



Article

A New Method for Analyzing the Performance of the Harmony Search Algorithm

Shouheng Tuo ^{1,2,*} , Zong Woo Geem ^{3,*}  and Jin Hee Yoon ⁴¹ School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, Xi'an 710121, China² Shaanxi Key Laboratory of Network Data Intelligent Processing, Xi'an University of Posts and Telecommunications, Xi'an 710121, China³ Department of Energy IT, Gachon University, Seongnam 13120, Korea⁴ School of Mathematics and Statistics, Sejong University, Seoul 05006, Korea; jin9135@sejong.ac.kr

* Correspondence: tuo_sh@xupt.edu.cn (S.T.); geem@gachon.ac.kr (Z.W.G.)

Received: 29 July 2020; Accepted: 21 August 2020; Published: 24 August 2020



Abstract: A harmony search (HS) algorithm for solving high-dimensional multimodal optimization problems (named DIHS) was proposed in 2015 and showed good performance, in which a dynamic-dimensionality-reduction strategy is employed to maintain a high update success rate of harmony memory (HM). However, an extreme assumption was adopted in the DIHS that is not reasonable, and its analysis for the update success rate is not sufficiently accurate. In this study, we reanalyzed the update success rate of HS and now present a more valid method for analyzing the update success rate of HS. In the new analysis, take-k and take-all strategies that are employed to generate new solutions are compared to the update success rate, and the average convergence rate of algorithms is also analyzed. The experimental results demonstrate that the HS based on the take-k strategy is efficient and effective at solving some complex high-dimensional optimization problems.

Keywords: harmony search; update success rate; high-dimensional optimization problem; take-k strategy; take-all strategy

1. Introduction

Harmony search (HS) [1–5], which is a meta-heuristic search algorithm that mimics the process of improvising a musical harmony, has been extensively employed in the fields of complex scientific computing and engineering optimization. Deng et al. developed an improved HS algorithm to construct examples for an algebra system in 2015 [6]. Tuo et al. proposed a hybrid algorithm based on HS and teaching-learning-based optimization (TLBO) for solving complex high-dimensional optimization problems; the algorithm showed efficient performance [7]. In the fields of engineering optimization, an enhanced HS was proposed to solve dynamic economic emission dispatch problems; the experimental results indicated better solutions and less computational time [8]. An adaptive dynamic HS is presented for optimizing aircraft panels [9]. Xu and Wang developed a harmony search algorithm for minimizing serve energy by examining information caching and upgrading techniques of the Internet of Things (IoT) [10]. To solve complex engineering design optimization, Yi et al. proposed an online variable-fidelity surrogate-assisted HS with a multi-level screening strategy [11]. In addition, HS has also been applied in manufacturing systems [12], microgrid planning with ESS [13], flow shop scheduling problems [14] etc. [15–17].

In recent years, various variants of HS were proposed to improve the performance for solving complex optimization problems [17–24], such as accelerating search speed, enhancing the power of global exploration and balancing the tradeoff between diversification and intensification. Zhang et al.

reviewed the HS and its variants with respect to the algorithm structure [23]. D. Manjarres et al. analyzed the main characteristics and application portfolio of HS, and presented future research lines of HS in 2013 [24]. Nasir et al. compared the study of HS in China, Japan and Korea [25]. Although, the HS and its variants have shown strong power for solving some complex optimization problems, they have low-efficiency and low-precision solutions for solving high-dimensional (≥ 500) multimodal optimization problems. In 2015, an HS algorithm named “A harmony search algorithm for high-dimensional multimodal optimization problems (DIHS) [4]” was proposed. In DIHS, to determine why the standard HS has low efficiency and low precision in solving high-dimensional optimization problems, the take-one search strategy and take-all search strategy were compared to analyze the success rate of newly generated solutions for updating the worst solution in harmony memory (HM). In the comparative analysis, an extreme case of solutions for the HM is adopted assuming that all solutions need only adjust one-dimensional values to achieve the global optimal solution. The analytical results [4] are as follows:

- (1) The update success rate (defined as the rate that the new generated harmony is better than the worst harmony in HM.) of the take-one strategy is

$$R_{take-one} = \frac{1}{D} \times \frac{HMS - 1}{HMS} \quad (1)$$

where D is the dimension of the optimization problem and HMS denotes the size of the harmony memory.

- (2) The update success rate of the take-all strategy is

$$R_{take-all} = \left(\frac{HMS - 1}{HMS} \right)^D \quad (2)$$

In recent research, we identified a key analysis error in the article of DIHS, when the dimension D is larger than the HMS , the update success rate formulas [4] are incorrect. Although the error does not affect the correctness of the final conclusion of the article, it may cause readers to misunderstand and incorrectly apply the DIHS algorithm. To correct the error, in this work, we introduce a new method for analyzing the take-one search and take-all search strategies.

The contributions of this article are listed as follows:

- (1) New analysis strategies, namely, the take- k and take-all schemes, are presented. The analysis assumptions used are closer to the conditions in the actual optimization process than are the previous extreme assumptions.
- (2) An adaptive take- k strategy is proposed for improving the harmony search algorithm.

The remainder of this paper is organized as follows. HS is introduced in Section 2. Section 3 presents detailed analyses of the take-one strategy and take-all strategy. Section 4 introduces new take- k and take-all strategies. The simulation experiments are performed and the results are analyzed in Section 5. In Section 6, discussions are presented.

2. Standard Harmony Search Algorithm

HS is a very simple real-code optimization algorithm that can be easily used to solve continuous and discrete optimization problems. This algorithm contains three operators:

- (1) The combinatorial operator is designed to select values from HM and obtain a new harmony. The objective is to improve the new harmony based on the historical memory of musicians.
- (2) The local-adjusting operator fine-tunes the potential solutions near the new harmony with the aim of finding the most precise solution. This approach can aggravate spatial disturbances associated with a large fret width and fine-tune harmonies with a small fret width.

(3) The random search operator is designed to explore unknown search areas and prevent the algorithm from falling to local optima.

The pseudocode of the standard HS algorithm is expressed as algorithm 1:

Algorithm 1. The pseudocode of standard HS algorithm.

Input:

MaxT: maximum number of iterations

HMCR: harmony memory consideration rate

HMS: harmony memory size PAR: pitch-adjusting rate

fw: fret width

Output: Best harmony χ^{best}

1. Randomly initialize HM

$$\text{HM} = \begin{bmatrix} X^1 \\ X^2 \\ \vdots \\ X^{\text{HMS}} \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_D^1 \\ x_1^2 & x_2^2 & \cdots & x_D^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{\text{HMS}} & x_2^{\text{HMS}} & \cdots & x_D^{\text{HMS}} \end{bmatrix}$$

$$x_i^j = x_i^L + \text{rand}(0, 1) \times (x_i^U - x_i^L), i = 1, 2, \dots, D; j = 1, 2, \dots, \text{HMS}.$$

x_i^L and x_i^U denote the lower bound and upper bound of the i th decision variable, respectively.

2. Improvise a new harmony $X^{\text{new}} = (x_1^{\text{new}}, x_2^{\text{new}}, \dots, x_D^{\text{new}})$ as follow.
-

For $I = 1 \rightarrow D$

If $\text{rand}(0, 1) < \text{HMCR}$

$$x_i^{\text{new}} = r \in \{x_i^1, x_i^2, \dots, x_i^{\text{HMS}}\}$$

If $\text{rand}(0, 1) < \text{PAR}$

$$x_i^{\text{new}} = x_i^{\text{new}} + \text{rand}(0, 1) \times fw_i$$

End

Else

$$x_i^{\text{new}} = x_i^L + \text{rand}(0, 1) \times (x_i^U - x_i^L)$$

End

End

3. Update HM with the new harmony as follows:

$$X^{\text{IDworst}} = \begin{cases} X^{\text{new}}, & X^{\text{new}} \text{ is better than } X^{\text{IDworst}} \\ X^{\text{IDworst}}, & \text{otherwise} \end{cases}$$

where IDworst is the index of the worst harmony in HM.

4. If the terminal condition is met, then go to step 2.
 5. Output the best harmony of HM.
-

3. Take-One Strategy and Take-All Strategy

In the original paper about DIHS, an extreme case of solutions in the HM is adopted assuming that all solutions need only adjust one-dimensional values to achieve the global optimal solution. HM can be converted to the following matrix by swapping rows and columns.

$$\text{HM} = \begin{bmatrix} x_1^1 & x_2^* & \cdots & x_{\text{HMS}}^* & x_{\text{HMS}+1}^* & \cdots & x_D^* \\ x_1^* & x_2^* & \cdots & x_{\text{HMS}}^* & x_{\text{HMS}+1}^* & \cdots & x_D^* \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_1^* & x_2^* & \cdots & x_{\text{HMS}}^* & x_{\text{HMS}+1}^* & \cdots & x_D^* \end{bmatrix} \quad (3)$$

In the HM, x_i^* ($i = 1, 2, \dots, D$) indicates that the i th decision variable has an optimal value. $x_1^1, x_2^2, \dots, x_{HMS}^{HMS}$ are decision variables for which the optimal value has not been obtained.

