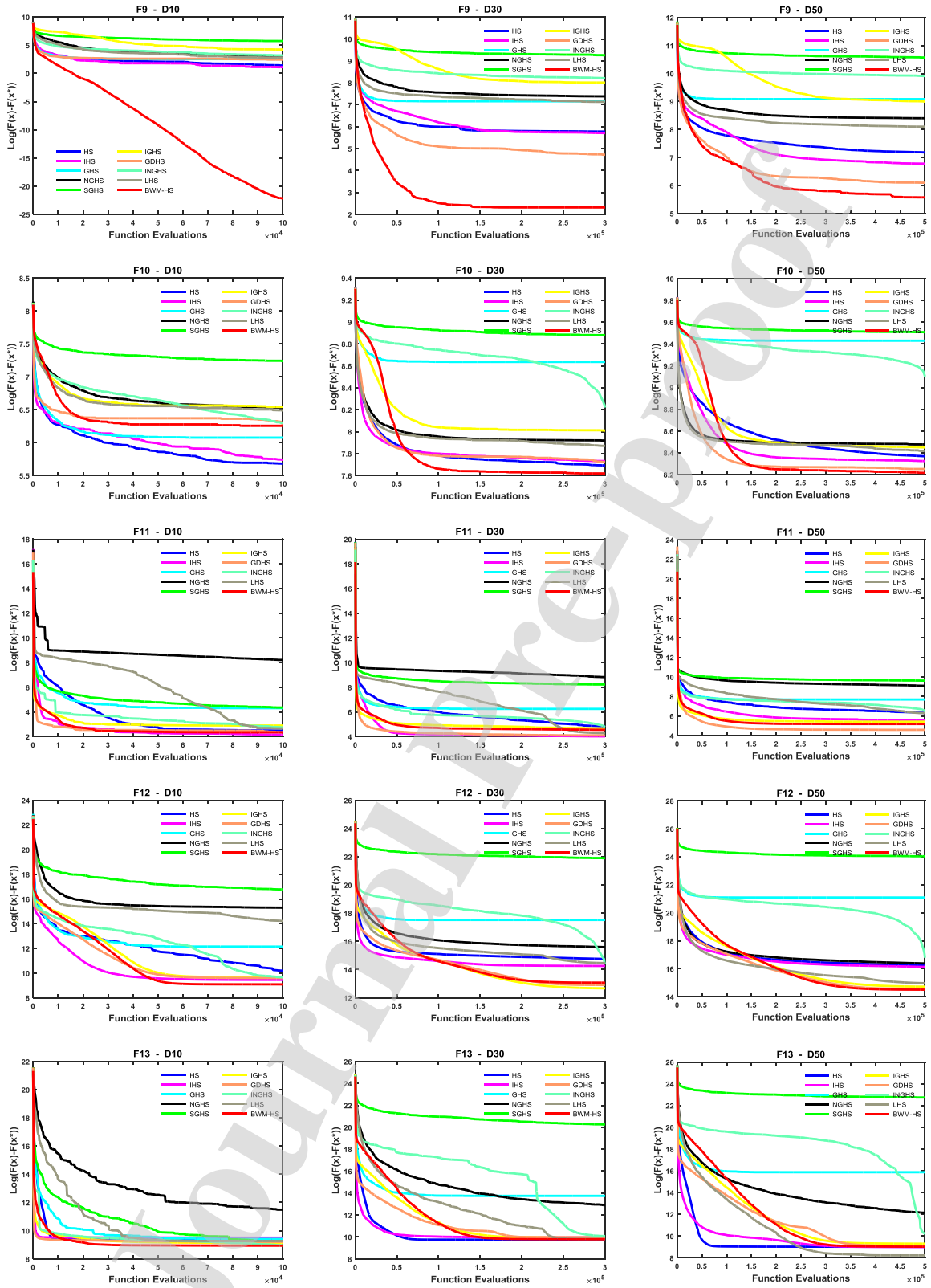
rule (Eq. (4)) focus on exploration around the current best harmony and other randomly selected harmonies in HM, respectively. Second, large value of BW at the early iterations of search process increases the capability of detecting promising regions in the search space.

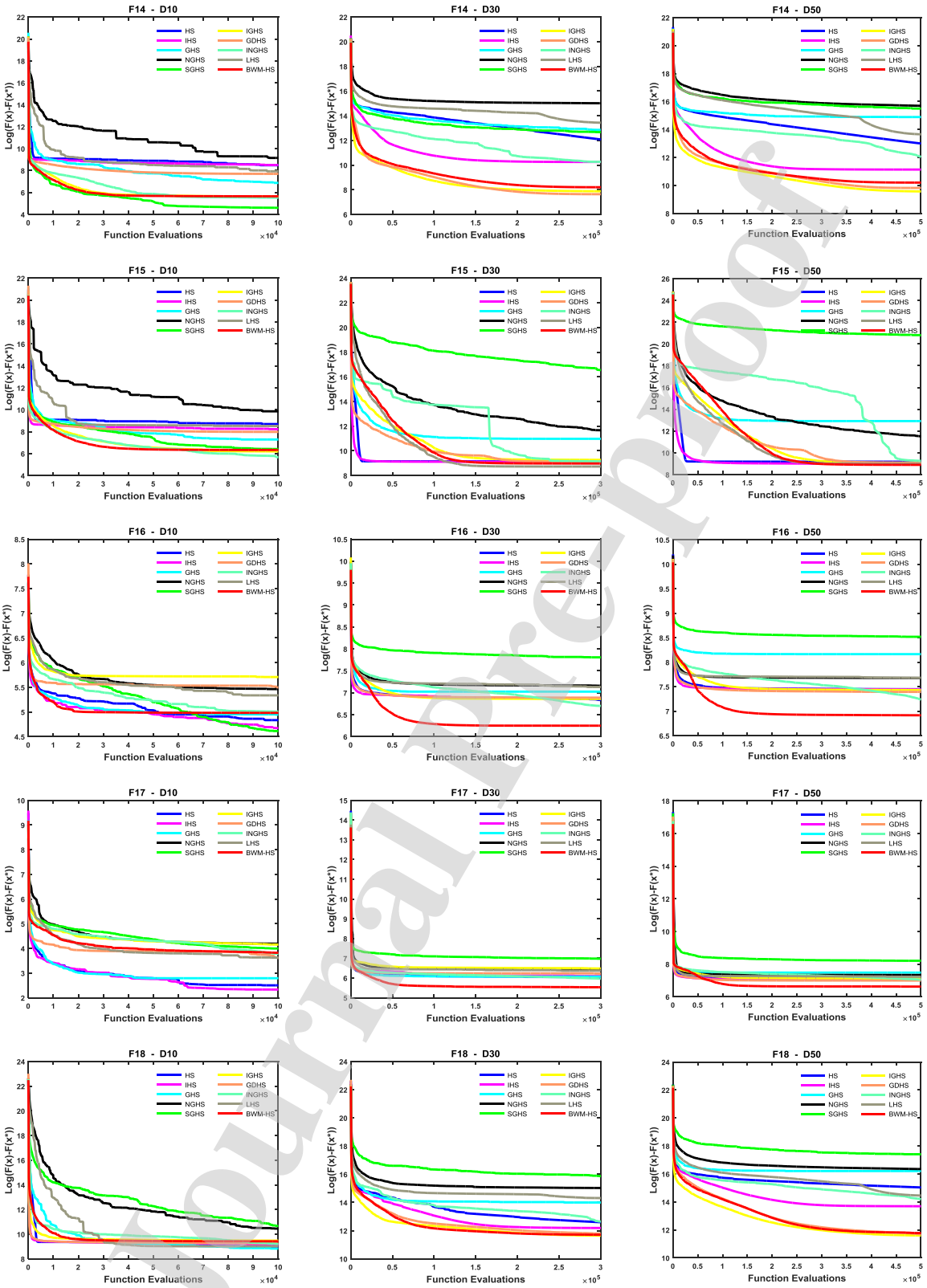
Hybrid functions constitute of enormous numbers of local optimal that are scattered in the search space. Hence, these functions are used to show the ability of algorithms to escape local optimal. The BWM_HS employs the self-adaptive selection parameter that gives more chance to the *worst* rule (Eq. (3)) in generation of new harmonies at the final stages of the search process. In addition, as mentioned before, the value of PAR should be large at these stages, so the pitch adjustment process performs continuous perturbation around the selected pitches with Gaussian distribution. These key features have prevented the BWM_HS of being trapped in to local optimal of these functions and so it has gained better mean values than other algorithms for *f12* and *f13* on 10D; *f16* and *f17* on 30D; and *f12*, *f15*, *f16*, *f17*, *f19*, and *f20* on 50D.