Under this assumption,

(1) The success rate of update of the take-one strategy should be $\frac{HMS}{D} * \frac{HMS-1}{HMS}$, not $\frac{1}{D} * \frac{HMS-1}{HMS}$, where $\frac{HMS}{D}$ indicates the probability of selecting one of the previous HMS columns from HM to be adjusted. $\frac{HMS-1}{HMS}$ represents the probability of obtaining the best value x_i^* ($i = 1, 2, \dots, D$) of the i th decision variable.

(2) The success rate of update of the take-all strategy should be $\left(\frac{HMS-1}{HMS}\right)^{HMS}$, not $\left(\frac{HMS-1}{HMS}\right)^D$, because all of the values of decision variables $x_{HMS+1}, x_{HMS+2}, \dots, x_D$ are optimal.

Under this condition, the update success rate of the take-all strategy is independent of the dimension of the problem, which is inconsistent with the results of the previous analysis. If we assume that there is one and only one nonoptimal value in all columns of the matrix HM, formulas (1) and (2) are correct.

However, based on many experimental results, the concept of the DIHS algorithm should be correct, but the premise assumptions were too extreme for this analysis. Therefore, in this work, we employ more realistic hypothetical conditions to reanalyze the dynamic-dimensionality-reduction adjustment (DDRA) strategy using take-k strategy and take-all strategy.

4. Take-K Strategy and Take-All Strategy

4.1. Concepts and Assumptions

The take-k strategy means that k decision variables of X^{new} ($= X^{IDworst}$) are chosen for adjusting. If the adjusted X^{new} is better than the old $X^{IDworst}$, then the $X^{IDworst}$ in HM is replaced by the adjusted X^{new} . If the value of k is equal to dimension D , the take-k strategy is referred to as the take-all strategy.

The take-k strategy employs the following step to improvise the harmony (see Algorithm 2).

Algorithm 2. The Pseudocode for improvising new harmony for the take-k s

$X^{new} = X^{IDworst}$

For $i = 1 \rightarrow D$

If $\text{rand}(0,1) < k/D$ // each dimension has a k/D chance of being adjusted.

If $\text{rand}(0,1) < \text{HMCR}$

$x_i^{new} = x_i^j$ //combinatorial operator

If $\text{rand}(0,1) < \text{PAR}$

$x_i^{new} = x_i^{new} + \text{rand}(0,1) \times fw(i)$ //local-adjusting operator

End

Else

$x_i^{new} = x_i^L + \text{rand}(0,1) \times (x_i^U - x_i^L)$ // random search operator

End

End

End

Assumption 1. Suppose that in the i th column of HM, there is only one value that is not equal to x_i^* ($i = 1, 2, \dots, D$). In the j^{th} row of HM, there are D/HMS values that do not reach the optimal value, as follows:

$$HM = \begin{bmatrix} y_1 & x_2^* & \cdots & x_{HMS}^* & y_{HMS+1} & x_{HMS+2}^* & \cdots & x_D^* \\ x_1^* & y_2 & \cdots & x_{HMS}^* & x_{HMS+1}^* & y_{HMS+1} & \cdots & x_D^* \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^* & x_2^* & \cdots & y_{HMS} & x_{HMS+1}^* & x_{HMS+2}^* & \cdots & y_D \end{bmatrix} \quad (4)$$

Assumption 2. Suppose that in generation t , the probability that each dimension in HM is able to obtain a better value by employing three optimal operators (combinatorial operator, local-adjusting operator, and random search operator) than it currently holds is equal to $p(t) = \frac{HMS-1}{HMS}$.

In the take- k strategy, first $X^{new} = X^{IDworst}$; then, each dimension x_i^{new} ($i = 1, 2, \dots, D$) has a k/D chance of being updated by the three operators. If $k=D$, the take- k and take-all strategies are equivalent.

If x_i^{new} is chosen from HM for adjustment, the probability of improving x_i^{new} is $p(t)$, and the probability of the value worsening is $1 - p(t)$.

Assumption 3. Suppose that there are k decision variables in X^{new} that are selected to be adjusted. If more than $1/2$ of the decision variables achieved improved values, a better solution was obtained.

Definition 1. Gain value. $X^{new}(t)$ is a new improvised solution in the t -th iteration. The gain value $G(t)$ obtained at the t -th iteration is defined as

$$G(t) = \begin{cases} f(X^{IDworst}) - f(X^{new}) & , \quad f(X^{new}) < f(X^{IDworst}) \\ 0 & , \quad \text{otherwise} \end{cases} \quad (5)$$

The gain value $G(\Delta t)$ after Δt iterations is defined as

$$G(\Delta t) = f(X^{IDbest}(t_2)) - f(X^{IDbest}(t_1)) \quad (6)$$

where $\Delta t = t_2 - t_1$, $X^{IDbest}(t_2)$ and $X^{IDbest}(t_1)$ denote the best solutions at times t_2 and t_1 ($t_2 > t_1$), respectively.

4.2. Update Success Rate of Take- k and Take-All Strategies

To analyze the update success rate of the take- k and take-all strategies, Figure 1 introduces the judging criteria for generating a better solution, X^{new} . In the take- k strategy, first, the values of X^{new} are initialized to the values of the worst harmony $X^{IDworst}$ in HM. Second, X^{new} is adjusted according to algorithm 2. If more than $1/2$ of the decision variables of X^{new} have been improved, we think X^{new} is improved successfully. Otherwise, X^{new} is considered to be an improvisation.

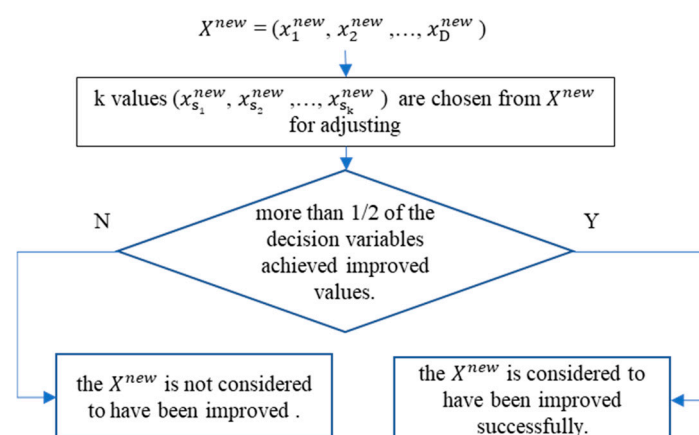


Figure 1. Flow chart that shows adjustments to X^{new} .

By assumption 3, the probability that a better solution can be obtained using the take- k strategy is

$$R_{take-k} = \sum_{i=k/2+1}^k C_k^i \times p(t)^i (1-p(t))^{k-i} \quad (7)$$

where $i = k/2 + 1$ to k , which means that among the k decision variables, more than $k/2$ decision variables obtain better values.

The take-all strategy is equivalent to the take- D strategy ($k = D$), and the probability of obtaining a better solution is

$$R_{take-all} = \sum_{i=D/2+1}^D C_D^i \times p(t)^i (1-p(t))^{D-i} \quad (8)$$

From Equation (7), the update success rate R_{take-k} is independent of the dimension D . Figure 2 shows the success rate curves of the take- k ($k = 0.1D$) and take-all strategies as the dimension D changes.

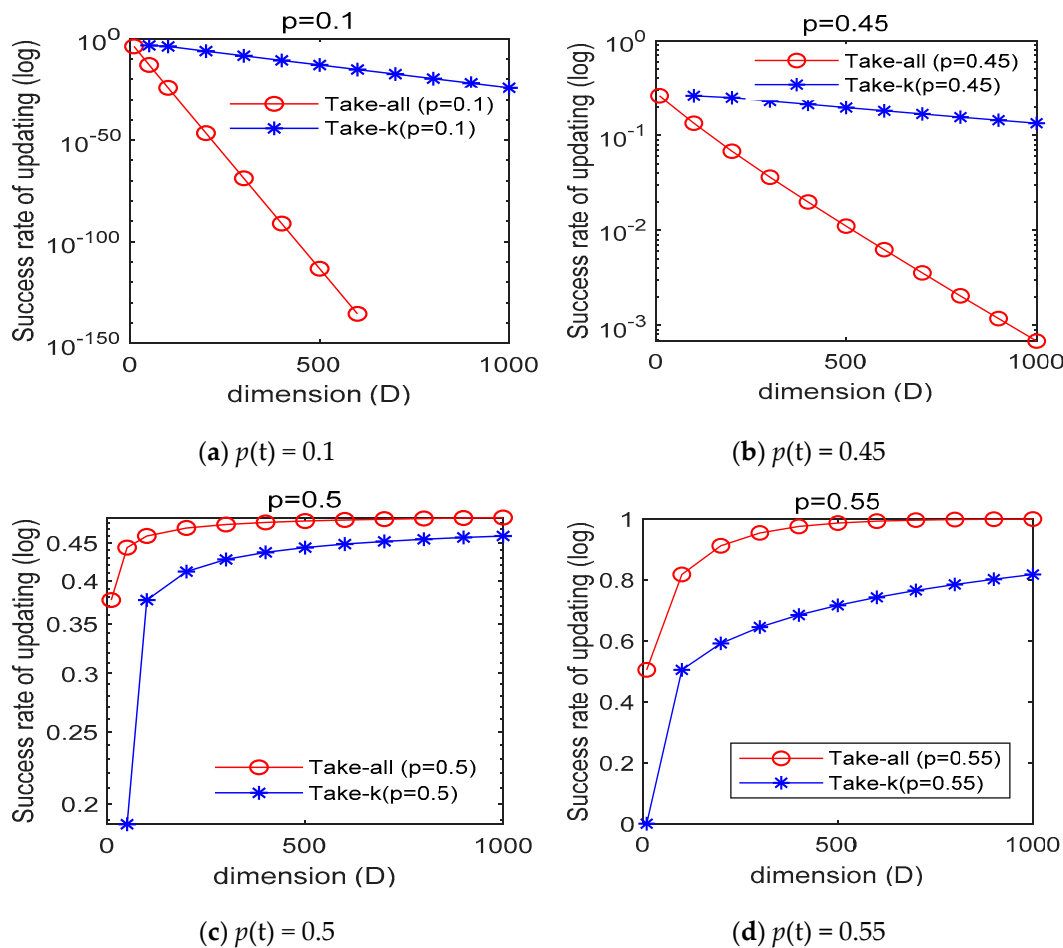


Figure 2. Comparison of the take- k and take-all strategies ($k = 0.1D$). (a) $p(t) = 0.1$; (b) $p(t) = 0.45$; (c) $p(t) = 0.5$; (d) $p(t) = 0.55$.

As shown in Figure 2, when $p(t) < 0.5$, the update success rate for both strategies decreases with dimension D , but the take-all curve decreases much faster than the take- k curve. When $p > 0.5$, the update success rate of the take-all strategy is higher than that of the take- k strategy for a high-dimensional optimization problem; moreover, if $D > 500$, the success rate of the take-all approach is close to 1. However, in practice, the value of $p(t)$ is usually less than 0.5.

Figure 3 shows a comparison of the update success rate of the take- k and take-all strategies when using $p(t) \in \{0.3, 0.25, 0.2, 0.15, 0.1, 0.05\}$. In Figure 3a,b, the x -axis represents the probability that decision variable x_i ($i = 1, 2, \dots, D$) in HM can obtain a better value than the current value; the y -axis shows the update success rate. We can see that as D and k increase, the success rate of updating the HM significantly decreases. Moreover, the smaller the value of $p(t)$ is, the smaller the success rate.

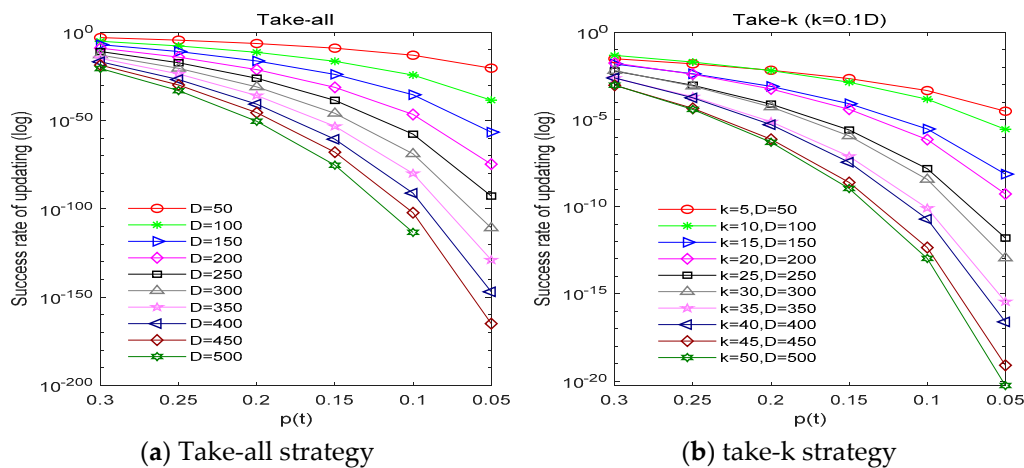


Figure 3. Update success rate.

Generally, for a complex optimization problem, in the search process of an intelligent optimization algorithm, the population has excellent initial diversity, and each decision variable has a high probability of achieving an improved value. As the search progresses, the diversity of the population decreases, and the probability of finding better individuals decreases. However, if the population is clustered around a local optimal solution (the local optimal solution has not yet been obtained), then the probability of finding a better solution is usually high. However, if the population has converged to a local optimum (the local optimal solution has been found), the probability of finding a better solution is 0.

To adapt to the search process, DIHS adopted a DDRA strategy. An adaptive take-k strategy can be employed in the DIHS algorithm as follows:

$$k = K_{\max} - (K_{\max} - K_{\min}) \cdot \left(\frac{t}{\text{MaxFEs}} \right)^a \quad (9)$$

where K_{\max} and K_{\min} represent the maximum and minimum values of k , respectively. a is a parameter used to adjust the rate of descent of k , taking a value of 2 in this paper. MaxFEs denotes the Maximum number of function evaluations (FEs).

The parameters PAR and fw are the same as those in the original DIHS method [4].

$$\text{PAR}(t) = \text{textPAR}_{\text{PARmin}} + (\text{PAR}_{\max} - \text{PAR}_{\min}) \times \frac{t}{\text{MaxFEs}} \quad (10)$$

$$fw(t) = \begin{cases} fw_{\max} \times \left(\frac{fw_{\text{mid}}}{fw_{\max}} \right)^{\left(\frac{t}{\text{MaxFEs}/2} \right)^2}, & t \leq \frac{\text{MaxFEs}}{2} \\ fw_{\text{mid}} \times \left(\frac{fw_{\min}}{fw_{\text{mid}}} \right)^{\left(\frac{t - \text{MaxFEs}/2}{\text{MaxFEs}/2} \right)^2}, & t > \frac{\text{MaxFEs}}{2} \end{cases} \quad (11)$$

where PAR_{\max} and PAR_{\min} represent the maximum and minimum pitch adjustment rates, respectively; fw_{\min} and fw_{\max} denote the minimum and maximum fret widths, respectively; and fw_{mid} is a value between fw_{\min} and fw_{\max} .

The parameters PAR and fw are both used to balance the exploration power and exploitation power of DIHS. Equation (9) is equivalent to Equation (1) of literature [4] in which the parameters a , HMS , PAR and fw have been analyzed in detail.

5. Simulation Experiments

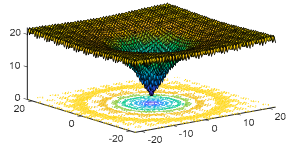
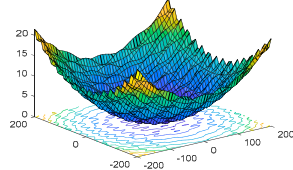
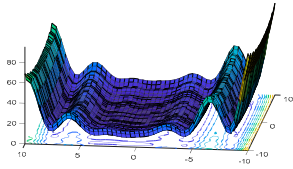
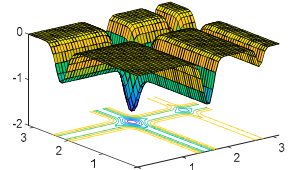
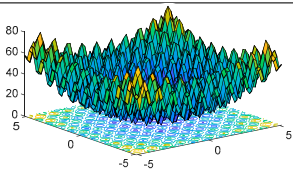
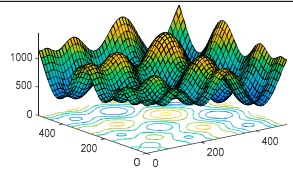
5.1. Experimental Environment and Parameter Settings

To investigate the take-k and take-all strategies, six classical benchmark functions [26] (see Table 1) are employed to test the performance of the two strategies. Take-all-based HS, take-k-based HS and take-k-based DIHS are compared based on the six benchmark functions. The dimensions (D) of all functions are equal to 1000 in the experiments. In the take-k-based HS, the value of k is set to 0.01D. In the take-k-based DIHS, parameters are set as follows:

$$K_{max} = 0.1D, K_{min} = 0.01D; PAR_{max} = 0.75, PAR_{min} = 0.02;$$

$$fw_{max} = (x_i^U - x_i^L)/500; fw_{min} = (x_i^U - x_i^L) \times 10^{-15}; fw_{mid} = (x_i^U - x_i^L) \times 10^{-5}.$$