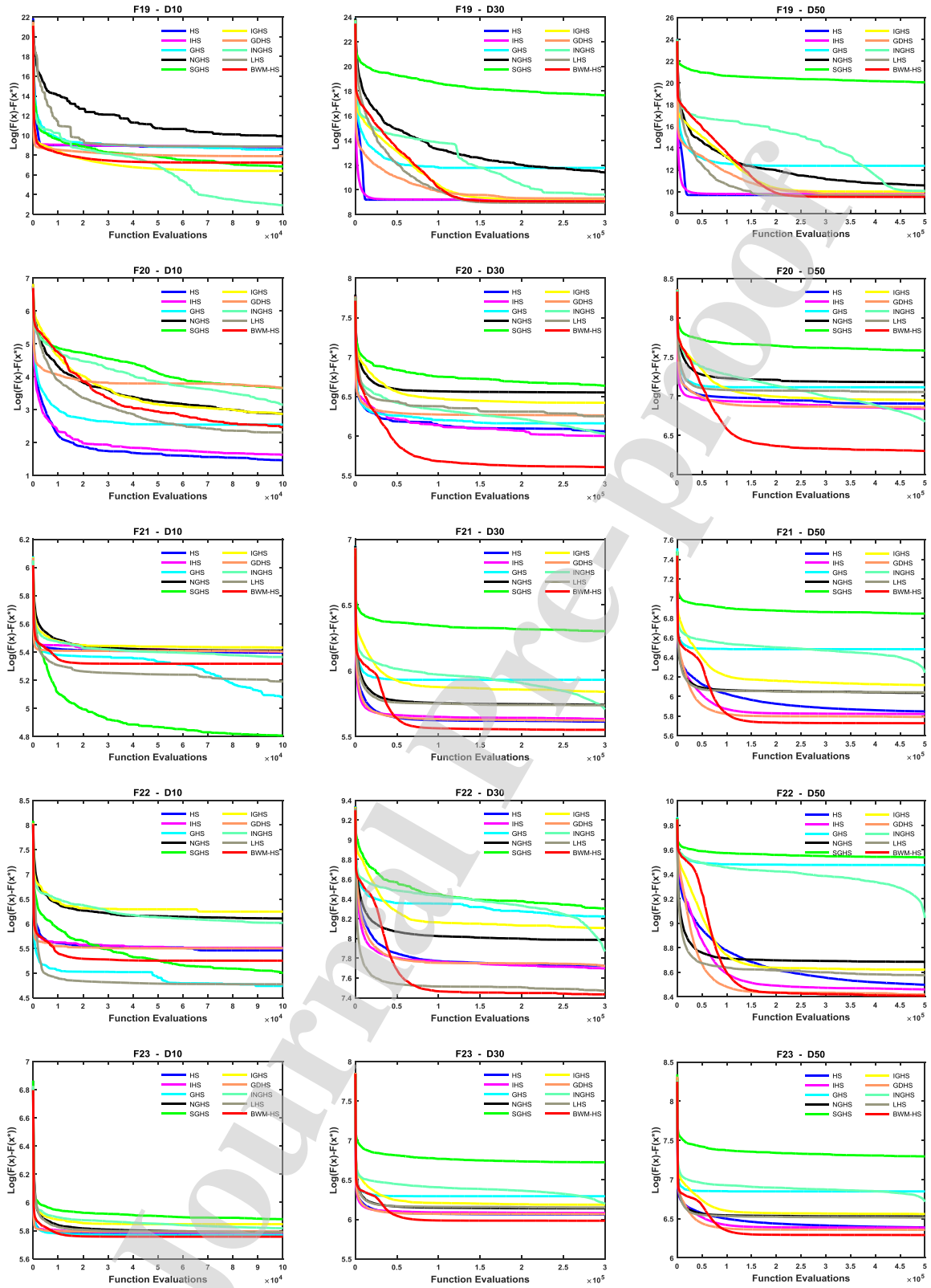
The proposed algorithm performs well on composition functions and obtained better mean values than other algorithms for *f23*, *f25*, *f27*, and *f30* on 10D; for all composition functions except *f25* and *f30* on 30D; and all composition functions on 50D. Since these functions have various shapes in different areas of the search space, the reason for using them is to investigate the ability of algorithms to create a proper balance between exploration and exploitation capabilities. For this purpose, BWM_HS employs the aforementioned self-adaptive selection scheme that dynamically changes the parameter *Etha* in descending order as the number of iterations increases. Based on the value of this parameter, the scheme probabilistically decides between selecting the *best* or the *worst* rules that exhibit different search behaviors. In this way, it maintains a proper balance between exploration and exploitation abilities during the search process.

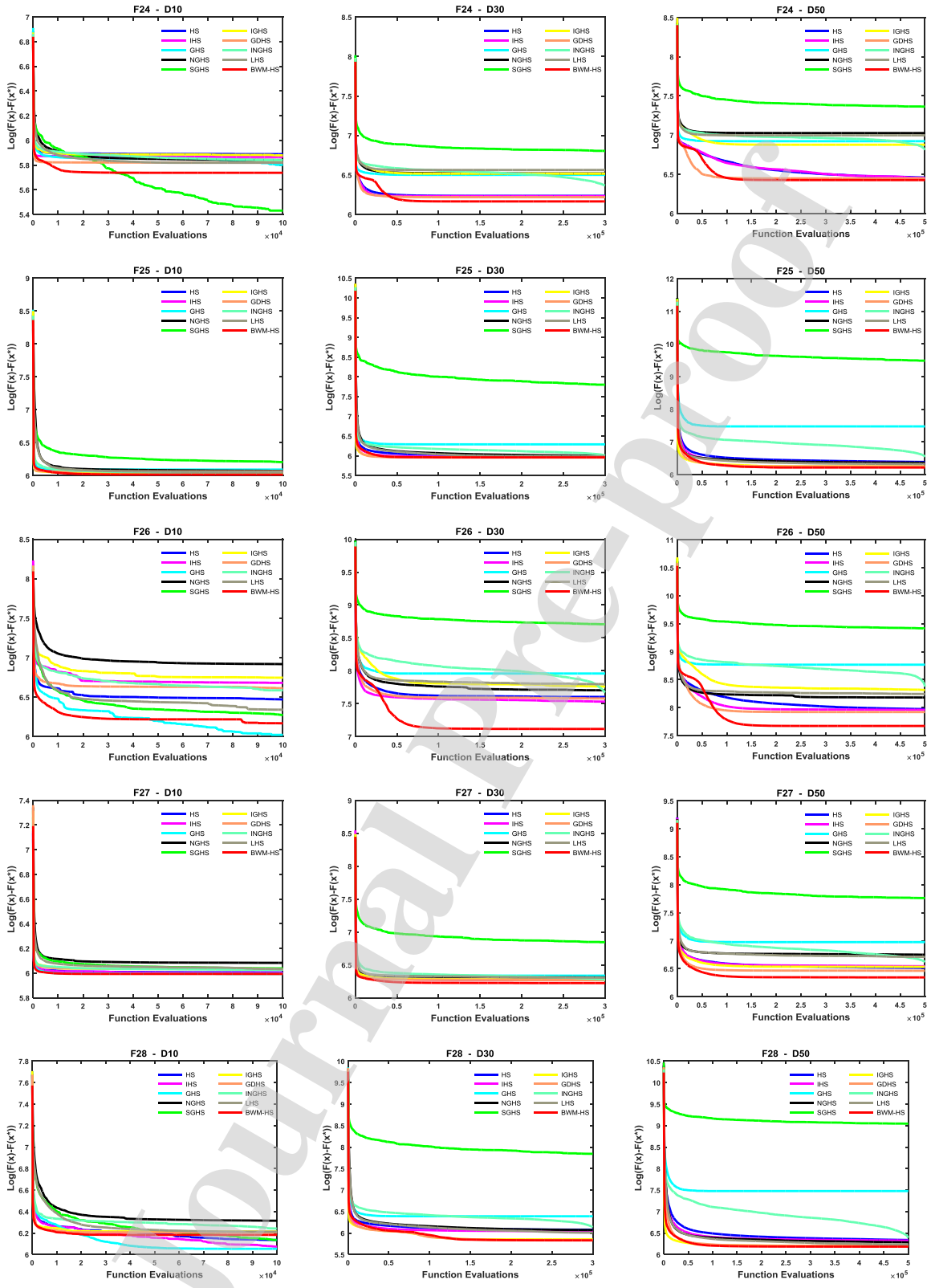












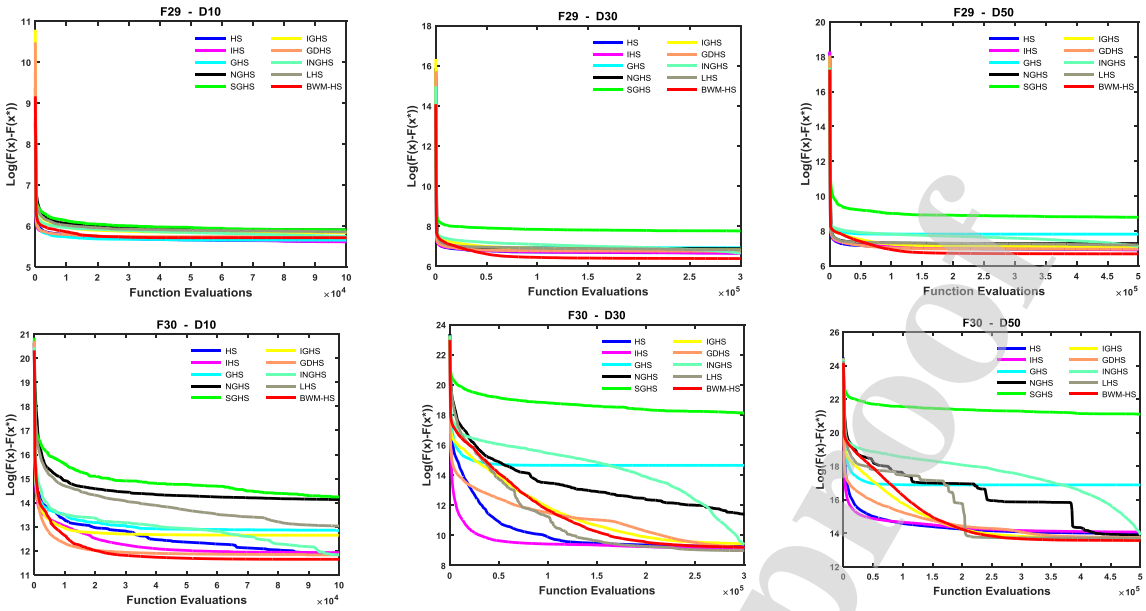


Fig 5: The convergence rate graphs in terms of the logarithmic scale of the mean value of CEC 2017 benchmark functions