Table 1. Six classical benchmark functions.

| Function Name and Formula | Function Graph |
|--|---|
| Ackley $f(X) = -20e^{-\frac{1}{\sqrt{D}} \sum_{i=1}^D x_i^2} - e^{\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)} + 20 + e$ |  |
| Griewank $f(X) = \frac{1}{4000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \left(\cos\left(\frac{x_i}{\sqrt{i}}\right) \right) + 1$ |  |
| Levy $f(X) = \sin^2(\pi y_1) + \sum_{i=1}^{D-1} \left[(y_i - 1)^2 (1 + 10 \sin^2(\pi y_i + 1)) \right] + (y_D - 1)^2 (1 + 10 \sin^2(2\pi y_D))$ $y_i = 1 + \frac{x_i - 1}{4}, i = 1, 2, \dots, D$ |  |
| Michalewics $f(X) = -\sum_{i=1}^D \sin(x_i) \left(\sin(ix_i^2/\pi) \right)^{2m}, m = 10$ |  |
| Rastrigin $f(X) = 10D + \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i))$ |  |
| Schwefel 2.26 $f(X) = 418.982887272434D - \sum_{i=1}^D x_i^2 \sin(\sqrt{ x_i })$ |  |

Simulation experiments are performed to compare the update success rate and convergence of the three algorithms. Figure 4 shows the success rate curves and convergence curves. To further

investigate the search ability of the three algorithms, we trace the values of the gains obtained during time Δt and the convergence rate.

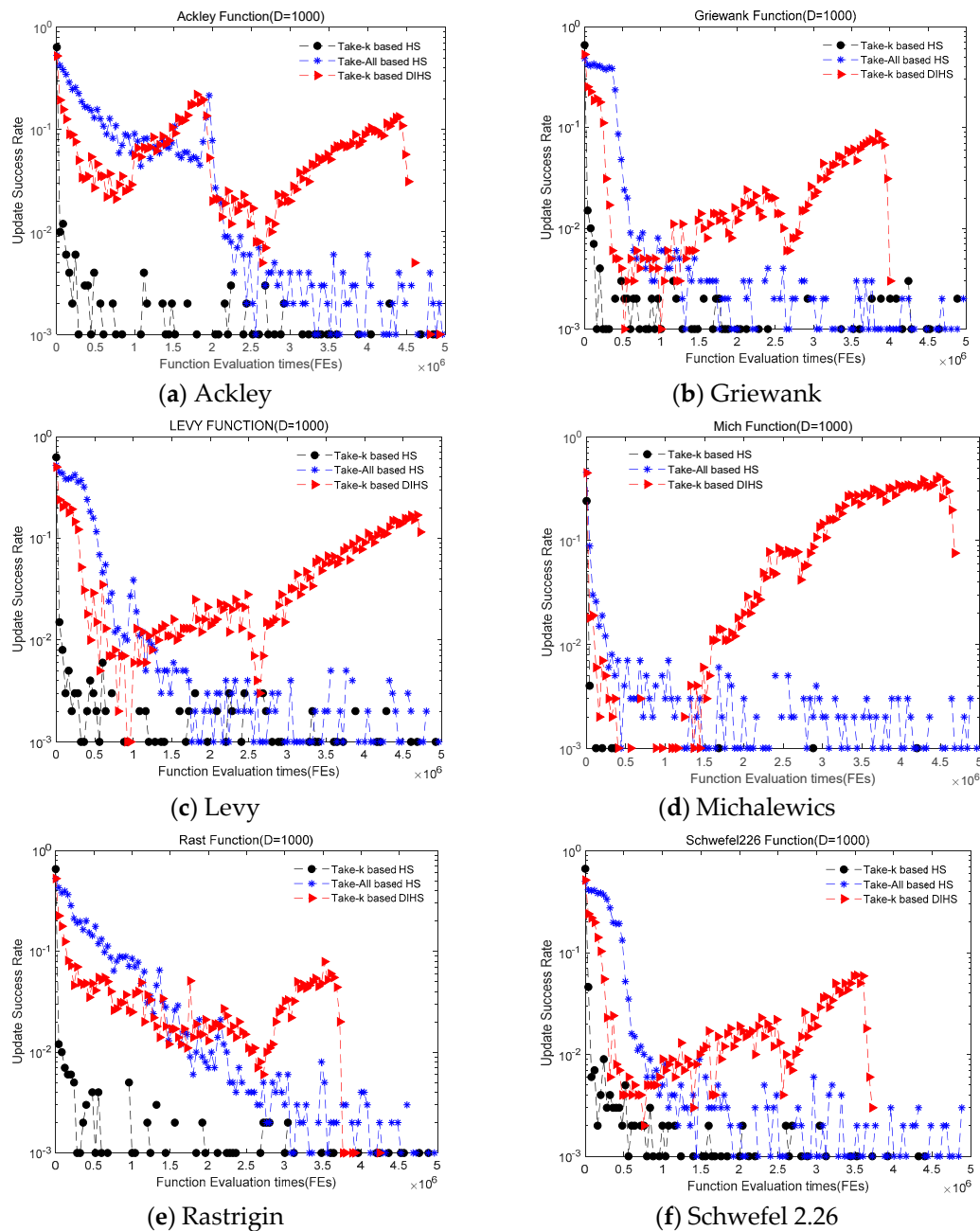


Figure 4. Update success rate comparisons of the take-all and take-k strategies.

5.2. Update Success Rate and Convergence Speed

As shown in Figure 4, at the beginning of the search process, the take-all-based HS has a higher update success rate than the take-k-based HS. As the search progresses, the update success rate of the take-all-based HS decreases dramatically, and the take-k-based HS and DIHS methods are able to maintain a high update success rate during the search. The take-all-based HS approach can be compared to take-k-based DIHS; the former has a higher success rate at the beginning stage of the search. However, in the mid to late stages of the search, the update success rate for the take-k-based DIHS gradually increases and exceeds the success rate for the take-k-based HS.

From the convergence curves of the algorithms in Figure 5, the take-all-based HS algorithm has the slowest convergence rate among the three compared algorithms. The take-k-based HS has a high convergence rate during the initial stage of the search, but it is much slower than take-k-based DIHS overall. Of the three algorithms, DIHS has the strongest ability to obtain high-precision optimal solutions for all test functions.

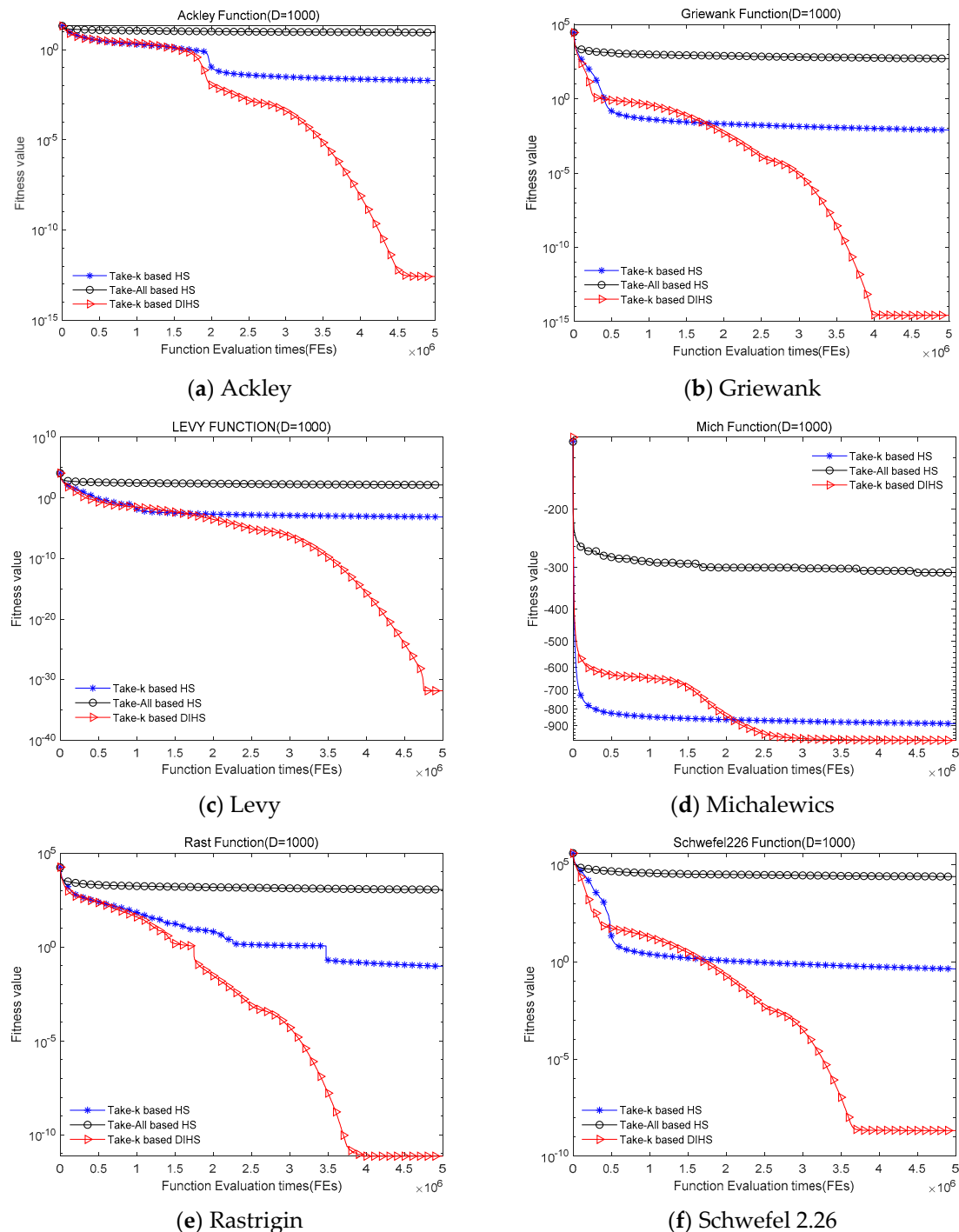


Figure 5. Convergence curves of the take-all and take-k strategies.

Figure 6 presents the gain values that are traced during the search. Although the take-k strategy has a higher probability of finding an improved solution than the take-all strategy during the search process, the improvement over the best solution $G(\Delta t)$ of the take-k strategy may be small.

As shown in Figure 6, the standard HS algorithms that are based on the take-k and take-all strategies have higher gain values than the take-k-based DIHS at certain times, but they are prone to have zero (less than 1×10^{-30}) gain values at points in the search process. The take-k-based DIHS seldom obtains a zero-gain value before the global optimal solution is reached, except for the Michalewics function. The gain values of the take-k-based DIHS become very small late in the search process because the DIHS algorithm has converged to the global optimal solution.

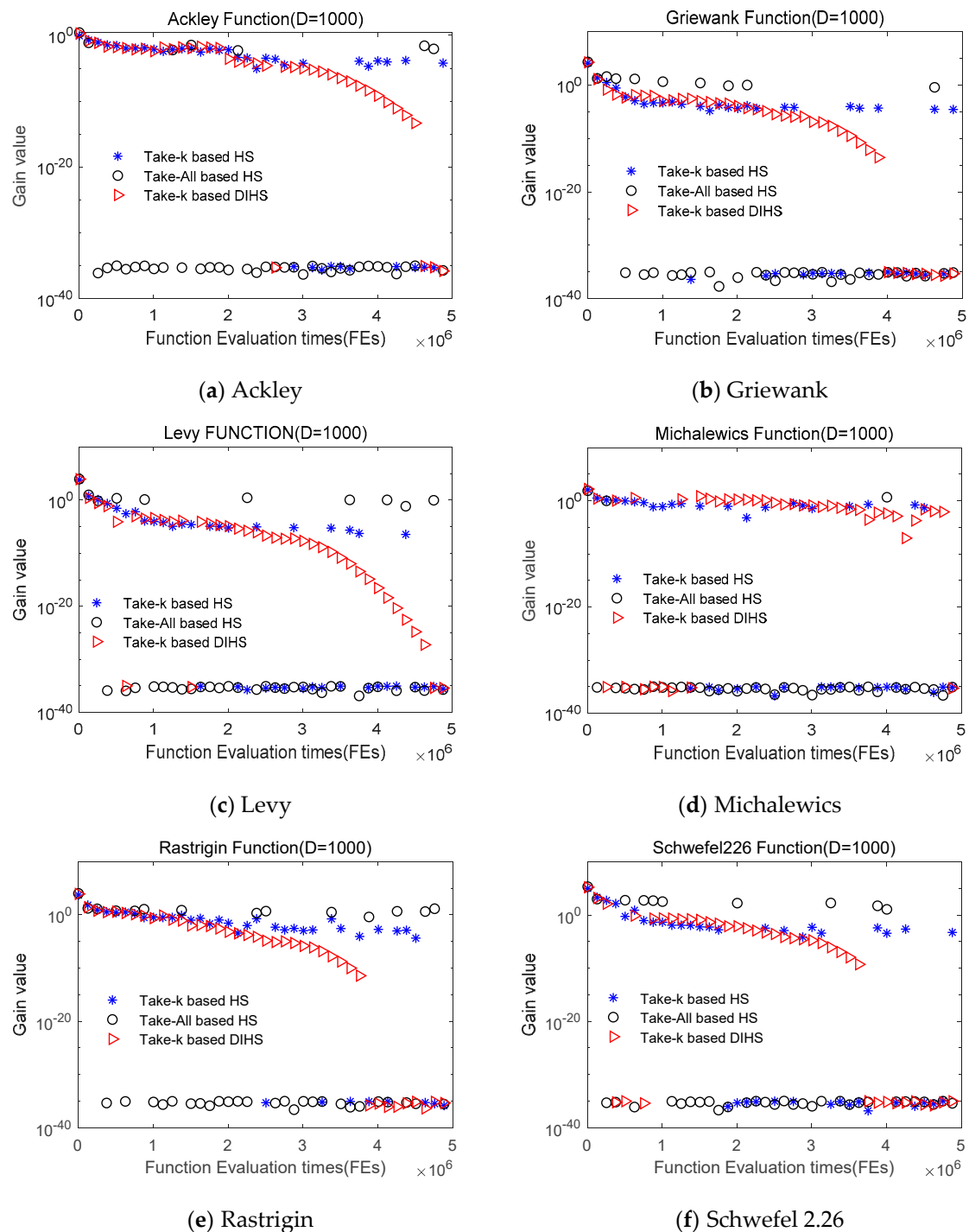


Figure 6. Gain value $G(\Delta t)$ of three algorithms during the search process.

To further investigate the performance of the take-k strategy for HS, six classical high-dimensional optimization problems with multimodal functions (sphere shift function, Schwefel shift function,

Rastrigin shift function, Ackley shift function, Griewank shift function and fast fractal double dip function) [27,28] are tested. Figure 7 presents the update success rate of three algorithms (take-k-based HS, take-all-based HS and take-k-based DIHS); the results confirm the previous conclusions.

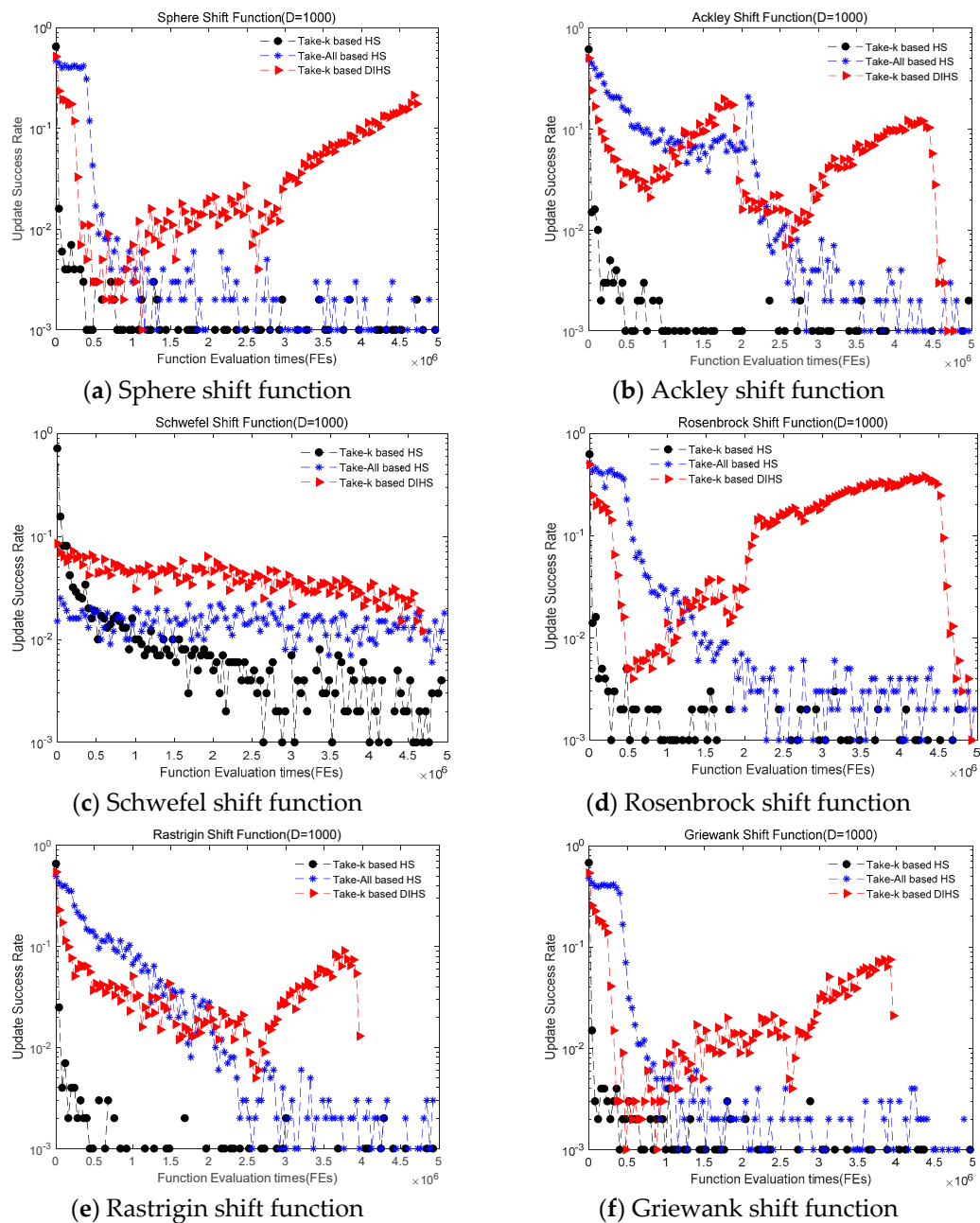


Figure 7. Update success rate of the algorithms for six classical high-dimensional optimization problems.