Fig. 5 shows graphs corresponding to the logarithmic scale of the mean error values for solving the benchmark functions by the considered algorithms in terms of the number of function evaluations. This figure shows that for some functions, including 10D of *f1*, *f12*; 30D of *f2*, *f3*, *f10*; and 50D of *f1*, *f2*, *f3*, and *f12* the BWM_HS has outperformed the competitive algorithms because of better local search ability at the end of optimization process provided by the *mean* rule and the pitch adjustment process. To explain further, at the final stage of search, almost all harmonies have moved toward the best harmony, hence the mean of harmonies in HM is close to the best harmony in the search space. As a result, this rule (Eq. (4)) can enhance the neighborhood exploitation of the current best harmony and creates more chances to find a potential better harmony. Furthermore, at these stages, the BW has small values and the PAR has large values that would result in further enhancement of local exploitation ability and so the improvement of final solutions.

Furthermore, for some other functions including 10D of *f5*, *f7*, *f8*, *f23*, *f30*; 30D of *f5*, *f7*, *f8*, *f9*, *f16*, *f17*, *f21*, *f22*, *f23*, *f24*, *f26*, *f27*, *f28*, *f29*; and 50D of *f5*, *f7*, *f8*, *f16*, *f17*, *f21*, *f23*, *f25*, *f26*, *f27*, *f28*, *f29*, and f30 the BWM_HS has shown a better performance by escaping from local optima at the early stages of optimization process thanks to the proper balancing of self-adaptive selection mechanism. Most of these functions are either multimodal or composition functions that have a huge number of local optimal. Thus, a well-balanced optimization algorithm requires a strong exploration at initial iterations, a mechanism to escape from local optimal at final iteration. As mentioned before, the BWM_HS algorithm always generates one new harmony by the *mean* rule. Thus, at initial iterations that all harmonies are widely distributed in the search space, this rule is in favor of exploration by searching a wide region between randomly selected harmonies and the center of all harmonies. Furthermore, the second new harmony is produced through a probabilistic selection between the *worst* and *mean* rules. At the early stages of search, the *Etha* parameter has large values that gives more chance to the best rule (Eq. (2)). However, in these stages, the diversity between harmonies is

high so the best rule will generate new harmonies that explore the best solution. In brief, the *mean* and *best* rules enhance the exploration ability of the BWM_HS at initial iterations. On the other hand, at final iterations, the *Etha* parameter has large values so the *worst* rule (Eq. (3)) should get considerably more opportunity than the *best* rule for selection. Therefore, the new harmony generated by this rule will help the BWM_HS to escape from local optimal at final stages of search.

Finally, for some functions such as 10D of *f4*, *f6*, *f9*; 30D of *f6*, *f10*, *f20*, *f22*; and 50D of *f6*, *f9*, *f10*, *f12*, *f20*, and *f30* the performance of BWM_HS is superior to the compared algorithms. Again, as explained above, this is mainly because of proper trade between exploration and exploration that helps the BWM_HS to avoid local optima during the search process and also due to improved exploitation ability that gives it a high potential to improve the quality of final solutions.

To sum up, by considering Tables 4-6 and Fig.5, the proposed BWM_HS algorithm is superior to the other recent variants of HS in terms of convergence speed, robustness, and solution accuracy.

4.1.8. Time complexity

The time complexity of competing algorithms are calculated according to the guidelines provided in the definition document of CEC 2017 test suite [38]. Table 7 shows the results of parameters T0, T1, and $\widehat{T2}$ that are used for calculating the time complexity of competing algorithms according to these guidelines. The parameter T0 is the computing time of the test program defined in [38]. T1 denotes the computing time of an algorithm for 2×10^5 function evaluations of *f*18 (in CEC 2017) with a defined D dimension. The average computing time for five runs of the algorithm with 2×10^5 evaluations of the same D dimensional *f*18 is $\widehat{T2}$. By considering these parameters, the time complexity of algorithms is estimated by ($\widehat{T2}$ − T1)/T0.

Table 7. Complexity time (in second)

Algorithms	HS	IHS	GHS	NGHS	SGHS	IGHS	GDHS	INGHS	LHS	BWM_HS
10D, T0 = 0.098, T1 = 1.465										
	12.345	13.116	19.211	20.109	18.670	50.981	30.005	33.237	9.313	14.140
	111.02	118.88	181.08	190.24	175.56	505.26	291.22	324.20	80.08	129.33
Rank	2	3	6	7	5	10	8	9	1	4
30D, T0 = 0.098, T1 = 1.709										
	40.186	39.441	69.820	76.318	40.238	149.22	81.365	101.38	25.615	60.103
	392.62	385.02	695.01	761.31	393.15	1505.2	812.81	1017.1	243.93	595.85
Rank	3	2	6	7	4	10	8	9	1	5
50D, T0 = 0.098, T1 = 2.072										
	75.811	71.742	103.28	115.22	59.332	245.35	155.43	177.19	42.837	97.102
	752.43	710.91	1032.7	1154.6	584.28	2482.4	1564.9	1786.9	415.96	969.69
Rank	4	3	6	7	2	10	8	9	1	5

Mean Rank	3	2.66	6	7	3.66	10	8	9	1	4.66
Final Rank	3	2	6	7	4	10	8	9	1	5

As per the data presented in the last row of Table 7, the BWM_HS algorithm is ranked fifth and the other algorithms can be sorted in ascending order by their average ranks as follows: GHS, NGHS, GDHS, INGHS and IGHS. Thus, BWM_HS has less computation time than half of the comparative algorithms. However, it presented more computation time than four other algorithms that can be due to the computations that it needs to find the best harmony or calculate the mean of all harmonies in HM which are used in the *best* and *mean* rules, respectively.

4.1.9. Computational complexity

The order of computational complexity demonstrate the computational efficiency of algorithms [77]. In this subsection, the computational complexity is calculated in terms of O notation for the process of initialization, finding the best and worst harmonies, improvisation and update of BWM_HS algorithm. The computational complexity for initialization of HM is O(HMS×D), where D is the number of decision variables of optimization problem. The computational complexity for obtaining the best and worst harmonies in the current HM that are used in the best and worst rules is O(2×HMS) which is equivalent to O(HMS). The computational complexity of improvising process is O(2×D) which is equivalent to O(D). On the other hand, the computational for updating process in the worst case is O(HMS). Note that the algorithm complexity for initialization of parameters, computing self- adaptive control parameter, selecting the better harmony between two new generated harmonies, and calculating the mean of harmonies in HM are O(1). Therefore, if the maximum number of iterations is equal to MaxIt, then the computational complexity of the BWM_HS will be less than O(HMS×D) + O((HMS+D+HMS) ×MaxIt). In summary, it can be argued that the computational complexity of the BWM_HS algorithm is equal to O(HMS×D) + O((HMS+D)×MaxIt), which is similar to the computational complexity of the canonical HS.

4.2. Application of the proposed algorithm for clustering problems

Nowadays, the amount of raw data that are stored in various domains of science and technology increases continuously. This fact has attracted the attention of many researchers in data mining field for designing techniques that discover and extract the meaning of data more efficiently. Data clustering is one of the traditional data mining methods [78, 79] that identifies similar groups of data in a data set based on a predefined similarity measurement. In fact, clustering task can be considered as an optimization problem that tries to minimize intra cluster cohesion and maximize extra cluster separation. In general, clustering algorithms can be divided into hierarchical and partitional clustering categories [80]. Agglomerative and divisive modes are two approaches that are applied in hierarchical clustering [81]. More precisely, in the agglomerative mode each data point is taken in a separate cluster in the beginning and then two most similar clusters are merged at each step. In the divisive mode, all data are considered as one cluster in the beginning and then each cluster is divided into two clusters until termination criteria are reached [82]. Partitional clustering can be divided

into two categories: crisp and fuzzy clustering. In the crisp clustering each data point belongs to only one cluster but in fuzzy clustering, by using fuzzy membership function, each data point is allowed to belong to more than one cluster at the same time [83]. The K-means algorithm [84] and fuzzy c-means (FCM) [85] are the most popular algorithms used in partitional clustering. Although these techniques have been employed in many science domains including machine learning [86], pattern recognition [87], image processing [88], and bioinformatics [89] they require a prior knowledge about the number of clusters [90] and also trap in local optima [91] are two major drawbacks of partitional clustering. To cope these drawbacks, evolutionary optimization algorithms such as PSO [92], HS [93],artificial bee colony (ABC) [94], CS [95], and ant colony (ACO) [96] have shown promising methods and a good literature survey on the early works can be found in [97].

The following subsections illustrate how the proposed BWM_HS algorithm is employed for solving clustering problems when the number of clusters is known. First, a clustering problem is formally defined in Subsection 4.2.1. Then, Subsection 4.2.2 explains how a candidate solution for a clustering problem is encoded as a harmony vector in BWM_HS algorithm. Thereafter, Subsection 4.2.3 describes an objective function to measure the merits of harmonies. Next, in Subsection 4.2.4 the performance of BWM_HS algorithm is investigated against several benchmark data clustering problems and is compared with two groups of algorithms. The experimental setup for these two groups of algorithms is presented in Subsection 4.2.5. Finally, experimental results and discussion about the performance of first and second groups of algorithms are presented in Subsection 4.2.6 and Subsection 4.2.7, respectively.