Table 2 summarized the test results of six algorithms (take-all-based HS, take-k-based HS, SaDE [29], CoDE [30], CLPSO [31] and take-k-based DIHS) for the six high-dimensional optimization problems.

Table 2. Comparisons of six algorithms for six high-dimensional optimization problems (D=1000) (Best/Mean/Worst denotes the best/mean/worst fitness values, respectively, obtained in 30 runs).

| Algorithm | Function | Best | Mean | Worst | Runtime (s) | Function | Best | Mean | Worst | Runtime (s) |
|-------------------|---------------------|------------------------|------------------------|------------------------|--------------------|--------------------|------------------------|------------------------|------------------------|--------------------|
| HS (Take-all) | Sphere shift | 6.13×10^4 | 6.44×10^4 | 6.69×10^4 | 1.02×10^3 | Rastrigin shift | 1.31×10^3 | 1.35×10^3 | 1.38×10^3 | 1.52×10^3 |
| Take-k based HS | | 1.28×10^{-1} | 1.32×10^{-1} | 1.36×10^{-1} | 3.77×10^2 | | 9.76×10^{-2} | 1.07×10^{-1} | 1.26×10^{-1} | 6.14×10^2 |
| SaDE | | 4.05×10^{-3} | 8.70×10^{-3} | 1.34×10^{-2} | 1.08×10^3 | | 5.79×10^3 | 6.97×10^3 | 8.16×10^3 | 2.36×10^3 |
| CoDE | | 2.25×10^{-3} | 4.75×10^{-3} | 7.25×10^{-3} | 5.38×10^2 | | 3.59×10^3 | 3.81×10^3 | 4.04×10^3 | 1.08×10^3 |
| CLPSO | | 5.14×10^{-1} | 5.51×10^{-1} | 5.87×10^{-1} | 3.81×10^3 | | 4.20×10^2 | 4.56×10^2 | 4.92×10^2 | 4.27×10^3 |
| Take-k based DIHS | | 5.90×10^{-34} | 1.01×10^{-32} | 2.43×10^{-32} | 4.85×10^2 | | 0 | 0 | 0 | 1.26×10^3 |
| HS (Take-all) | Schwefel shift | 5.75×10^1 | 5.81×10^1 | 5.87×10^1 | 1.32×10^3 | Griewank shift | 5.64×10^2 | 5.77×10^2 | 5.91×10^2 | 1.60×10^3 |
| Take-k based HS | | 4.85×10^1 | 5.13×10^1 | 5.43×10^1 | 3.70×10^2 | | 7.32×10^{-3} | 3.23×10^{-2} | 7.10×10^{-2} | 8.08×10^2 |
| SaDE | | 8.04×10^1 | 8.14×10^1 | 8.23×10^1 | 2.15×10^3 | | 1.19×10^{-3} | 4.11×10^{-2} | 8.09×10^{-2} | 1.83×10^3 |
| CoDE | | 1.15×10^2 | 1.16×10^2 | 1.17×10^2 | 6.57×10^2 | | 3.32×10^{-4} | 9.16 | 1.83×10^1 | 7.50×10^2 |
| CLPSO | | 6.99×10^1 | 7.48×10^1 | 7.97×10^1 | 4.30×10^3 | | 2.77×10^{-2} | 2.94×10^{-2} | 3.12×10^{-2} | 4.38×10^3 |
| Take-k based DIHS | | 2.56×10^1 | 2.60×10^1 | 2.64×10^1 | 6.39×10^2 | | 2.44×10^{-15} | 2.50×10^{-15} | 2.55×10^{-15} | 8.66×10^2 |
| HS (Take-all) | Rosenbrock shift | 2.21×10^9 | 2.37×10^9 | 2.48×10^9 | 1.58×10^3 | Ackley shift | 9.23 | 9.4 | 9.62 | 1.65×10^3 |
| Take-k based HS | | 2.10×10^3 | 2.28×10^3 | 2.62×10^3 | 7.99×10^2 | | 1.91×10^{-2} | 1.97×10^{-2} | 2.00×10^{-2} | 9.71×10^2 |
| SaDE | | 4.03×10^3 | 4.42×10^3 | 4.81×10^3 | 1.75×10^3 | | 1.58×10^1 | 1.59×10^1 | 1.60×10^1 | 1.86×10^3 |
| CoDE | | 2.84×10^3 | 2.89×10^3 | 2.94×10^3 | 6.18×10^2 | | 1.46×10^1 | 1.49×10^1 | 1.53×10^1 | 7.80×10^2 |
| CLPSO | | 5.94×10^3 | 6.26×10^3 | 6.58×10^3 | 4.12×10^3 | | 1.35×10^1 | 1.54×10^1 | 1.73×10^1 | 4.43×10^3 |
| Take-k based DIHS | | 1.15×10^3 | 1.15×10^3 | 1.16×10^3 | 7.28×10^2 | | 2.31×10^{-13} | 2.31×10^{-13} | 2.31×10^{-13} | 9.82×10^2 |

As shown in Table 2, the best/mean/worst solution of take-k-based DIHS is superior to those of the other algorithms for all functions. The take-k-based HS ($k = 0.01D$) has the shortest run time. Additionally, take-all-based HS requires more than twice the run time of the take-k-based HS algorithm. The take-k-based DIHS requires a slightly higher run time than the take-k-based HS algorithm. Compared to SaDE, CoDE and CLPSO, the take-k-based DIHS is optimal for all metrics.

5.3. Analysis of the Take-k-Based DIHS Results

As shown in Figures 4 and 7, for most test functions, the update success rate curve of the DIHS algorithm displays an “N-shaped” trajectory. The initial value is close to 50%, and this value then rapidly decreases to between 0.1% and 1%. Next, the curve slowly rises (to approximately 35%), and in the final stage, the success rate suddenly and rapidly drops to a very low value (close to 0).

We believe that this phenomenon is due to the following factors.

- (1) The population is randomly initialized, and every individual has a bad fitness value. In this case, DIHS has a higher probability than the other algorithms of finding an improved solution.
- (2) In the early stage of the search, the DIHS algorithm perturbs the search space to find the global optimal region with a high perturbation step fw , resulting in a rapid decrease in the probability of finding a better solution than the current solution.

5.4. Convergence Rate

To test the convergence ability of algorithms during the search, we employ the average convergence rate of HS for t generations. The average convergence rate at time t [32] is defined as Formula (12):

$$R(t|HM_0) = 1 - \left(\frac{f^{\text{opt}} - f^{\text{best}}(HM_t)}{f^{\text{opt}} - f^{\text{best}}(HM_0)} \right)^{1/t} \quad (12)$$

where HM_0 and HM_t respectively denote the initial population (harmony memory, HM) and the harmony memory after t generations. $f^{\text{best}}(HM_t)$ represents the best fitness value among the individuals of HM_t . f^{opt} denotes the optimal fitness of the problem.

Figure 8 shows the convergence rates of 8 complex benchmark functions by using standard HS (take-all-based HS), take-k-based HS and take-k-based DIHS algorithms. It can be seen from the figure that on all functions, the take-k-based method is superior to the take-all method, and the take-k-based DIHS has a higher convergence rate than other algorithms. At the beginning of the search process, the take-k-based DIHS can maintain a high convergence rate that declines slowly. In the second half of the

search, the take-k-based DIHS has a progressively higher convergence rate for most functions (e.g., Ackley, Griewank and so on). In particular, the convergence rate on the Rastrigin shift function grows to 1. The results of this experiment further validate that the take-k strategy and dynamic dimensionality reduction are correct and effective improvement strategies for improving harmony search to solve complex high-dimensional optimization problems.

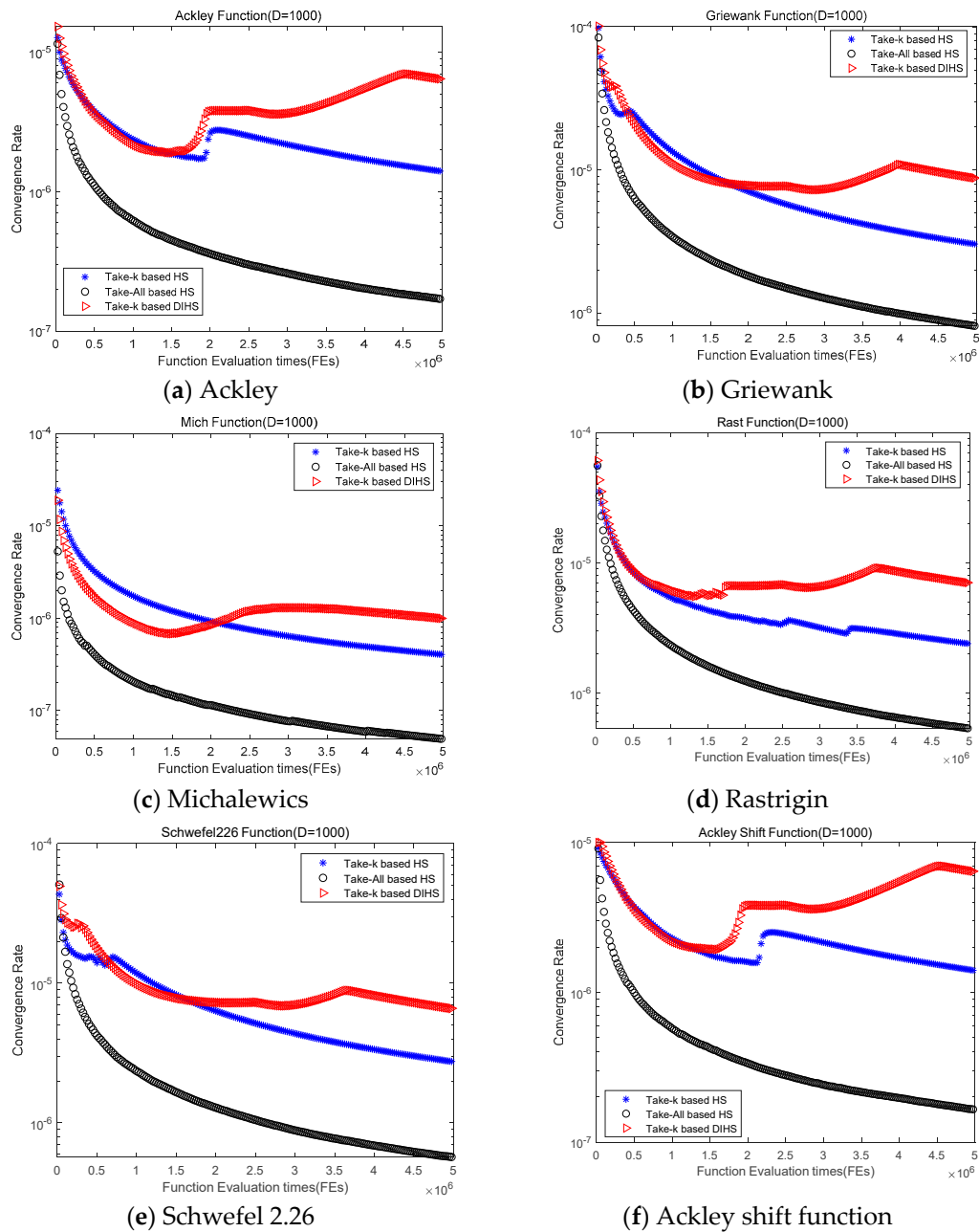


Figure 8. Cont.

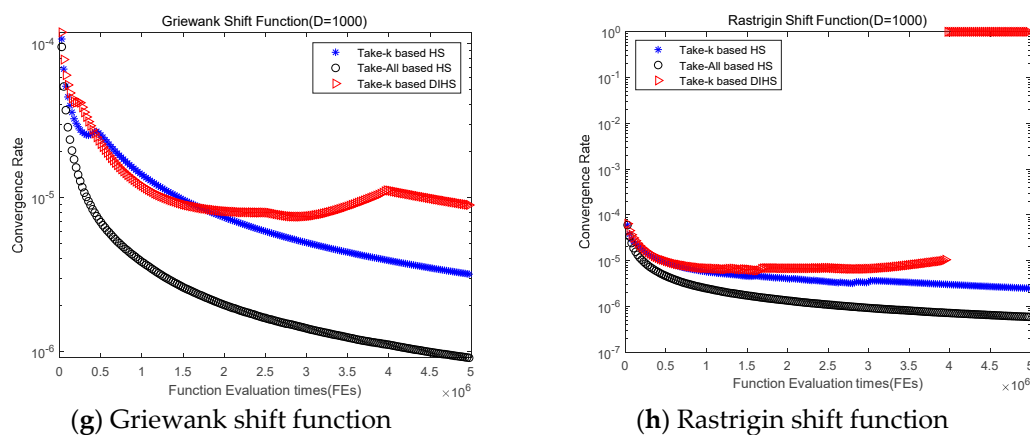


Figure 8. Convergence rates of 12 test functions by using three algorithms.

6. Discussion

Although many algorithms, such as cooperative coevolution (CC) algorithms [33–36] (decomposition-based algorithms), have been designed to solve high-dimensional optimization problems and achieved notable progress, they are still ineffective for some complex optimization problems [37]. This work aims to discover why effective HS algorithms that are able to solve small-scale optimization problems are not effective in solving high-dimensional optimization problems. By comparing take-k-based HS with take-all-based HS, it is found that take-all-based HS has a very low update success rate in the later search stage when solving high-dimensional optimization problems.

At the beginning of the search, a large k -value in the take-k-based HS is beneficial for accelerating the exploration process and obtaining a large gain value $G(\Delta t)$ at each iteration, but the update success rate decreases rapidly as the search progresses. A small k -value is beneficial for achieving a high update success rate, but the obtained gain value at each iteration is small.

To improve the performance of take-k-based HS, DIHS employs the dynamic take-k strategy, dynamic pitch adjustment rate and dynamic fret width. The experimental results demonstrate that the dynamic take-k strategy is very effective for balancing the update success rate and gain value in the search process. However, the take-k-based DIHS is still not effective for obtaining high-precision globally optimal solution for some high-dimensional optimization problems (e.g., Rosenbrock function) in which all decision variables are highly interconnected.

In this work, the goal is to improve the HS algorithm without employing any complex strategies, such as CC and decomposition-based algorithms. The proposed approach can be used in various applications and requires a search algorithm that is as simple as possible. Recently, the HS algorithm and its variants have received considerable attention, such as in portfolio selection [38] and searching for pathogenic sites based on the entire human genome [39–41]. For HS, improving the search speed and solving complex high-dimensional optimization problems remain an important focus of future research. The take-k-based strategy is a promising approach for improving the global search power, enhancing the solution quality and accelerating the speed.

Author Contributions: Conceptualization, S.T. and Z.W.G.; methodology, S.T.; formal analysis, S.T.; writing—original draft preparation, S.T.; writing—review and editing, S.T.; supervision, Z.W.G. and J.H.Y.; funding acquisition, S.T. and J.H.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Education of Humanities and Social Science project, China under Nos. 19YJCZH148. This work was also supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2020R1A2C1A01011131).

Conflicts of Interest: The authors declare no conflict of interest.