4.2.1. Formal definition of a clustering problem

In general, a clustering problem is defined as follows [97]. Suppose $X = \{o_1, \dots, o_n\}$ is a set of n patterns in a d dimensional space. Then, clustering X is a process of partitioning X into k clusters $\{C_1, \dots, C_k\}$ where the following conditions must hold:

1. Union of clusters must be equal to the original data set, that is: $\forall i, j \in [1, k]$, $\bigcup_{j=1}^{k} C_j = X$
2. Each pattern must belong to one cluster, that is: $\forall i, j \in [1, k]$, $C_i \cap C_j = \emptyset$
3. There can be no empty clusters, that is: $\forall j \in [1, k]$, $C_j \neq \emptyset$

4.2.2. Encoding a candidate solution

For the purpose of data clustering by the BWM_HS algorithm, each candidate solution must represent a possible partitioning of a dataset into K subsets. In this study, the i-th solution for a dataset with dimension d is encoded as $(C_{i1}, C_{i2}, \dots, C_{iK})$ where C_{ij} stands for a *d-dimensional* center of the j-th subset. Thus, each solution contains $K \times d$ components as shown in Fig. 6.

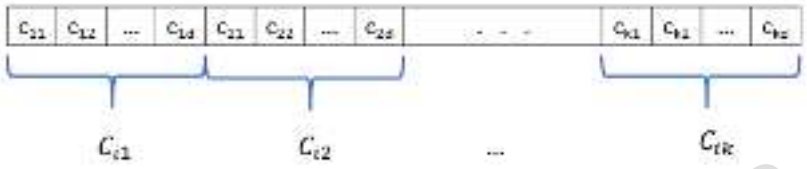


Fig 6: Presentation of a candidate solution as a harmony vector for k clusters and d features

4.2.3. Objective function

The BWM_HS algorithm starts solving a data clustering problem by a random initialized population of candidate solutions. Then, it employs an objective function in the form of a cost function as a guide for evolving these candidate solutions toward the optimal solution which is a solution with minimum value for the objective function. For the purpose of calculating $f_i$as the cost of a candidate solution i, first each data point in the dataset is assigned to the closest center encoded in this solution. Then, assuming that $C_{in}$is the n-th center of the solution i and X_j^nrepresents the j-th data point assigned to the C_{in}:

$$f_i = \sum_{n=1}^{K} \sum_{j=1}^{N} \left\| X_j^n - C_{in} \right\|^2 \qquad (10)$$

In other words, the BWM_HS algorithm uses the above objective function to search for a solution that has the minimum distance between data the points in the dataset and their corresponding centers.

Table 8.
Datasets used to evaluate the performance of BWM_HS algorithm on the clustering problems.

Data	Instance	Attribute	Number of cluster
Iris	150	4	3
CMC	1473	9	3
Wine	178	13	3
Breast Cancer	683	9	2
Vowel	871	3	6
Glass	214	9	6
Aggregation	788	2	7
D31	3100	2	31
R15	600	2	15
Balance	624	4	3

4.2.4. Performance evaluation on data clustering problems

The performance of BWM_HS algorithm on data clustering problems is verified by comparing against two groups of algorithms. The first group includes the original HS and its eight recent variants mentioned in Subsection 4.1.5. The second group is composed of ten well-known algorithms in the literature, namely K-mean [98], K-mean++ [99], PSO [100], GA [101], SA [102], TS [103], ACO [96], HBMO [104], KH [105], IKH [106]. All experiments are evaluated on ten datasets including six real world and four synthetic datasets.

The real world datasets are Iris, Wine, Glass, Breast Cancer, CMC, and Vowel that are collected from machine learning laboratory of California University [107]. Synthetic datasets are Aggregation, D31, R15, and Balance data which are available in [108]. More details about these datasets, including the number of instances, the number of features, and the number of clusters are shown in Table 8.

4.2.5. Experimental setup

Parameters configuration for the first group are similar to those presented in Subsection 4.1.2, except that the maximum number of function evaluations is set to 10000. Furthermore, 51 independent runs of BWM_HS and its variants are performed on these datasets. Since the results of the compared algorithms in the second group are directly taken from [106], so for a fair judgment, 100 independent runs of the BWM_HS algorithm are performed.

4.2.6. Experimental results for the first group of algorithms

Statistical results achieved for the first group of algorithms are presented in Table 9 in terms of best, worst, mean, and standard deviation of objective function values on each dataset. Furthermore, the statistical ranking of each considered algorithm based on the mean values of objective function for each dataset is reported. For the sake of clarity, the best results are shown in boldface.

Table 9.
The statistical results of BWM_HS and the first group of algorithms obtained for data clustering problems.

Data set	Algorithm	HS	IHS	GHS	NGHS	SGHS	IGHS	GDHS	INGHS	LHS	BWM_HS
Iris	**Best**	**96.6555**	**96.6555**	99.5505	96.71345	123.1486	**96.6555**	**96.6555**	96.7083	**96.6555**	**96.6555**
	Worst	96.6688	96.6686	110.4516	121.3898	149.6680	120.7233	120.7233	121.0103	97.5736	**96.6555**
	Mean	96.6561	96.6557	103.9093	98.0114	138.8485	97.5993	97.1274	99.1508	97.1307	**96.6555**
	S.D	0.00260	0.00184	2.866919	3.426370	6.005988	4.718242	3.370173	6.440808	0.36338	**0.00000**
	Rank	3	2	9	7	10	6	4	8	5	1
Wine	**Best**	16293.18	16292.37	16350.57	16294.07	16436.31	16292.18	16292.21	16298.30	16292.21	**16292.18**
	Worst	16303.29	16294.58	18592.58	18044.45	17194.41	16294.17	16292.76	16376.80	16297.62	**16292.67**
	Mean	16296.34	16292.82	16902.25	16378.72	16778.09	16292.49	16292.41	16304.20	16293.73	**16292.33**
	S.D	2.424883	0.430584	527.6493	267.4371	154.5660	0.542013	0.227240	10.69181	1.122733	**0.221830**
	Rank	6	4	10	8	9	3	2	7	5	1
Glass	**Best**	219.570	219.219	327.270	211.228	396.134	210.429	211.427	240.771	**210.710**	210.720
	Worst	252.538	250.130	407.647	258.145	475.400	243.438	244.484	306.125	**243.817**	244.800
	Mean	232.936	231.633	361.647	226.382	433.800	228.540	226.240	269.458	**224.308**	225.025
	S.D	10.0134	**9.04800**	20.9674	12.2717	18.6445	11.4279	10.8157	13.3741	11.2294	11.2876
	Rank	7	6	9	4	10	5	3	8	1	2
Cancer	**Best**	2964.388	2964.387	3010.327	2965.540	3641.002	**2964.387**	**2964.387**	2965.876	2964.393	**2964.387**
	Worst	2964.694	2964.413	3189.732	3002.029	4668.892	**2964.387**	**2964.387**	2975.599	2965.953	**2964.387**
	Mean	2964.402	2964.390	3085.664	2970.728	4095.102	**2964.387**	**2964.387**	2969.335	2964.558	**2964.387**
	S.D	0.043281	0.004853	41.05577	7.546165	233.2559	0.000002	**0.000000**	2.351176	0.253660	**0.000000**
	Rank	5	4	9	8	10	1	1	7	6	1
CMC	**Best**	5532.664	5532.420	5755.254	5534.659	6268.122	**5532.185**	**5532.185**	5556.540	5532.218	**5532.185**
	Worst	5550.224	5534.655	6381.004	5657.507	6793.977	**5532.185**	**5532.185**	5665.435	5543.602	**5532.185**
	Mean	5535.763	5532.893	5971.885	5550.110	6512.789	**5532.185**	**5532.185**	5597.175	5534.142	**5532.185**
	S.D	3.671972	0.438970	139.4013	22.24325	125.4466	0.000003	0.000001	28.37201	2.588881	**0.000000**
	Rank	6	4	9	7	10	1	1	8	5	1