References

- Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A new heuristic optimization algorithm: Harmony search. *Simulation* **2001**, *76*, 60–68. [\[CrossRef\]](#)
- Geem, Z.W.; Kim, J.H.; Loganathan, G.V. Harmony search optimization: Application to pipe network design. *Int. J. Model. Simul.* **2002**, *22*, 125–133. [\[CrossRef\]](#)
- Lee, K.S.; Geem, Z.W. A new structural optimization method based on the harmony search algorithm. *Comput. Struct.* **2004**, *82*, 781–798. [\[CrossRef\]](#)
- Tuo, S.; Yong, L.; Deng, F.; Li, Y.; Lin, Y.; Lu, Q. A harmony search algorithm for high-dimensional multimodal optimization problems. *Digit. Sign. Process.* **2015**, *46*, 151–163. [\[CrossRef\]](#)
- Mahdavi, M.; Fesanghary, M.; Damangir, E. An improved harmony search algorithm for solving optimization problems. *Appl. Math. Comput.* **2007**, *188*, 1567–1579. [\[CrossRef\]](#)
- Deng, F.; Tuo, S.; Yong, L.; Zhou, T. Construction example for algebra system using harmony search algorithm. *Math. Probl. Eng.* **2015**, *2015*, 15. [\[CrossRef\]](#)
- Tuo, S.; Yong, L.; Deng, F.; Li, Y.; Lin, Y.; Lu, Q. HSTLBO: A hybrid algorithm based on Harmony Search and Teaching-Learning-Based Optimization for complex high-dimensional optimization problems. *PLoS ONE* **2017**, *12*, e0175114. [\[CrossRef\]](#)
- Li, Z.; Zou, D.; Kong, Z. A harmony search variant and a useful constraint handling method for the dynamic economic emission dispatch problems considering transmission loss. *Eng. Appl. Artif. Intell.* **2019**, *84*, 18–40. [\[CrossRef\]](#)
- Keshtegar, B.; Hao, P.; Wang, Y.; Li, Y. Optimum design of aircraft panels based on adaptive dynamic harmony search. *Thin Walled Struct.* **2017**, *118*, 37–45. [\[CrossRef\]](#)
- Xu, C.; Wang, X. Transient content caching and updating with modified harmony search for Internet of Things. *Digit. Commun. Netw.* **2019**, *5*, 24–33. [\[CrossRef\]](#)
- Yi, J.; Gao, L.; Li, X.A.; Shoemaker, C.; Lu, C. An on-line variable-fidelity surrogate-assisted harmony search algorithm with multi-level screening strategy for expensive engineering design optimization. *Knowl. Based Syst.* **2019**, *170*, 1–19. [\[CrossRef\]](#)
- Li, G.; Zeng, B.; Liao, W.; Li, X.; Gao, L. A new AGV scheduling algorithm based on harmony search for material transfer in a real-world manufacturing system. *Adv. Mech. Eng.* **2018**, *10*, 1–12. [\[CrossRef\]](#)
- Jiao, Y.; Wu, J.; Tan, Q.K.; Tan, Z.F.; Wang, G. An optimization model and modified Harmony Search algorithm for microgrid planning with ESS. *Discret. Dyn. Nat. Soc.* **2017**, *11*, 1–11. [\[CrossRef\]](#)
- Zhao, F.; Liu, Y.; Zhang, Y.; Ma, W.; Zhang, C. A hybrid harmony search algorithm with efficient job sequence scheme and variable neighborhood search for the permutation flow shop scheduling problems. *Eng. Appl. Artif. Intell.* **2017**, *65*, 178–199. [\[CrossRef\]](#)
- Saka, M.P.; Hasançebi, O.; Geem, Z.W. Metaheuristics in structural optimization and discussions on harmony search algorithm. *Swarm Evolut. Comput.* **2016**, *28*, 88–97. [\[CrossRef\]](#)
- Moon, Y.Y.; Geem, Z.W.; Han, G.T. Vanishing point detection for self-driving car using harmony search algorithm. *Swarm Evolut. Comput.* **2018**, *41*, 111–119. [\[CrossRef\]](#)
- Kim, Y.K.; Yoon, Y.; Geem, Z.W. A comparison study of harmony search and genetic algorithm for the max-cut problem. *Swarm Evolut. Comput.* **2019**, *44*, 130–135. [\[CrossRef\]](#)
- Das, S.; Mukhopadhyay, A.; Roy, A.; Abraham, A.; Panigrahi, B.K. Exploratory power of the harmony search algorithm: Analysis and improvements for global numerical optimization. *IEEE Trans. Syst. Man Cybern. Part B (Cybernetics)* **2011**, *41*, 89–106. [\[CrossRef\]](#)
- Pan, Q.-K.; Suganthan, P.N.; Tasgetiren, M.F.; Liang, J.J. A self-adaptive global best harmony search algorithm for continuous optimization problems. *Appl. Math. Comput.* **2010**, *216*, 830–848. [\[CrossRef\]](#)
- Peraza, C.; Valdez, F.; Garcia, M.; Melin, P.; Castillo, O. A new fuzzy harmony search algorithm using fuzzy logic for dynamic parameter adaptation. *Algorithms* **2016**, *9*, 19. [\[CrossRef\]](#)
- Kattan, A.; Abdullah, R. A dynamic self-adaptive harmony search algorithm for continuous optimization problems. *Appl. Math. Comput.* **2013**, *219*, 8542–8567. [\[CrossRef\]](#)
- Assad, A.; Deep, K.A. Hybrid harmony search and simulated annealing algorithm for continuous optimization. *Inform. Sci.* **2018**, *450*, 246–326. [\[CrossRef\]](#)
- Zhang, T.; Geem, Z.W. Review of harmony search with respect to algorithm structure. *Swarm Evolut. Comput.* **2019**, *48*, 31–43. [\[CrossRef\]](#)

24. Manjarres, D.; Landa-Torres, I.; Gil-Lopez, S.; Del Ser, J.; Bilbao, M.N.; Salcedo-Sanz, S.; Geem, Z.W. A survey on applications of the harmony search algorithm. *Eng. Appl. Artif. Intell.* **2013**, *26*, 1818–1831. [\[CrossRef\]](#)
25. Nasir, M.; Sadollah, A.; Yoon, J.H.; Geem, Z.W. Comparative study of Harmony Search algorithm and its applications in China, Japan and Korea. *Appl. Sci.* **2020**, *10*, 3970. [\[CrossRef\]](#)
26. Fukushima, M. *Test Functions for Unconstrained Global Optimization*; Springer: New York, NY, USA, 2006; Available online: http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm (accessed on 10 June 2018).
27. Tang, K.; Yao, X.; Suganthan, P.N.; MacNish, C.; Chen, Y.P.; Chen, C.M.; Yang, Z. *Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization*; China & Nanyang Technological University: Singapore, 2008; Available online: <http://www.ntu.edu.sg/home/EPNSugan/> (accessed on 10 June 2018).
28. Tang, K.; Li, X.; Suganthan, P.N.; Yang, Z.; Weise, T. *Benchmark Functions For The CEC'2010 Special Session And Competition On Large Scale Global Optimization*; Technical report; Nature Inspired Computation and Applications Laboratory, USTC, China & Nanyang Technological University: Singapore, 2009; Available online: <http://nical.ustc.edu.cn/cec10ss.php> (accessed on 10 June 2018).
29. Qin, A.K.; Huang, V.L.; Suganthan, P.N. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Trans. Evol. Comput.* **2009**, *13*, 398–417. [\[CrossRef\]](#)
30. Wang, Y.; Cai, Z.; Zhang, Q. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Trans. Evol. Comput.* **2011**, *15*, 55–66. [\[CrossRef\]](#)
31. Liang, J.J.; Qin, A.K.; Suganthan, P.N.; Baskar, S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comput.* **2006**, *10*, 281–295. [\[CrossRef\]](#)
32. Potter, M.A.; Jong, K.A.D. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolut. Comput.* **2000**, *8*, 1–29. [\[CrossRef\]](#)
33. He, J.; Lin, G. Average convergence rate of evolutionary algorithms. *IEEE Trans. Evol. Comput.* **2015**, *20*, 316–321. [\[CrossRef\]](#)
34. Omidvar, M.; Li, X.; Mei, Y.; Yao, X. Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Trans. Evol. Comput.* **2014**, *18*, 378–393. [\[CrossRef\]](#)
35. Omidvar, M.N.; Kazimipour, B.; Li, X.; Yao, X. Cbccc3- a contribution-based cooperative co-evolutionary algorithm with improved exploration/exploitation balance. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24 July 2016; Volume 7, pp. 3541–3548.
36. Omidvar, M.N.; Yang, M.; Mei, Y.; Li, X.; Yao, X. DG2: A faster and more accurate differential grouping for large-scale black-box optimization. *IEEE Trans. Evol. Comput.* **2017**, *21*, 929–942. [\[CrossRef\]](#)
37. Liu, J.; Peng, H.; Wu, Z.; Chen, J.; Deng, C. A hybrid deep grouping algorithm for large scale global optimization. *IEEE Trans. Evol. Comput.* **2020**. [\[CrossRef\]](#)
38. Tuo, S.H.; He, H. Solving complex cardinality constrained mean-variance portfolio optimization problems using hybrid HS and TLBO algorithm economic computation and economic cybernetics studies and research. *Acad. Econ. Stud.* **2018**, *52*, 231–248.
39. Tuo, S.H.; Zhang, J.; Yuan, X.; He, Z.; Liu, Y.; Liu, Z. Niche harmony search algorithm for detecting complex disease associated high-order SNP combinations. *Sci. Rep.* **2017**, *7*, 11529. [\[CrossRef\]](#)
40. Tuo, S.H.; Zhang, J.; Yuan, X.; Zhang, Y.; Liu, Z. FHSA-SED: Two-locus model detection for genome-wide association study with harmony search algorithm. *PLoS ONE* **2016**, *11*, e0150669. [\[CrossRef\]](#)
41. Tuo, S.H.; Liu, H.; Chen, H. Multi-population harmony search algorithm for the detection of high-order SNP interactions. *Bioinformatics* **2020**. [\[CrossRef\]](#)