Vowel	**Best**	149114.5	148967.3	161267.4	149513.6	188598.5	**148967.2**	**148967.2**	153012.9	149021.6	**148967.2**
	Worst	157222.3	156320.3	187724.9	161242.9	215113.1	158471.7	153051.9	185157.7	158673.5	**150817.5**
	Mean	150218.7	150398.9	173352.5	151581.2	202059.9	149837.7	149872.3	164492.8	150873.5	**149360.8**
	S.D	1508.637	1712.748	5538.420	2424.059	5829.390	1540.346	1541.246	8247.124	1192.501	**502.0275**
	Rank	4	5	9	7	10	2	3	8	6	1
D31	**Best**	3626.3350	3728.9029	4787.0306	2923.3434	5403.0420	2882.0209	2882.0212	5064.6063	2934.3696	**2882.0208**
	Worst	4225.9637	4151.5161	5593.7272	3220.2086	5868.9743	3179.6179	3191.9124	6014.4814	6071.2646	**3173.2904**
	Mean	3912.9181	3872.9798	5262.6100	3063.6365	5697.6216	3000.1455	2999.1614	5615.5771	3536.0342	**2979.1903**
	S.D	122.90172	96.850913	182.72410	70.117813	89.901140	85.438441	89.909183	194.44823	1043.8854	**83.628206**
	Rank	7	6	8	4	10	3	2	9	5	1
R15	**Best**	233.25693	230.31254	462.47115	225.52039	616.73578	224.59843	224.59841	392.92072	224.67022	**224.59840**
	Worst	283.21533	277.58897	602.81221	322.98580	729.86294	344.25352	305.28612	608.94598	305.72322	**300.15652**
	Mean	253.41651	246.01644	524.40662	255.47022	670.08528	256.00883	245.53037	488.95534	243.12209	**242.08500**
	S.D	**14.098716**	14.431809	33.746656	25.616851	25.797841	27.682396	19.676761	47.889389	25.059237	21.471147
	Rank	6	4	9	7	10	8	3	9	2	1
Aggregation	**Best**	2712.0074	2712.0074	2738.4291	2712.5086	2977.8366	2712.0074	2712.0074	2715.9806	2712.4115	**2712.0074**
	Worst	**2783.7565**	2783.3499	2934.1774	2890.0433	3261.0260	2869.1896	2882.5309	2874.8595	2883.0967	2869.1896
	Mean	2730.1224	2729.9584	2810.3702	2746.4873	3126.0313	2736.2298	2735.3683	2740.0253	2735.1545	**2724.2048**
	S.D	**19.426654**	22.856100	38.173120	39.282482	72.431310	31.019883	30.845703	30.590973	29.436781	28.554195
	Rank	3	2	9	8	10	6	5	7	4	1
Balance	**Best**	**1423.8204**	**1423.8204**	1424.8956	1424.1737	1431.1051	**1423.8204**	**1423.8204**	1424.0024	1423.8367	**1423.8204**
	Worst	1426.3144	1426.0864	1436.4003	1435.2369	1449.0564	1425.7235	**1425.6639**	1429.3443	1435.4351	**1425.6639**
	Mean	1425.3687	1425.0205	1429.7470	1428.3691	1441.4862	1424.2321	1424.1689	1426.0024	1426.3599	**1423.9039**
	S.D	0.805825	0.90241	3.094692	3.024332	3.648116	0.752749	0.702669	1.239655	3.172472	**0.35376**
	Rank	**5**	4	9	8	10	3	2	6	7	1
Mean rank		5.19	4.09	9	6.79	9.82	3.74	2.59	7.69	4.57	1.1
Final rank		6	4	9	7	10	3	2	8	5	1

Table 9 clearly shows that the BMW_HS algorithm is superior to other algorithms. More precisely, according to all criteria, the BWM_HS has obtained either better or similar results compared with other algorithms for Iris, Wine, Cancer, CMC, Vowel, D31, and Balance datasets. Furthermore, the BWM_HS has yielded better or similar results for the mean criterion on all datasets except the Glass dataset where it has the second rank for this criterion. Finally, the last row of this table indicates that the BWM_HS has outperformed the compared algorithms by earning the first rank for mean criterion while the rank of other algorithms can be sorted in the following order: GDHS, IGHS, IHS, LHS, HS, NGHS, INGHS, GHS, and SGHS.

Fig. 7 shows convergence graphs corresponding to the objective function values for these algorithms in terms of the number of function evaluations for each dataset. This figure indicates that for Iris dataset BWM_HS, IHS, IGHS, and GDHS have the same convergence trends in the early stages of optimization process, but later the BWM_HS outperforms the others through better exploitation behavior. For Wine dataset, all algorithms start optimization with almost similar convergence behavior, however, at the end the BWM_HS, IGHS, and GDHS achieve the smallest mean objective function value. For Glass dataset, GDHS has a faster convergence than the others in the early stages of search, but at final stages the BWM_HS performs with better exploitation and so it gains the second rank. For Cancer and CMC datasets, BWM_HS, GDHS, and IGHS have the same convergence trend through the optimization process and show a better performance than the other algorithms. For Vowel dataset, GDHS has a faster convergence than the BWM_HS at the beginning, but as the search process proceeds, the BWM_HS presents better exploitation ability and yields the lowest mean objective function value. For D31 and R15 datasets, BWM_HS balances

properly between exploration and exploitation and reaches to the smallest mean objective function value. Finally, for Balance and Aggregation datasets, IGHS shows a faster convergence than the others at the start of search process but at the end the BWM_HS finds better solutions.

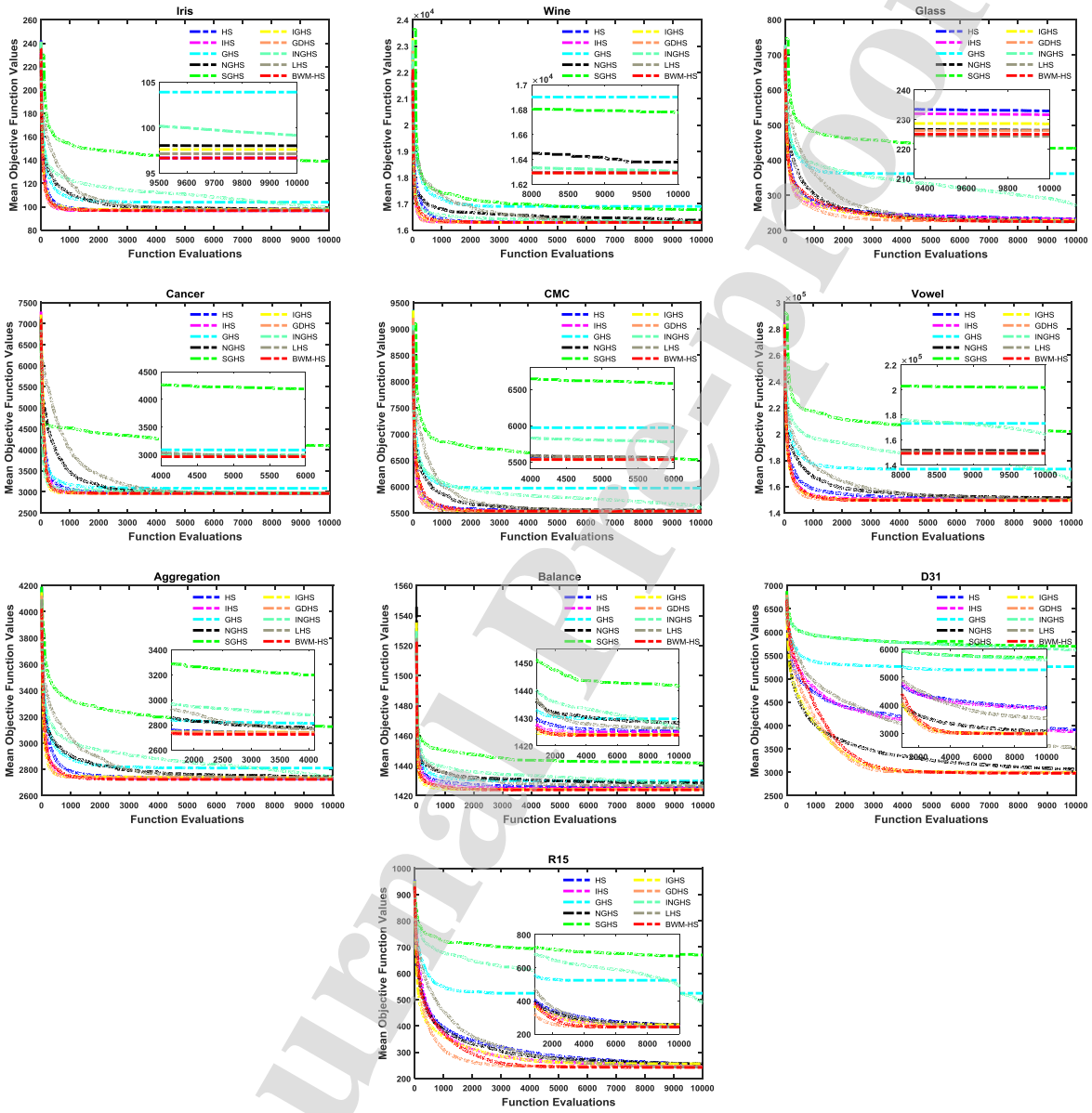


Fig 7: The convergence rate graphs in terms of the mean objective function values of clustering problems obtained by HS variants

4.2.7. Experimental results for the second group of algorithms

This subsection compares the results obtained by BWM_HS for Iris, Wine, Glass, Cancer, CMC, and Vowel datasets with the results of ten well-known algorithms in the literature. As Table. 10 presents, the statistical results are in terms of best, worst, mean, and standard deviation of objective function values on each dataset.

Table 10.
The statistical results of BWM_HS and second group of algorithms for data clustering problems.

Data set	Algorithms	K-mean	K-mean++	GA	SA	TS	ACO	HBMO	PSO	KH	IKH	BWM_HS
Iris	**Best**	97.3259	97.3259	113.9865	97.4573	97.3659	97.1007	97.7520	96.8942	**96.6555**	**96.6555**	**96.6555**
	Worst	123.9695	122.2789	139.7783	102.0100	98.5695	97.8085	97.7576	97.8973	**96.6555**	**96.6555**	**96.6555**
	Mean	106.5766	98.5817	125.1970	99.9570	97.8680	97.1715	96.9532	97.2328	**96.6555**	**96.6555**	**96.6555**
	S.D	12.938	5.578	14.563	2.01	0.53	0.367	0.531	0.34716	1.9e-06	9.8e-06	**0.00000**
	Rank	10	8	11	9	7	5	4	6	1	1	1
Wine	**Best**	16555.68	16555.68	16530.53	16473.4	16666.23	16530.53	16357.28	16345.97	**16292.19**	16292.21	**16292.18**
	Worst	18294.85	18294.85	16530.53	18083.25	16837.54	16530.53	16357.28	16562.32	18293.6	16294.30	**16292.67**
	Mean	17251.35	16816.55	16530.53	17521.09	16785.45	16530.53	16357.28	16417.47	16579.66	16292.85	**16292.33**
	S.D	874.148	637.140	**0**	753.084	52.073	**0**	**0**	85.4974	424.9147	0.706742	0.221830
	Rank	10	9	6	11	8	5	3	4	7	2	1
Glass	**Best**	215.73	215.36	282.32	275.16	283.79	273.46	247.71	270.57	**210.242**	210.252	210.720
	Worst	227.35	223.71	286.77	287.18	286.47	280.08	249.54	283.52	251.274	**222.800**	244.800
	Mean	218.70	217.56	278.37	282.19	279.87	269.72	245.73	275.71	**215.722**	215.935	225.025
	S.D	2.456	2.455	4.1387	4.238	4.1927	3.5848	**2.4381**	4.5571	5.44876	2.73792	11.2876
	Rank	4	3	9	11	10	7	6	8	1	2	5
Cancer	**Best**	2988.43	2986.96	3249.46	2993.45	3251.37	3046.06	3112.42	2973.50	**2964.387**	**2964.387**	**2964.387**
	Worst	2999.19	2988.43	3427.43	3421.95	3434.16	3242.01	3210.78	3318.88	3580.312	2964.393	**2964.387**
	Mean	2988.99	2987.9	2999.32	3239.17	2982.84	2970.49	2989.94	3050.04	2971.977	2964.389	**2964.387**
	S.D	2.469	0689	229.734	230.192	232.217	90.5002	103.471	110.8013	62.26148	0.001258	**0.000000**
	Rank	7	6	9	11	5	3	8	10	4	2	1
CMC	**Best**	5703.20	5703.20	5756.598	5849.03	5993.594	5819.135	5713.980	5700.985	5693.72	5693.72	**5532.185**
	Worst	5704.57	5705.37	5812.648	5966.94	5999.805	5912.430	5725.350	5923.249	6755.956	5693.78	**5532.185**
	Mean	5705.37	5704.19	5705.630	5893.48	5885.062	5701.923	5699.267	5820.965	5737.234	5693.735	**5532.185**
	S.D	1.033	0.955	50.3694	50.867	40.84562	45.63470	12.69000	46.95969	178.0245	0.007975	**0.000000**
	Rank	6	5	7	11	10	4	3	9	8	2	1
Vowel	**Best**	149398.6	149394.5	159153.4	149370.4	162108.5	159458.1	161431.0	148976.0	**148976.2**	**148967.2**	**148967.2**
	Worst	162455.7	161845.5	165991.6	165986.4	165996.4	165939.8	165804.7	**149121.2**	158503.0	158600.5	150817.5
	Mean	151987.9	151445.3	149513.7	161566.3	149468.3	149935.6	149201.6	**148999.8**	150035.9	150173.4	149360.8
	S.D	3425.250	3119.751	3105.544	2847.085	2846.235	3485.382	2746.042	**28.81347**	1707.842	1732.452	502.0275
	Rank	10	9	5	11	4	6	2	1	7	8	3
Mean rank		7.83	6.66	7.83	10.66	7.33	5	4.33	6.33	4.66	2.83	2
Final rank		10	7	9	11	8	5	3	6	4	2	1

The experimental results given in this table show that the BWM_HS algorithm has won the competition for clustering the six mentioned datasets. More precisely, the BWM_HS, KH, and IKH have obtained the best statistical results for Iris dataset according to the best, worst, and mean criteria while the BWM_HS has proved to be a more stable algorithm than the KH and IKH by yielding a better statistical result for the standard deviation criterion. In addition to Iris dataset, the BWM_HS has succeeded to achieve better results than the other considered algorithms according to all criteria for Cancer and CMC datasets, particularly it has significantly improved the statistical results for clustering CMC. Furthermore, for Wine dataset, the BWM_HS has obtained either better or equal results on all criteria except the standard deviation. For Vowel datasets, the BWM_HS has won the competition only on the best criterion but it has lost according to all other criteria to PSO algorithm. Generally, as the last row of Table 9 indicates, the BWM_HS has outperformed other compared algorithms by earning the first rank for the mean criterion while IKH, HBMO, KH, ACO, PSO, k-mean++, TS, GA, k-mean, and SA have achieved the second to twelfth ranks, respectively.

5. Conclusion

This paper proposes the BWM_HS algorithm as a new variant of HS that utilizes more efficiently the valuable information stored in HM to guide the search process. For this purpose, the BWM_HS uses a modified memory consideration procedure as the random harmony selection scheme is replaced with three novel pitch selection and production rules. Two new harmonies are generated at each iteration of BWM_HS algorithm by applying these three rules to further utilize the information of HM. To generate a pitch, except one rule that is deterministically selected, other two rules are chosen by a probabilistic self-adaptive selection scheme to balance between the exploration and exploitation abilities during the search course. Of note that the proposed BWM_HS maintains the simple framework of canonical HS so it is straightforward to implement. The effect of the three pitch production rules on the performance of BWM_HS is investigated by building six modified versions of BWM_HS where each version includes a different combination of these pitch production rules. Then, an experiment with 51 independent runs of the BWM_HS and its six modified versions is conducted on 10D, 30D and 50D benchmark functions of CEC 2017 test suite that in general, BWM_HS produce better results than other versions for optimization of 53.3% of the functions. These statistical results clearly show the superiority of BWM_HS on its six variants according to the accuracy, stability, and convergence rate. So, the best possible combination of pitch production rules is the one that includes all proposed rules and eliminating any of these rules will decrease the performance of the resulted algorithm.

Another experiment is performed that reports the statistical results of 51 independent runs of BWM_HS with different configuration of HMS including 5, 10, 20, 40, 80, 150, and 300 on 30D benchmark functions of CEC 2017. The aim of this experiment is to evaluate the effect of HMS on the performance of BWM_HS since as the HMS is increased the diversity of HM is improved but the convergence speed is decreased. The results indicate that the BWM_HS presents the best performance when HMS is set to 5 and achieves better results than other HMS values for optimization of 26.6% of the functions.

The third experiment compares the performance of BWM_HS with eight recent variants of HS algorithms including IHS, GHS, NGHS, SGHS, IGHS, GDHS, INGHS, and LHS. Again, the experiment involves 51 independent runs of the BWM_HS and the eight variants of HS on 10D, 30D, and 50D benchmark functions of CEC 2017.The statistical results are reported in terms of best, worst, mean, and standard deviation of error function values obtained in all runs. Furthermore, some graphs corresponding to the logarithmic scale of the mean error values in terms of the number of function evaluations for the compared algorithm are drawn for some functions. Furthermore, the Wilcoxon signed-rank test with the significant level of 0.05 is carried out on the mean results to perform a pair-wise comparison between the BWM_HS and its eight variants of HS. Gathered statistical results over all dimensions prove that BWM_HS outperforms HS, IHS, GHS, NGHS, SGHS, IGHS, GDHS, INGHS, and LHS for the optimization of 77.7%, 74.2%, 90%, 100%, 92.2%, 68.8%, 54.4%, 84.4%, and 77.7% of the functions, respectively. All these confirm that the BWM_HS outperforms the competitive algorithms according to accuracy, stability, and convergence speed criteria.

Finally, the BWM_HS algorithm is employed for the purpose of data clustering to investigate its capability for solving real world optimization problems. To this end, first the BWM_HS is

applied on ten benchmark datasets and its obtained results are compared with HS and eight state-of-the-art variants of HS. It is observed that BWM_HS produces better mean objective functions values than other versions for clustering of 90% of the datasets. Second, its results for six benchmark datasets are compared with the results of ten well-known clustering algorithms in the literature. The results are reported as the mean, best, worst, and standard deviation of objective function value that is calculated as the distance between data points in a dataset and their corresponding centers. In this case, BWM_HS outperforms other well-known algorithms for clustering of 66.6% of the datasets. These results show the superiority of the BWM_HS to the compared algorithms.

References

[1] W. Kuo, M.J. Zuo, Optimal reliability modeling: principles and applications, John Wiley & Sons, 2003.
[2] A. Gábor, J.R. Banga, Robust and efficient parameter estimation in dynamic models of biological systems, BMC systems biology, 9 (2015) 74.
[3] F.A. Fernandes, Polymerization kinetics of Fischer-Tropsch reaction on iron based catalysts and product grade optimization, Chemical Engineering & Technology: Industrial Chemistry-Plant Equipment-Process Engineering-Biotechnology, 28 (2005) 930-938.
[4] S.V. Aksenov, B. Church, A. Dhiman, A. Georgieva, R. Sarangapani, G. Helmlinger, I.G. Khalil, An integrated approach for inference and mechanistic modeling for advancing drug development, FEBS letters, 579 (2005) 1878-1883.
[5] F.A. Tillman, C.-L. Hwang, W. Kuo, Optimization Techniques for System Reliability with RedundancyЧA Review, IEEE Transactions on Reliability, 26 (1977) 148-155.
[6] A. Miró, C. Pozo, G. Guillén-Gosálbez, J.A. Egea, L. Jiménez, Deterministic global optimization algorithm based on outer approximation for the parameter estimation of nonlinear dynamic biological systems, BMC bioinformatics, 13 (2012) 90.
[7] T.A. Adams II, W.D. Seider, Practical optimization of complex chemical processes with tight constraints, Computers & Chemical Engineering, 32 (2008) 2099-2112.
[8] L. Jourdan, M. Basseur, E.-G. Talbi, Hybridizing exact methods and metaheuristics: A taxonomy, European Journal of Operational Research, 199 (2009) 620-629.
[9] H. Rakhshani, A. Rahati, E. Dehghanian, Cuckoo search algorithm and its application for secondary protein structure prediction, in: 2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI), IEEE, 2015, pp. 412-417.
[10] S. Ghambari, A. Rahati, An improved artificial bee colony algorithm and its application to reliability optimization problems, Applied Soft Computing, 62 (2018) 736-767.
[11] B. Subudhi, D. Jena, A differential evolution based neural network approach to nonlinear system identification, Applied Soft Computing, 11 (2011) 861-871.
[12] H. Rakhshani, E. Dehghanian, A. Rahati, Hierarchy cuckoo search algorithm for parameter estimation in biological systems, Chemometrics and Intelligent Laboratory Systems, 159 (2016) 97-107.
[13] H. Rakhshani, L. Idoumghar, J. Lepagnot, M. Brévilliers, A. Rahati, Accelerating Protein Structure Prediction Using Active Learning and Surrogate-Based Optimization, in: 2018 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2018, pp. 1-6.
[14] E. Dehghanian, M. Kaykhaii, M. Mehrpur, Comparison of single best artificial neural network and neural network ensemble in modeling of palladium microextraction, Monatshefte für Chemie-Chemical Monthly, 146 (2015) 1217-1227.

[15] H.-H. Tsai, B.-M. Chang, X.-P. Lin, Using decision tree, particle swarm optimization, and support vector regression to design a median-type filter with a 2-level impulse detector for image enhancement, Information Sciences, 195 (2012) 103-123.
[16] E. Dehghanian, S.Z. Gheshlaghi, A multiobjective approach in constructing a predictive model for Fischer-Tropsch synthesis, Journal of Chemometrics, 32 (2018) e2969.
[17] R. Masadeh, B.A. Mahafzah, A. Sharieh, Sea Lion Optimization Algorithm, Sea, 10 (2019).
[18] M.A. Asmaran, A.A. Sharieh, B.A. Mahafzah, Chemical Reaction Optimization Algorithm to Find Maximum Independent Set in a Graph, Editorial Preface From the Desk of Managing Editor…, 10 (2019).
[19] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, simulation, 76 (2001) 60-68.
[20] D. Manjarres, I. Landa-Torres, S. Gil-Lopez, J. Del Ser, M.N. Bilbao, S. Salcedo-Sanz, Z.W. Geem, A survey on applications of the harmony search algorithm, Engineering Applications of Artificial Intelligence, 26 (2013) 1818-1831.
[21] H.-b. Ouyang, L.-q. Gao, S. Li, X.-y. Kong, Improved novel global harmony search with a new relaxation method for reliability optimization problems, Information Sciences, 305 (2015) 14-55.
[22] M. El-Abd, An improved global-best harmony search algorithm, Applied Mathematics and Computation, 222 (2013) 94-106.
[23] M. Karimi, A. Askarzadeh, A. Rezazadeh, Using tournament selection approach to improve harmony search algorithm for modeling of proton exchange membrane fuel cell, Int. J. Electrochem. Sci, 7 (2012) 6426-6435.
[24] M.A. Al-Betar, I.A. Doush, A.T. Khader, M.A. Awadallah, Novel selection schemes for harmony search, Applied Mathematics and Computation, 218 (2012) 6095-6117.
[25] M.A. Al-Betar, M.A. Awadallah, A.T. Khader, A.L.a. Bolaji, Tournament-based harmony search algorithm for non-convex economic load dispatch problem, Applied Soft Computing, 47 (2016) 449-459.
[26] X. Zhao, Z. Liu, J. Hao, R. Li, X. Zuo, Semi-self-adaptive harmony search algorithm, Natural Computing, 16 (2017) 619-636.
[27] M.G. Omran, M. Mahdavi, Global-best harmony search, Applied mathematics and computation, 198 (2008) 643-656.
[28] Q.-K. Pan, P.N. Suganthan, M.F. Tasgetiren, J.J. Liang, A self-adaptive global best harmony search algorithm for continuous optimization problems, Applied Mathematics and Computation, 216 (2010) 830-848.
[29] B. Keshtegar, M.O. Sadeq, Gaussian global-best harmony search algorithm for optimization problems, Soft Computing, 21 (2017) 7337-7349.
[30] R. Enayatifar, M. Yousefi, A.H. Abdullah, A.N. Darus, LAHS: a novel harmony search algorithm based on learning automata, Communications in Nonlinear Science and Numerical Simulation, 18 (2013) 3481-3497.
[31] Z.W. Geem, Particle-swarm harmony search for water network design, Engineering Optimization, 41 (2009) 297-311.
[32] E.A. Portilla-Flores, Á. Sánchez-Márquez, L. Flores-Pulido, E. Vega-Alvarado, M.B.C. Yáñez, J.A. Aponte-Rodríguez, P.A. Niño-Suárez, Enhancing the harmony search algorithm performance on constrained numerical optimization, IEEE Access, 5 (2017) 25759-25780.
[33] M. Castelli, S. Silva, L. Manzoni, L. Vanneschi, Geometric selective harmony search, Information Sciences, 279 (2014) 468-482.
[34] M. Shabani, S.A. Mirroshandel, H. Asheri, Selective refining harmony search: A new optimization algorithm, Expert Systems with Applications, 81 (2017) 423-443.
[35] D. Zou, L. Gao, J. Wu, S. Li, Novel global harmony search algorithm for unconstrained problems, Neurocomputing, 73 (2010) 3308-3318.
[36] Z. Guo, S. Wang, X. Yue, H. Yang, Global harmony search with generalized opposition-based learning, Soft Computing, 21 (2017) 2129-2137.

[37] H. Ouyang, W. Wu, C. Zhang, S. Li, D. Zou, G. Liu, Improved harmony search with general iteration models for engineering design optimization problems, Soft Computing, 23 (2019) 10225-10260.
[38] N. Awad, M. Ali, J. Liang, B. Qu, P. Suganthan, Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization, Tech. Rep., (2016).
[39] V. Kumar, J.K. Chhabra, D. Kumar, Parameter adaptive harmony search algorithm for unimodal and multimodal optimization problems, Journal of Computational Science, 5 (2014) 144-155.
[40] M. Khalili, R. Kharrat, K. Salahshoor, M.H. Sefat, Global dynamic harmony search algorithm: GDHS, Applied Mathematics and Computation, 228 (2014) 195-219.
[41] J. Kalivarapu, S. Jain, S. Bag, An improved harmony search algorithm with dynamically varying bandwidth, Engineering Optimization, 48 (2016) 1091-1108.
[42] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Applied mathematics and computation, 188 (2007) 1567-1579.
[43] B. Alatas, Chaotic harmony search algorithms, Applied Mathematics and Computation, 216 (2010) 2687-2699.
[44] K. Luo, J. Ma, Q. Zhao, Enhanced self-adaptive global-best harmony search without any extra statistic and external archive, Information Sciences, 482 (2019) 228-247.
[45] B.H.F. Hasan, I.A. Doush, E. Al Maghayreh, F. Alkhateeb, M. Hamdan, Hybridizing harmony search algorithm with different mutation operators for continuous problems, Applied Mathematics and Computation, 232 (2014) 1166-1182.
[46] A. Kattan, R. Abdullah, A dynamic self-adaptive harmony search algorithm for continuous optimization problems, Applied Mathematics and Computation, 219 (2013) 8542-8567.
[47] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, B.K. Panigrahi, Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 41 (2010) 89-106.
[48] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Ieee, 1995, pp. 39-43.
[49] P. Yadav, R. Kumar, S.K. Panda, C. Chang, An intelligent tuned harmony search algorithm for optimisation, Information Sciences, 196 (2012) 47-72.
[50] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization, 11 (1997) 341-359.
[51] S. Das, P.N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE transactions on evolutionary computation, 15 (2011) 4-31.
[52] Z. Guo, H. Yang, S. Wang, C. Zhou, X. Liu, Adaptive harmony search with best-based search strategy, Soft Computing, 22 (2018) 1335-1349.
[53] H. Wang, Z. Wu, Y. Liu, J. Wang, D. Jiang, L. Chen, Space transformation search: a new evolutionary technique, in: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, ACM, 2009, pp. 537-544.
[54] Y. Wang, Z. Guo, Y. Wang, Enhanced harmony search with dual strategies and adaptive parameters, Soft Computing, 21 (2017) 4431-4445.
[55] H.-b. Ouyang, L.-q. Gao, S. Li, X.-y. Kong, Q. Wang, D.-x. Zou, Improved harmony search algorithm: LHS, Applied Soft Computing, 53 (2017) 133-167.
[56] H. Ouyang, W. Wu, C. Zhang, S. Li, D. Zou, G. Liu, Improved harmony search with general iteration models for engineering design optimization problems, Soft Computing, (2018) 1-36.
[57] R.V. Rao, V.J. Savsani, D. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Computer-Aided Design, 43 (2011) 303-315.
[58] M.A. Al-Betar, A.T. Khader, M.A. Awadallah, M.H. Alawan, B. Zaqaibeh, Cellular harmony search for optimization problems, Journal of Applied Mathematics, 2013 (2013).
[59] E. Alba, B. Dorronsoro, The state of the art in cellular evolutionary algorithms, in: Cellular Genetic Algorithms, Springer, 2008, pp. 21-34.

[60] M.A. Al-Betar, M.A. Awadallah, A.T. Khader, Z.A. Abdalkareem, Island-based harmony search for optimization problems, Expert Systems with Applications, 42 (2015) 2026-2035.
[61] J. Yi, L. Gao, X. Li, J. Gao, An efficient modified harmony search algorithm with intersect mutation operator and cellular local search for continuous function optimization problems, Applied Intelligence, 44 (2016) 725-753.
[62] R. Sarkhel, N. Das, A.K. Saha, M. Nasipuri, An improved Harmony Search Algorithm embedded with a novel piecewise opposition based learning algorithm, Engineering Applications of Artificial Intelligence, 67 (2018) 317-330.
[63] G. Wang, L. Guo, A novel hybrid bat algorithm with harmony search for global numerical optimization, Journal of Applied Mathematics, 2013 (2013).
[64] A. Layeb, A hybrid quantum inspired harmony search algorithm for 0–1 optimization problems, Journal of Computational and Applied Mathematics, 253 (2013) 14-25.
[65] G.-G. Wang, A.H. Gandomi, X. Zhao, H.C.E. Chu, Hybridizing harmony search algorithm with cuckoo search for global numerical optimization, Soft Computing, 20 (2016) 273-285.
[66] H.-b. Ouyang, L.-q. Gao, X.-y. Kong, S. Li, D.-x. Zou, Hybrid harmony search particle swarm optimization with global dimension selection, Information Sciences, 346 (2016) 318-337.
[67] L.M. Abualigah, A.T. Khader, E.S. Hanandeh, A hybrid strategy for krill herd algorithm with harmony search algorithm to improve the data clustering, Intelligent Decision Technologies, (2018) 1-12.
[68] A. Assad, K. Deep, A Hybrid Harmony search and Simulated Annealing algorithm for continuous optimization, Information Sciences, 450 (2018) 246-266.
[69] L. Wang, H. Hu, R. Liu, X. Zhou, An improved differential harmony search algorithm for function optimization problems, Soft Computing, (2018) 1-26.
[70] S. Tuo, L. Yong, T. Zhou, An improved harmony search based on teaching-learning strategy for unconstrained optimization problems, Mathematical Problems in Engineering, 2013 (2013).
[71] R.C. Eberhart, Y. Shi, J. Kennedy, Swarm intelligence, Elsevier, 2001.
[72] J. Cheng, L. Wang, Y. Xiong, An improved cuckoo search algorithm and its application in vibration fault diagnosis for a hydroelectric generating unit, Engineering Optimization, 50 (2018) 1593-1608.
[73] W.-l. Xiang, M.-q. An, Y.-z. Li, R.-c. He, J.-f. Zhang, An improved global-best harmony search algorithm for faster optimization, Expert Systems with Applications, 41 (2014) 5788-5803.
[74] C.-M. Wang, Y.-F. Huang, Self-adaptive harmony search algorithm for optimization, Expert Systems with Applications, 37 (2010) 2826-2837.
[75] S. Aine, R. Kumar, P. Chakrabarti, Adaptive parameter control of evolutionary algorithms to improve quality-time trade-off, Applied Soft Computing, 9 (2009) 527-540.
[76] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization, Journal of Heuristics, 15 (2009) 617.
[77] A. Lin, W. Sun, H. Yu, G. Wu, H. Tang, Adaptive comprehensive learning particle swarm optimization with cooperative archive, Applied Soft Computing, 77 (2019) 533-546.
[78] C.-W. Tsai, K.-W. Huang, C.-S. Yang, M.-C. Chiang, A fast particle swarm optimization for clustering, Soft computing, 19 (2015) 321-338.
[79] C.J. Carmona, S. Ramírez-Gallego, F. Torres, E. Bernal, M.J. del Jesús, S. García, Web usage mining to improve the design of an e-commerce website: OrOliveSur. com, Expert Systems with Applications, 39 (2012) 11243-11249.
[80] J. Han, J. Pei, M. Kamber, Data mining: concepts and techniques, Elsevier, 2011.
[81] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM computing surveys (CSUR), 31 (1999) 264-323.
[82] B. King, Step-wise clustering procedures, Journal of the American Statistical Association, 62 (1967) 86-101.

[83] L. Kaufman, P.J. Rousseeuw, Finding groups in data: an introduction to cluster analysis, John Wiley & Sons, 2009.
[84] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Oakland, CA, USA, 1967, pp. 281-297.
[85] F.T. Evers, F. Höppner, F. Klawonn, R. Kruse, T. Runkler, Fuzzy cluster analysis: methods for classification, data analysis and image recognition, John Wiley & Sons, 1999.
[86] A.R. Anaya, J.G. Boticario, Application of machine learning techniques to analyse student interactions and improve the collaboration process, Expert Systems with Applications, 38 (2011) 1171-1181.
[87] N. Bassiou, C. Kotropoulos, Long distance bigram models applied to word clustering, Pattern Recognition, 44 (2011) 145-158.
[88] N.M. Portela, G.D. Cavalcanti, T.I. Ren, Semi-supervised clustering for MR brain image segmentation, Expert Systems with Applications, 41 (2014) 1492-1497.
[89] B. Zheng, S.W. Yoon, S.S. Lam, Breast cancer diagnosis based on feature extraction using a hybrid of K-means and support vector machine algorithms, Expert Systems with Applications, 41 (2014) 1476-1482.
[90] J.C. Bezdek, Pattern recognition with fuzzy objective function algorithms, Springer Science & Business Media, 2013.
[91] R.J. Hathaway, J.C. Bezdek, Local convergence of the fuzzy c-means algorithms, Pattern recognition, 19 (1986) 477-480.
[92] S.C. Cohen, L.N. de Castro, Data clustering with particle swarms, in: 2006 IEEE International Conference on Evolutionary Computation, IEEE, 2006, pp. 1792-1798.
[93] R. Forsati, M. Mahdavi, M. Kangavari, B. Safarkhani, Web page clustering using harmony search optimization, in: 2008 Canadian Conference on Electrical and Computer Engineering, IEEE, 2008, pp. 001601-001604.
[94] D. Karaboga, C. Ozturk, A novel clustering approach: Artificial Bee Colony (ABC) algorithm, Applied soft computing, 11 (2011) 652-657.
[95] S. Goel, A. Sharma, P. Bedi, Cuckoo Search Clustering Algorithm: A novel strategy of biomimicry, in: 2011 World Congress on Information and Communication Technologies, IEEE, 2011, pp. 916-921.
[96] P. Shelokar, V.K. Jayaraman, B.D. Kulkarni, An ant colony approach for clustering, Analytica Chimica Acta, 509 (2004) 187-195.
[97] S.J. Nanda, G. Panda, A survey on nature inspired metaheuristic algorithms for partitional clustering, Swarm and Evolutionary computation, 16 (2014) 1-18.
[98] S.Z. Selim, M.A. Ismail, K-means-type algorithms: A generalized convergence theorem and characterization of local optimality, IEEE Transactions on pattern analysis and machine intelligence, (1984) 81-87.
[99] D. Arthur, S. Vassilvitskii, k-means++: The advantages of careful seeding, in: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2007, pp. 1027-1035.
[100] S. Rana, S. Jasola, R. Kumar, A review on particle swarm optimization algorithms and their applications to data clustering, Artificial Intelligence Review, 35 (2011) 211-222.
[101] U. Maulik, S. Bandyopadhyay, Genetic algorithm-based clustering technique, Pattern recognition, 33 (2000) 1455-1465.
[102] S.Z. Selim, K. Alsultan, A simulated annealing algorithm for the clustering problem, Pattern recognition, 24 (1991) 1003-1008.
[103] Y. Liu, Z. Yi, H. Wu, M. Ye, K. Chen, A tabu search approach for the minimum sum-of-squares clustering problem, Information Sciences, 178 (2008) 2680-2704.
[104] M. Fathian, B. Amiri, A honeybee-mating approach for cluster analysis, The International Journal of Advanced Manufacturing Technology, 38 (2008) 809-821.

[105] A.H. Gandomi, A.H. Alavi, Krill herd: a new bio-inspired optimization algorithm, Communications in nonlinear science and numerical simulation, 17 (2012) 4831-4845.
[106] R. Jensi, G.W. Jiji, An improved krill herd algorithm with global exploration capability for solving numerical function optimization problems and its application to data clustering, Applied Soft Computing, 46 (2016) 230-245.
[107] C.L. Blake, C. Merz, UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences, in, 1998.
[108] C.d. M. Learning, 2016. http://cs.joensuu.fi/sipu/datasets/.

Highlights

- A novel harmony search is proposed with three pitch selection and production rules.
- The rules utilize the gathered knowledge during search in generation of harmonies.
- A probabilistic self-adaptive selection scheme is used to balance search behaviors.
- The algorithm is applied on numerical optimization and data clustering problems.
- Results show the superiority of proposed algorithm to the compared algorithms.

Kazem Talaei: Implementation, Methodology, Investigation, Writing-Original draft preparation **Amin Rahati:** Supervision, Reviewing and Editing, Validation. **Lhassane Idoumghar:** Supervision

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: