algorithms

MDPI

Article

Kernel Clustering with a Differential Harmony Search Algorithm for Scheme Classification

Yu Feng, Jianzhong Zhou * and Muhammad Tayyab

School of Hydropower and Information Engineering, Huazhong University of Science and Technology, Wuhan 430074, China; ferrychip@hust.edu.cn (Y.F.); mtayyab@hust.edu.cn (M.T.)
* Correspondence: jz.zhou@hust.edu.cn; Tel.: +86-27-8754-3127

Academic Editor: Javier Del Ser Lorente
Received: 8 October 2016; Accepted: 11 January 2017; Published: 14 January 2017

Abstract: This paper presents a kernel fuzzy clustering with a novel differential harmony search algorithm to coordinate with the diversion scheduling scheme classification. First, we employed a self-adaptive solution generation strategy and differential evolution-based population update strategy to improve the classical harmony search. Second, we applied the differential harmony search algorithm to the kernel fuzzy clustering to help the clustering method obtain better solutions. Finally, the combination of the kernel fuzzy clustering and the differential harmony search is applied for water diversion scheduling in East Lake. A comparison of the proposed method with other methods has been carried out. The results show that the kernel clustering with the differential harmony search algorithm has good performance to cooperate with the water diversion scheduling problems.

Keywords: kernel fuzzy clustering; differential harmony search; water diversion scheduling

1. Introduction

Metaheuristics are designed to find, generate, or select a heuristic that provides a good solution to an optimization problem [1]. By searching over a large set of feasible solutions, metaheuristics can often find good solutions with less computational effort [2]. There has been a widespread usage of metaheuristics and their applications in artificial intelligence, e.g., transit network design problems [3], sewer pipe networks [4], water distribution systems [5], sizing optimization of truss structures [6], ordinary differential equations [7], and so forth. Using metaheuristic algorithms, complex problems are not far from finding their solutions [7]. Metaheuristics are good at finding near-optimal solutions to numerical real-valued problems [8].

Harmony search (HS) is a new phenomenon-mimicking algorithm, proposed by Geem in 2001 [9]. It is a relatively recent metaheuristic method based on musical performances. In the HS algorithm, a new solution is generated by pitch adjustment and random selection. HS has the ability to deal with discontinuous optimization problems, as well as continuous problems. Comparing with other artificial intelligent algorithms such as the genetic algorithm and its variants, the HS algorithm requires fewer parameters and these parameters are easy to set. Moreover, HS can overcome the drawback of GA's building block theory. The advantages of HS have resulted in much interest in recent years, and HS has been widely applied in many fields, such as email classification [10], single machine scheduling problems [11], and so on.

Although HS has its advantages, i.e., it is good at identifying the high-performance regions of the solution space at a reasonable time, it gets into trouble in performing local searches for numerical applications [12]. In order to improve the optimization ability, different variants of HS have been developed. Mahdavi et al. presented an improved harmony search algorithm (IHS) which changes the parameters dynamically with the generation number [12]. The IHS algorithm presents a strategy of parameter adjustment which improves the performances of HS algorithm. However, the user needs to specify the new parameters, which are not easy to set, and it still performs local searches for some numerical applications. Other modifications, such as the global-best harmony search algorithm (GHS) [13] and chaotic harmony search algorithms (CHS) [14], have shown better performance than the classical HS, but they still have their own shortcomings. The GHS algorithm generates new harmonies using the variable from the best harmony. However, the algorithm cannot be adopted when the variables have different value ranges. This drawback limits the scope of the application of this method. The CHS generates new solutions following the chaotic map, but simulations show that CHS still suffers from a local optimum when dealing with some numerical applications.

Fuzzy clustering is a class of algorithms for cluster analysis which determine the affinities of samples using mathematical methods. It divides samples into classes or clusters so that items in the same class are as similar as possible, while items in different classes are dissimilar. It is a useful technique for analyzing statistical data in multi-dimensional space.

Fuzzy clustering has gained a significant level of interest for its ability to classify the samples which are multi-attributed and difficult to analyze. In recent years, there have been a large number of studies on fuzzy clustering. Fuzzy c-means clustering (FCM) [14] and fuzzy k-means clustering (FKM) [15] are widely used to categorize similar data into clusters. The kernel fuzzy clustering algorithm (KFC) [16] and the weighted fuzzy kernel-clustering algorithm (WFKCA) [17] are also efficient ways for cluster analysis. Research has shown that WFKCA has good convergence properties and the prototypes obtained can be well represented in the original space. However, these clustering methods have the same problem: the iterative solution is not the optimal solution. In order to overcome this drawback, in this paper, we combine the KFC with the HS algorithm to help the KFC perform better.

Although the modifications of HS have shown a better performance than the classical HS, the performances still need improvement. In this paper, we proposed a new differential harmony search algorithm (DHS), and applied it to kernel fuzzy clustering. A comparison of the proposed method with other methods has been carried out. Finally, the proposed method is applied to the water diversion scheduling assessment in East Lake, which is a new study in the East Lake Ecological Water Network Project. The water diversion scheduling tries to divert water from the Yangtze River to the East Lake Network, aiming to improve water quality in the sub-lakes. Using a hydrodynamic simulation model and a water quality model, the quality of the lakes at the end of the water diversion can be simulated. In order to obtain a better improvement of the water quality and reduce the economic cost, the water diversion scheme must be carefully developed. The diversion scheduling in the East Lake Network is a multi-objective problem, however, multi-objective evolutionary algorithms cannot be adopted because the water quality simulation is time-consuming. Thus, we made some typical schemes in the feasible range, and obtain the scheme results by the water quality model. The purpose of the kernel clustering with differential harmony search method is to classify the results in order to find out the schemes which perform better than others.

This paper is organized as follows: Section 2 presents an overview of the harmony search algorithm and kernel fuzzy clustering; Section 3 describes the modification and the combination of kernel fuzzy clustering and the differential harmony search algorithm; Section 4 discusses the computational results; and Section 5 provides the summary of this paper.

2. Harmony Search and Kernel Fuzzy Clustering

2.1. Harmony Search Algorithm

Harmony search (HS) is a phenomenon-mimicking algorithm inspired by the improvisation process of musicians proposed by Geem in 2001 [9]. The method is a population-based evolutionary algorithm and it is a simple and effective solution to find a result which optimizes an objective function. Parameters of the harmony search algorithm usually include harmony memory size (*hms*), harmony memory considering rate (*hmcr*), pitch adjustment rate (*par*), and fret width (*fw*).

The main steps of the classical harmony search mainly include memory consideration, pitch adjustment, and randomization. Details of HS are explained as follows:

Step 1 Initialize algorithm parameters

This step specifies the HS algorithm parameters, including *hms*, *hmcr*, *par*, and *fw*.

Step 2 Initializing the harmony memory

In HS, each solution is called a harmony. The harmony memory (HM) is equivalent to the population of other population-based algorithms. In HM, all solution vectors are stored. In this step, random vectors, as many as *hms*, are generated following Equation (1):

$$x_i^j = (UB_i - LB_i) \cdot U(0,1) + LB_i, i = 1, 2, \cdots, D \tag{1}$$

where LB_i and UB_i are the lower and upper bounds of the *i*th variable. D represents the dimensions of the problem. $U(0, 1)$ is a uniformly-distributed random number between 0 and 1.

Step 3 Improvising a new harmony

In this step, a new harmony $x' = (x'_1, x'_2, \cdots, x'_D)$ is generated based on three rules: memory consideration, pitch adjustment, and random selection. The procedure is shown in Algorithm 1.

Algorithm 1 Improvisation of a New Harmony

```
For $i$ = 1 to $D$ do
  If $U(0,1) \leq hmcr$ then
    $x'_i = x_i^j$ where $j$ is a random integer from $(1, 2, \cdots, hms)$
    If $U(0,1) \leq par$ then
      $x'_i = x'_i + U(-1,1) \times fw$
    end if
  else
    $x'_i = (UB_i - LB_i) \cdot U(0,1) + LB_i$
  end if
end for
```

Step 4 Update harmony memory

If the new harmony generated in Step 2 is better than the worst harmony stored in HM, replace the worst harmony with the new harmony.

Step 5 Check the terminal criteria

If the maximum number of improvisations (NI) is reached, then stop the algorithm. Otherwise, the improvisation will continue by repeating Steps 3 to 4.

2.2. Kernel Fuzzy Clustering

In the past decade, several studies on the kernel fuzzy clustering method (KFC) have been conducted [18,19]. A kernel method means using kernel functions to map the original sample data to a higher-dimensional space without ever computing the coordinates of the data in that space. The kernel method takes advantage of the fact that dot products in the kernel space can be expressed by a Mercer kernel K, given by $K(\mathrm{x}, \mathrm{y}) \equiv \Phi(\mathrm{x})^T \Phi(\mathrm{y})$, where $x, y \in R^d$ [20].

Suppose the sample dataset is given as $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_n\}$, and each sample has K attributes $\mathbf{X}_i = \{X_{i1}, X_{i2}, ..., X_{iK}\}$. Firstly, the original sample data is normalized following Equation (2):

$$x_{jk} = \frac{X_{jk} - X_k^{\min}}{X_k^{\max} - X_k^{\min}} \tag{2}$$

The purpose of kernel fuzzy clustering is to minimize the following objective function:

$$J = \sum_{i=1}^{C} \sum_{j=1}^{N} u_{ij}^m \, ||\Phi\left(\mathbf{x}_j'\right) - \Phi(\mathbf{v}_i')\,||^2 \tag{3}$$

$$\mathbf{x}_j' = \operatorname{diag}(\mathbf{w}) \cdot \mathbf{x}_j \tag{4}$$

$$\mathbf{v}_i' = \operatorname{diag}(\mathbf{w}) \cdot \mathbf{v}_i \tag{5}$$

Constraints are:

$$\sum_{i=0}^{C} u_{ij} = 1, 1 \le i \le N \tag{6}$$

$$\sum_{k=1}^{K} w_k = 1 \tag{7}$$

where C is the number of clusters, N is the number of samples, K is the number of features, $\mathbf{v}_i = [v_{i1}, v_{i2}, \ldots, v_{iL}]$ is the ith cluster center, u_{ij} is the membership of $\mathbf{x_i}$ in class i, $\mathbf{w} = [w_1, w_2, \cdots, w_K]^T$ is the feature-weight vector, and m is coefficient, $m > 1$.

According to the relevant literature [17,20,21], Equation (3) is simplified as:

$$J = 2\sum_{i=1}^{C} \sum_{j=1}^{N} u_{ij}^m \left(1 - K\left(\mathbf{x}_j', \mathbf{v}_i'\right)\right) \tag{8}$$

where $K(\mathbf{x}, \mathbf{x}')$ is a Mercer kernel function. In this paper, the (Gaussian) radial basis function (RBF) kernel is adopted:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'\,||_2^2}{2\sigma^2}\right) \tag{9}$$

The clustering center and membership matrix is:

$$u_{ij} = \left(\sum_{r=1}^{C} \frac{1 - K\left(\mathbf{x}_j', \mathbf{v}_i'\right)}{1 - K\left(\mathbf{x}_j', \mathbf{v}_i'\right)}\right)^{-\frac{1}{m-1}} \tag{10}$$

$$\mathbf{v}_i = \frac{\sum\limits_{j=1}^{N} u_{ij} \cdot K\left(\mathbf{x}_j', \mathbf{v}_i'\right) \cdot x_j}{\sum\limits_{j=1}^{N} u_{ij} \cdot K(\mathbf{x}_j{}', \mathbf{v}_i{}')} \tag{11}$$

3. DHS-KFC

3.1. Differential Harmony Search Algorithm

The harmony search (HS) algorithm tries to replace the worst item in the harmony memory (HM) if the generated harmony is better. However, some local optimums in HM have the probability of remaining unchanged for a long time. Additionally, the guide from the best harmony can also improve the local search ability of the algorithm. In this section, a novel modification of the harmony search algorithm, named differential harmony search (DHS), is proposed to help the algorithm converge faster.

In the proposed method, the harmonies in the HM which meet the specific conditions will be changed following the differential evolution strategy. Moreover, the new generated harmonies will consider the best vector.

3.1.1. A Self-Adaptive Solution Generation Strategy

The purpose of this strategy is to improve the local search ability of the algorithm. In the pitch adjustment step, the classical HS changes the random harmony selected from HM by small perturbations. Different from classical HS, the pitch adjustment in DHS will consider the best vector, as follows:

$$x_i' = \begin{cases} x_i' + fw \cdot U(-1,1) & ,\mathrm{u}(0,1) < \frac{1}{N} \cup \mathrm{u}(0,1) < cr \\ x_i^{best} + fw \cdot U(-1,1) & ,\text{otherwise} \end{cases} \tag{12}$$

where *cr* is the crossover probability; it generally varies from 0 to 1. Through a large number of numerical simulations, we obtain the conclusion that the suitable range of *cr* is [0.4, 0.9].

In this paper, the parameter *cr* is dynamically adjusted following Equation (13). When *cr* is smaller, the new harmony has a higher probability to use the value coming from the best vector, which means that the convergence speed will be faster; when *cr* is larger, the harmonies will retain their own diversity. Overall, using this strategy, the algorithm will have better local search ability in the early stage.

$$cr = 0.4 + \frac{CI}{2 \times \mathrm{NI}} \tag{13}$$

where *CI* is the current iteration, and NI is the maximum iteration.

3.1.2. A Differential Evolution-Based Population Update Strategy

The purpose of this strategy is to help the algorithm avoid from local convergence. When updating HM with the new harmony, some local optimums in HM have the probability of remaining unchanged for a long time. In DHS, if the harmony in HM remains unchanged for several iterations (sc = 20), it will be replaced with a new harmony using the differential evolution strategy shown in Equation (14):

$$x_i' = \begin{cases} x_i + fw \cdot (x_i'' - x_i''') & ,\mathrm{u}(0,1) < \frac{1}{N} \cup \mathrm{u}(0,1) < cr \\ x_i & ,\text{otherwise} \end{cases} \tag{14}$$

where x_i'' and x_i''' are random stored values from HM. Using the dynamic parameter *cr*, the harmonies have higher probability of variation in the later stage.

3.1.3. Implementation of DHS

The DHS algorithm consists of five steps, as follows:

Step 1 Initialize the algorithm parameters

This step specifies parameters, including harmony memory size (*hms*), harmony memory considering rate (*hmcr*), pitch adjustment rate (par), fret width (*fw*), and memory keep iteration (*sc*).

Step 2 Initialize the harmony memory

This step is consistent with the basic harmony search algorithm. New random vectors $(\mathbf{x}^1, \ldots, \mathbf{x}^{hms})$, as many as *hms*, are generated following Equation (15):

$$x_i^j = (UB_i - LB_i) \cdot U(0,1) + LB_i, i = 1,2,\ldots,D \tag{15}$$

Then each vector will be evaluated by the objective function and stored in the HM.

Step 3 Improvising a new harmony

In this step, a new random vector $x' = (x'_1, x'_2, \ldots, x'_D)$ is generated considering the self-adaptive solution generation strategy. The procedure is shown in Algorithm 2.

Algorithm 2 Improvisation of a New Harmony of DHS

```
for $i = 1, \cdots, n$ do
    if ($U(0,1) \leq hmcr$) then
        $x'_i = x_i^j, where\ j \sim U(1, hms)$
        if ($U(0,1) < par$) then
            $shift = U(-1,1) \times fw \times (UB_i - LB_i)$
            if ($U(0,1) < \frac{1}{N} \| U(0,1) < cr$) then
                $x'_i = x_i^{best} + shift$
            else
                $x'_i = x'{}_i + shift$
            end if
        end if
    else
        $x'_i = LB_i + U(0,1) \times (UB_i - LB_i)$
    end if
end for
```

Step 4 Update the harmony memory

If the vector generated in Step 3 is better than the worst vector in HM, replace the worst vector with the generated vector.

If the vector in HM remains unchanged for several iterations (*sc* = 20), replace it with a new vector generated, following Equation (14), if the new vector is better.

The procedure of step 4 is shown in Algorithm 3.

Algorithm 3 Update the Harmony Memory

```
if ($f(\mathbf{x}') < f(\mathbf{x}^{worst})$) then
    Replace $\mathbf{x}^{worst}$ with $\mathbf{x}'$
    Set $flag(worst) = 0$
end if
for $r = 1, \cdots,$ hms do
    $flag(r) = flag(r) + 1$
    if ($flag(r) > sc$) then
        for $i = 1, \cdots, n$ do
            if ($U(0,1) < \frac{1}{N} \| U(0,1) < cr$) then
                $shift = U(-1,1) \times (x_i^{r1} - x_i^{r2}), \forall r1, r2 = 1, 2, \cdots, hms$ and $r1 \neq r2$
                $x''_i = x_i^r + shift$
            else
                $x''_i = x_i^r$
            end if
        end for
        if ($f(\mathbf{x}'') < f(\mathbf{x}^r)$) then
Replace $\mathbf{x}^r$ with $\mathbf{x}''$
        end if
    end if
end for
```

Step 5 Check the stop criterion

Repeat Step 3 to Step 4 until the termination criterion is satisfied. In this paper, the termination criterion is the maximum number of evaluations.

3.2. DHS-KFC

In this paper, we use the proposed DHS to search the optimal result of the kernel fuzzy clustering (KFC). When DHS is applied to KFC, we must define the optimization variables. In this paper, the cluster center $\mathbf{v} = \{v_1, \cdots, v_n\}$ is chosen as the optimization variables. The weight matrix $\mathbf{w} = [w_1, \cdots, w_K]^T$ is given by the experts using a method which determines the relative importance of attributes, such as Fuzzy Analytic Hierarchy Process (FAHP) [22]. The membership matrix $\mathbf{u} = \{u_1, \cdots, u_n\}$ can be obtained by Equation (10). The objective function of KFC is shown as:

$$J = \sum_{i=1}^{C} \sum_{j=1}^{N} u_{ij}^{m} \, ||\Phi\left(\mathbf{x}_j'\right) - \Phi\left(\mathbf{v}_i'\right)||^2 \tag{16}$$

The DHS algorithm will find the optimized result which minimizes the objective function. The procedure is described as follows (Figure 1):

Step 1 initialize the parameters of the harmony search algorithm, initialize the harmony memory.

Step 2 initialize the parameters of the KFC, maximum generation N, and the weight matrix $\mathbf{w}$, set the initial value of the cluster center matrix $\mathbf{v}^0$ to a randomly-generated matrix. Then the membership matrix $\mathbf{u}^0$ can be obtained from Equation (10).

Step 3 generate a new solution vector based on the harmony search algorithm.

Step 4 obtain the cluster center matrix $\mathbf{v}^n$ from the solution vector, calculate the membership matrix $\mathbf{u}^n$ based on Equation (10), and then calculate J^n based on Equation (16).

Step 5 compare J^n with J^{n-1}. If J^n remains unchanged until 10 iterations, and go to Step 7.

Step 6 set the current iteration $n = n + 1$. If $n > N$ go to Step 7, otherwise go to Step 3.

Step 7 classify the samples based on their membership.

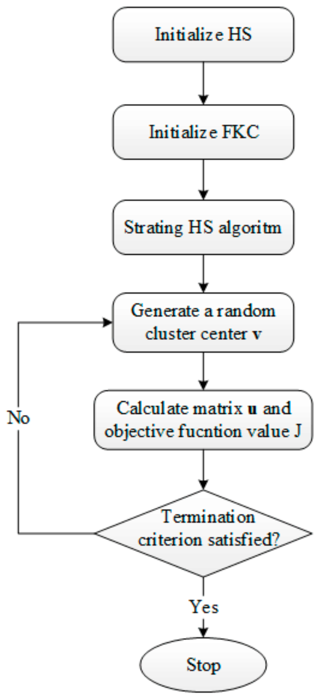

Initialize HS
Initialize FKC
Strating HS algoritm
Generate a random cluster center v
Calculate matrix **u** and objective fucntion value J
Termination criterion satisfied?
No
Yes
Stop

Figure 1. Framework of the proposed method.

4. Experiments

4.1. Numerical Experiments of DHS

4.1.1. Benchmark Function Tests

To evaluate the performance of the proposed DHS, we chose several famous typical benchmark functions, shown in Table 1. These functions are tested by simulating the HS, IHS, GHS, CHS, and the proposed DHS algorithms. Of the functions in Table 1, the Ackley function, Griewank function, Rastrigin function, and Rosenbrock function are multimodal functions. The Sphere function and Schwefel 2.22 function are unimodal functions.

Table 1. Test functions.

Function	Formula	Search Domain	Optimum
Ackley function	$f(x_1,\cdots,x_n) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + e + 20$	$-40 \le x_i \le 40$	$f(0,\cdots,0) = 0$
Griewank function	$f(x_1,\cdots,x_n) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	$-600 \le x_i \le 600$	$f(0,\cdots,0) = 0$
Rastrigin function	$f(x_1,\cdots,x_n) = An + \sum_{i=1}^{n}\left[x_i^2 - A\cos(2\pi x_i)\right]$	$A = 10$ $-5.12 \le x_i \le 5.12$	$f(0,\cdots,0) = 0$
Rosenbrock function	$f(x_1,\cdots,x_n) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	$-\infty \le x_i \le \infty$	$f(1,\cdots,1) = 0$
Sphere function	$f(x_1,\cdots,x_n) = \sum_{i=1}^{n} x_i^2$	$-5.12 < x_i < 5.12$	$f(0,\cdots,0) = 0$
Schwefel 2.22 function	$f(x_1,\cdots,x_n) = \sum_{i=1}^{n}\lvert x_i\rvert + \prod_{i=0}^{n}\lvert x_i\rvert$	$-10 < x_i < 10$	$f(0,\cdots,0) = 0$

Parameters of the algorithms are shown in Table 2. All of the experiments are performed using a computer with 3.80 GHz AMD Athlon x4 760k with 8 GB RAM. The source code is compiled with Java SE8.

Table 2. Parameters of the algorithms.

Algorithm	*hms*	*hmcr*	*par*	*fw*	NI
HS	50	0.9	0.3	0.005	100,000
IHS	50	0.9	Max: 0.5 Min: 0.1	Max: 0.01 Min: 0.001	100,000
GHS	50	0.9	0.3	0.005	100,000
CHS	50	0.9	0.3	0.005	100,000
DHS	50	0.9	0.3	0.005	100,000

We evaluated the optimization algorithms based on 10 dimensional versions of the benchmark functions. The maximum evaluation count (NI) is set to 100,000. Each function is tested 100 times for each algorithm.

Table 3 shows the maximum, minimum, means, and standard deviation of errors of the algorithms on each benchmark function. Table 4 shows the distribution of the results simulated by the algorithms. As demonstrated in Tables 3 and 4, it is clear that DHS outperforms the other variants of HS in almost all of the functions except the Rosenbrock function. In these cases, the minimum, maximum, means, and standard deviations obtained by DHS are smaller than the results obtained by HS and its variants.

When dealing with the Rosenbrock function, the global minimum of the Rosenbrock function is inside a long, narrow, parabolic-shaped flat valley, meaning that convergence to the global minimum is difficult. Although the minimum of GHS is smaller than DHS, the maximum, mean, and standard deviation show that DHS is more stable than GHS.

Table 3. Errors of test results of the algorithms.

Function		HS	IHS	GHS	CHS	DHS
Ackley function	Mean	8.49×10^{-3}	6.16×10^{-3}	3.95×10^{-3}	1.12×10^{-2}	1.57×10^{-13}
	Min	3.29×10^{-3}	2.44×10^{-3}	1.35×10^{-5}	3.56×10^{-3}	4.57×10^{-14}
	Max	1.62×10^{-2}	1.16×10^{-2}	2.10×10^{-2}	2.54×10^{-2}	4.69×10^{-13}
	Stdv	2.43×10^{-3}	1.67×10^{-3}	3.48×10^{-3}	3.79×10^{-3}	6.78×10^{-14}
Griewank function	Mean	1.58×10^{-2}	1.46×10^{-2}	5.00×10^{-4}	1.37×10^{-2}	5.45×10^{-10}
	Min	8.62×10^{-5}	1.92×10^{-5}	1.59×10^{-7}	9.62×10^{-5}	0
	Max	8.13×10^{-2}	7.38×10^{-2}	1.13×10^{-2}	9.15×10^{-2}	1.69×10^{-8}
	Stdv	1.81×10^{-2}	1.76×10^{-2}	1.24×10^{-3}	1.54×10^{-2}	2.06×10^{-9}
Rastrigin function	Mean	1.49×10^{-4}	8.20×10^{-5}	7.44×10^{-5}	2.45×10^{-4}	1.02×10^{-12}
	Min	2.66×10^{-5}	1.35×10^{-5}	1.82×10^{-8}	4.33×10^{-5}	0
	Max	3.30×10^{-4}	2.01×10^{-4}	1.17×10^{-3}	9.92×10^{-4}	6.20×10^{-11}
	Stdv	7.13×10^{-5}	4.22×10^{-5}	1.64×10^{-4}	1.42×10^{-4}	6.17×10^{-12}
Rosenbrock function	Mean	2.10	1.98	1.77	2.32	7.16×10^{-1}
	Min	5.65×10^{-3}	1.00×10^{-2}	1.80×10^{-6}	1.04×10^{-2}	6.22×10^{-4}
	Max	5.32	5.34 10	8.37	6.08	4.64
	Stdv	1.57	1.51	3.10	1.69	9.75×10^{-1}
Sphere function	Mean	7.33×10^{-7}	4.30×10^{-7}	3.35×10^{-7}	1.01×10^{-6}	1.85×10^{-28}
	Min	1.40×10^{-7}	1.02×10^{-7}	3.00×10^{-14}	2.21×10^{-7}	2.92×10^{-29}
	Max	2.03×10^{-6}	1.43×10^{-6}	3.58×10^{-6}	3.55×10^{-6}	7.97×10^{-28}
	Stdv	3.83×10^{-7}	2.24×10^{-7}	6.13×10^{-7}	6.07×10^{-7}	1.72×10^{-28}
Schwefel 2.22 function	Mean	3.07×10^{-3}	2.41×10^{-3}	3.23×10^{-3}	3.68×10^{-3}	3.12×10^{-12}
	Min	1.29×10^{-3}	1.12×10^{-3}	1.62×10^{-6}	1.23×10^{-3}	7.35×10^{-13}
	Max	5.12×10^{-3}	3.91×10^{-3}	1.32×10^{-2}	6.62×10^{-3}	1.33×10^{-11}
	Stdv	9.05×10^{-4}	5.91×10^{-4}	2.86×10^{-3}	1.08×10^{-3}	1.99×10^{-12}

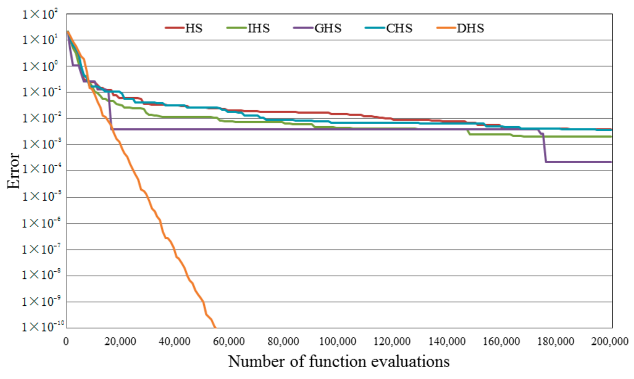
Table 4. Distribution of error of the algorithms.

Function	Precision	HS	IHS	GHS	CHS	DHS
Ackley function	$<1 \times 10^{-1}$	100%	100%	100%	100%	100%
	$<1 \times 10^{-2}$	74%	99%	93%	40%	100%
	$<1 \times 10^{-3}$	0%	0%	23%	0%	100%
	$<1 \times 10^{-4}$	0%	0%	2%	0%	100%
	$<1 \times 10^{-5}$	0%	0%	0%	0%	100%
	$<1 \times 10^{-6}$	0%	0%	0%	0%	100%
	$<1 \times 10^{-7}$	0%	0%	0%	0%	100%
Griewank function	$<1 \times 10^{-1}$	100%	100%	100%	100%	100%
	$<1 \times 10^{-2}$	32%	44%	99%	32%	100%
	$<1 \times 10^{-3}$	30%	44%	84%	27%	100%
	$<1 \times 10^{-4}$	1%	7%	47%	1%	100%
	$<1 \times 10^{-5}$	0%	0%	17%	0%	100%
	$<1 \times 10^{-6}$	0%	0%	5%	0%	100%
	$<1 \times 10^{-7}$	0%	0%	0%	0%	100%
	$<1 \times 10^{-1}$	100%	100%	100%	100%	100%
Rastrigin function	$<1 \times 10^{-2}$	100%	100%	100%	100%	100%
	$<1 \times 10^{-3}$	100%	100%	99%	100%	100%
	$<1 \times 10^{-4}$	25%	69%	81%	11%	100%
	$<1 \times 10^{-5}$	0%	0%	43%	0%	100%
	$<1 \times 10^{-6}$	0%	0%	14%	0%	100%
	$<1 \times 10^{-7}$	0%	0%	4%	0%	100%

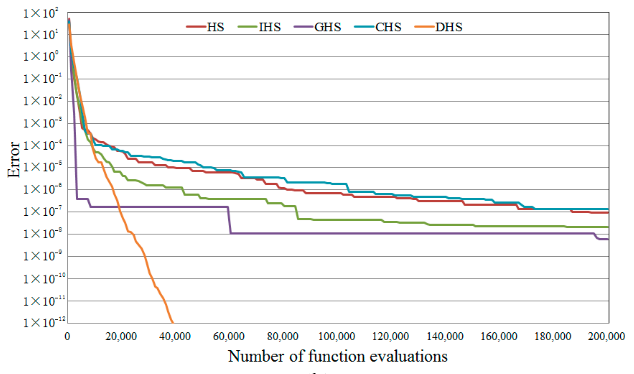
Table 4. *Cont.*

Function	Precision	HS	IHS	GHS	CHS	DHS
Rosenbrock function	$<1 \times 10^{-1}$	6%	9%	49%	4%	34%
	$<1 \times 10^{-2}$	2%	0%	29%	0%	10%
	$<1 \times 10^{-3}$	0%	0%	6%	0%	1%
	$<1 \times 10^{-4}$	0%	0%	3%	0%	0%
	$<1 \times 10^{-5}$	0%	0%	1%	0%	0%
	$<1 \times 10^{-6}$	0%	0%	0%	0%	0%
	$<1 \times 10^{-7}$	0%	0%	0%	0%	0%
Sphere function	$<1 \times 10^{-1}$	100%	100%	100%	100%	100%
	$<1 \times 10^{-2}$	100%	100%	100%	100%	100%
	$<1 \times 10^{-3}$	100%	100%	100%	100%	100%
	$<1 \times 10^{-4}$	100%	100%	100%	100%	100%
	$<1 \times 10^{-5}$	100%	100%	100%	100%	100%
	$<1 \times 10^{-6}$	77%	98%	89%	59%	100%
	$<1 \times 10^{-7}$	0%	0%	49%	0%	100%
Schwefel 2.22 function	$<1 \times 10^{-1}$	100%	100%	100%	100%	100%
	$<1 \times 10^{-2}$	100%	100%	98%	100%	100%
	$<1 \times 10^{-3}$	0%	0%	23%	0%	100%
	$<1 \times 10^{-4}$	0%	0%	4%	0%	100%
	$<1 \times 10^{-5}$	0%	0%	1%	0%	100%
	$<1 \times 10^{-6}$	0%	0%	0%	0%	100%
	$<1 \times 10^{-7}$	0%	0%	0%	0%	100%

We choose one typical unimodal function and one multimodal function to test the convergence speed of the algorithms. Figure 2 shows the convergence of the DHS algorithm compared to the variants of HS. The results clearly show that the DHS algorithm converges faster than other variants of HS.



(**a**)



(**b**)

Figure 2. Convergence characteristics: (**a**) Ackley function; (**b**) Sphere function.

4.1.2. Sensitivity Analysis of Parameters

In this section, the effect of each parameter in the search process of the DHS algorithm will be discussed.

Similar with the basic HS, the DHS algorithm has parameters that include *hms*, *hmcr*, *par*, and *fw*. The default parameters of DHS are *hms* = 50, *hmcr* = 0.9, *par* = 0.3, *fw* = 0.005, *sc* = 20. The maximum number of evaluations is set to 100,000. Then we change each parameter and test via the benchmark functions. Each scene runs 50 times. Tables 5–9 show the results of the optimization of the benchmark functions.

Table 5. The effect of *sc* on the means and standard deviations.

Function		***sc* = 20**	***sc* = 40**	***sc* = 60**	***sc* = 80**	***sc* = 100**
Ackley	Mean	9.12×10^{-13}	2.81×10^{-11}	3.87×10^{-10}	3.01×10^{-9}	1.33×10^{-8}
	Stdv	3.38×10^{-13}	1.15×10^{-11}	1.48×10^{-10}	1.18×10^{-9}	4.91×10^{-9}
Griewank	Mean	4.65×10^{-6}	2.63×10^{-5}	7.38×10^{-4}	3.13×10^{-3}	3.80×10^{-3}
	Stdv	2.91×10^{-5}	1.84×10^{-4}	3.86×10^{-3}	7.26×10^{-3}	9.49×10^{-3}
Rastrigin	Mean	5.97×10^{-15}	0	0	0	2.84×10^{-16}
	Stdv	4.22×10^{-14}	0	0	0	2.01×10^{-15}
Rosenbrock	Mean	1.32	1.55	1.85	1.49	1.72
	Stdv	1.36	1.40	1.64	1.29	1.57
Sphere	Mean	7.18×10^{-27}	9.36×10^{-24}	1.56×10^{-21}	9.81×10^{-20}	1.92×10^{-18}
	Stdv	4.38×10^{-27}	8.31×10^{-24}	9.09×10^{-22}	6.95×10^{-20}	1.42×10^{-18}
Schwefel 2.22	Mean	4.38×10^{-12}	3.82×10^{-11}	2.53×10^{-10}	1.09×10^{-9}	3.66×10^{-9}
	Stdv	1.99×10^{-12}	1.76×10^{-11}	9.65×10^{-11}	4.93×10^{-10}	1.64×10^{-9}

Table 6. The effect of *hms* on means and standard deviations.

Function		***hms* = 10**	***hms* = 30**	***hms* = 50**	***hms* = 70**	***hms* = 90**
Ackley	Mean	1.71×10^{-11}	3.11×10^{-15}	3.76×10^{-13}	1.58×10^{-9}	1.68×10^{-7}
	Stdv	3.45×10^{-13}	3.11×10^{-15}	1.66×10^{-13}	6.43×10^{-10}	8.20×10^{-8}
Griewank	Mean	6.58×10^{-2}	2.47×10^{-4}	9.37×10^{-6}	4.32×10^{-7}	2.64×10^{-5}
	Stdv	7.93×10^{-2}	1.74×10^{-3}	6.62×10^{-5}	7.10×10^{-7}	1.92×10^{-5}
Rastrigin	Mean	0	0	0	2.72×10^{-6}	2.46×10^{-6}
	Stdv	0	0	0	1.92×10^{-5}	1.74×10^{-5}
Rosenbrock	Mean	1.94	1.38	8.70×10^{-1}	7.11×10^{-1}	1.09
	Stdv	1.51	1.35	1.14	6.14×10^{-1}	6.05×10^{-1}
Sphere	Mean	1.16×10^{-89}	6.20×10^{-43}	1.97×10^{-28}	1.96×10^{-21}	2.62×10^{-17}
	Stdv	2.96×10^{-89}	8.53×10^{-43}	1.93×10^{-28}	1.36×10^{-21}	1.20×10^{-17}
Schwefel 2.22	Mean	1.57×10^{-41}	9.47×10^{-19}	2.92×10^{-12}	3.76×10^{-9}	2.27×10^{-7}
	Stdv	4.08×10^{-41}	6.63×10^{-19}	1.37×10^{-12}	1.78×10^{-9}	8.24×10^{-8}

The results in Table 5 show that the performance of the DHS algorithm will be better if *sc* is smaller. The value of *sc* determines the replacement frequency of the memories in the HM. Results show that *sc* values between 10 and 40 are suggested.

In Tables 6 and 7, we found that although values of *hms* and *hmcr* have impacts on the optimization results, but there is no obvious rule for the selection of *hms* and *hmcr*. *Hms* values between 30 and 70 are applicable for most cases, while *hmcr* values between 0.8 and 0.99 are suggested.

In Table 8, DHS performs better when *par* values are less than 0.3. In Table 9, we found that the algorithm is not sensitive to the parameter *fw*. A value of 0.005 is applicable for most cases.

Table 7. The effect of *hmcr* on the means and standard deviations.

Function		***hmcr* = 0.7**	***hmcr* = 0.8**	***hmcr* = 0.9**	***hmcr* = 0.99**
Ackley	Mean	3.95×10^{-10}	5.27×10^{-12}	1.73×10^{-13}	6.09×10^{-15}
	Stdv	5.56×10^{-10}	2.04×10^{-12}	6.69×10^{-14}	1.81×10^{-15}
Griewank	Mean	9.08×10^{-5}	7.97×10^{-7}	5.26×10^{-9}	2.46×10^{-4}
	Stdv	1.34×10^{-4}	1.13×10^{-6}	3.17×10^{-8}	1.74×10^{-3}
Rastrigin	Mean	7.39×10^{-14}	5.23×10^{-14}	2.58×10^{-8}	4.18×10^{-1}
	Stdv	3.73×10^{-13}	3.70×10^{-13}	1.82×10^{-7}	6.39×10^{-1}
Rosenbrock	Mean	7.88×10^{-1}	5.49×10^{-1}	1.26	1.72
	Stdv	4.52×10^{-1}	3.80×10^{-1}	1.43	1.54
Sphere	Mean	8.59×10^{-22}	1.09×10^{-25}	2.17×10^{-28}	7.11×10^{-32}
	Stdv	5.18×10^{-21}	8.31×10^{-26}	1.65×10^{-28}	6.02×10^{-32}
Schwefel 2.22	Mean	3.53×10^{-8}	5.96×10^{-10}	3.10×10^{-12}	1.43×10^{-14}
	Stdv	1.88×10^{-8}	4.04×10^{-10}	2.01×10^{-12}	6.77×10^{-15}

Table 8. The effect of *par* on the means and standard deviations.

Function		***par* = 0.1**	***par* = 0.3**	***par* = 0.5**	***par* = 0.7**	***par* = 0.9**
Ackley	Mean	3.04×10^{-15}	1.54×10^{-13}	5.74×10^{-10}	5.64×10^{-9}	3.02×10^{-8}
	Stdv	5.02×10^{-16}	6.46×10^{-14}	1.21×10^{-9}	4.21×10^{-9}	4.19×10^{-8}
Griewank	Mean	1.03×10^{-3}	2.23×10^{-9}	3.36×10^{-6}	5.70×10^{-5}	1.11×10^{-4}
	Stdv	3.37×10^{-3}	1.35×10^{-8}	1.06×10^{-5}	1.83×10^{-4}	2.98×10^{-4}
Rastrigin	Mean	0	1.64×10^{-7}	1.88×10^{-5}	8.43×10^{-2}	4.59×10^{-1}
	Stdv	0	1.16×10^{-6}	9.86×10^{-5}	2.67×10^{-1}	6.11×10^{-1}
Rosenbrock	Mean	3.36×10^{-1}	1.05	1.14	1.89	2.18
	Stdv	3.88×10^{-1}	1.17	1.35	1.45	1.54
Sphere	Mean	1.03×10^{-40}	1.82×10^{-28}	1.92×10^{-21}	2.81×10^{-19}	7.43×10^{-19}
	Stdv	1.08×10^{-40}	1.59×10^{-28}	4.61×10^{-21}	3.63×10^{-19}	2.10×10^{-18}
Schwefel 2.22	Mean	7.58×10^{-21}	2.85×10^{-12}	2.21×10^{-8}	6.46×10^{-8}	7.54×10^{-8}
	Stdv	5.31×10^{-21}	1.62×10^{-12}	1.07×10^{-8}	2.73×10^{-8}	3.71×10^{-8}

Table 9. The effect of *fw* on the means and standard deviations.

Function		***fw* = 0.001**	***fw* = 0.004**	***fw* = 0.007**	***fw* = 0.01**
Ackley	Mean	3.11×10^{-15}	2.97×10^{-15}	3.04×10^{-15}	2.82×10^{-15}
	Stdv	0	7.03×10^{-16}	5.02×10^{-16}	9.74×10^{-16}
Griewank	Mean	2.24×10^{-3}	2.41×10^{-3}	1.24×10^{-3}	2.95×10^{-3}
	Stdv	4.77×10^{-3}	5.48×10^{-3}	3.73×10^{-3}	5.86×10^{-3}
Rastrigin	Mean	1.90×10^{-5}	0	0	0
	Stdv	1.34×10^{-4}	0	0	0
Rosenbrock	Mean	4.10×10^{-1}	3.37×10^{-1}	4.42×10^{-1}	3.03×10^{-1}
	Stdv	6.34×10^{-1}	5.74×10^{-1}	8.63×10^{-1}	4.94×10^{-1}
Sphere	Mean	8.78×10^{-58}	2.91×10^{-57}	2.44×10^{-56}	1.23×10^{-55}
	Stdv	1.73×10^{-57}	4.10×10^{-57}	3.88×10^{-56}	2.55×10^{-55}
Schwefel 2.22	Mean	7.58×10^{-33}	2.49×10^{-32}	7.39×10^{-32}	2.05×10^{-31}
	Stdv	9.03×10^{-33}	1.94×10^{-32}	6.15×10^{-32}	1.92×10^{-31}

4.2. Numerical Experiments of DHS-KFC

To test the effectiveness of the DHS-KFC method, University of California Irvine (UCI) machine learning repositories: wine dataset and iris dataset are used here. The iris dataset consists of 50 samples from each of three species of iris. The wine dataset is the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The method is compared with the k-means and fuzzy cluster method. Classification results are shown in Table 10. The classification rates of wine and iris were higher for DHS-KFC than kmeans and fuzzy cluster.

Table 10. Results of the clustering method.

		Dataset	**k-Means**	**Fuzzy Cluster**	**DHS-KFC**
Wine data set	Class 1	59	61	63	59
	Class 2	71	63	63	70
	Class 3	48	54	52	49
	Error rates (%)		4.49%	4.49%	0.56%
Iris data set	Class 1	50	50	50	50
	Class 2	50	61	60	58
	Class 3	50	39	40	42
	Error rates (%)		11.33%	12.00%	10.67%

4.3. Case Study

East Lake is the largest scenery-related tourist attraction in Wuhan China, located on the south bank of the Yangtze River. The East Lake Network covers an area of 436 km^2, consisting of East Lake, Sha Lake, Yangchun Lake, Yanxi Lake, Yandong Lake, and Bei Lake. A map of the East Lake is shown in Figure 3. In recent years, climate change and human activities have influenced the lakes significantly. Increasing sewage has led to serious eutrophication. Most of the sub-lakes in the East Lake Network are highly polluted. Chemical oxygen demand (COD), total nitrogen (TN) and total phosphorus (TP) of the lakes are shown in Table 11.

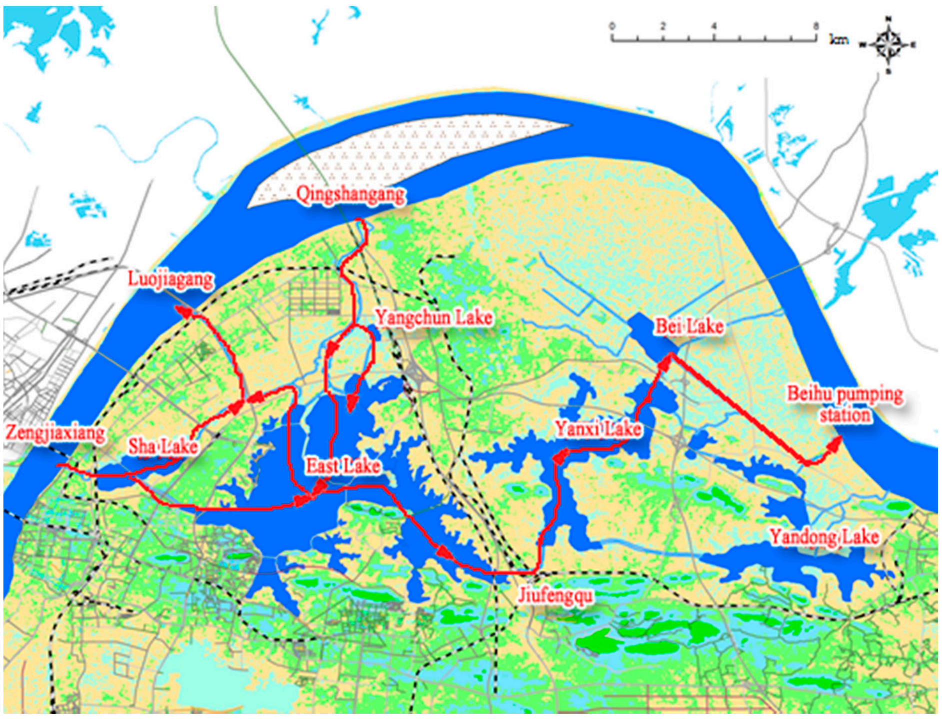


Figure 3. Map of the East Lake Network.

Table 11. Initial water qualities of the six lakes.

Lake	COD (mg/L)	TN (mg/L)	TP (mg/L)
East Lake	24	2.32	0.196
Sha Lake	50	6.11	0.225
Yangchun Lake	26	1.14	0.085
Yanxi Lake	34	3.82	0.2
Yandong Lake	10	0.7	0.05
Bei Lake	32	3.81	0.122

In recent years, the government has invested 15.8 billion yuan (RMB) to build the Ecological Water Network Project. The Water Network Connection Project is one of the subprojects; this project tries to transfer water from the Yangtze River through the diversion channels to improve the water quality in the lakes. In this project, Water diversion scheduling (WDS) is a new scheduling method combining with hydrodynamics. Previous works, including a hydrodynamic simulation model and water quality model, have already been done. Using these models, water quality of the lakes at the terminal time can be simulated. The purpose of WDS is to find a suitable scheme which has a better water quality result and lower economic cost. This is a multi-objective problem but, unfortunately, multi-objective evolutionary algorithms (MOEA) cannot be adopted because the water quality simulation is time-consuming. After a variety of simulations have been done, several feasible schemes and their simulation results had been made. However, the difference among the results is small. Thus, we use cluster analysis to summarize these schemes.

The water diversion scheduling in the East Lake Network is a multi-objective problem. To reduce the construction cost, the existing diversion channels include Zengjiaxiang, Qingshangang, Jiufengqu, Luojiagang, and Beihu pumping stations are considered. Water is brought from the Yangtze River to the East Lake Network through the Zengjiaxiang and Qingshangang channels, while water in the lakes is drained out through Luojiagang and Beihu pumping stations (Figure 4).

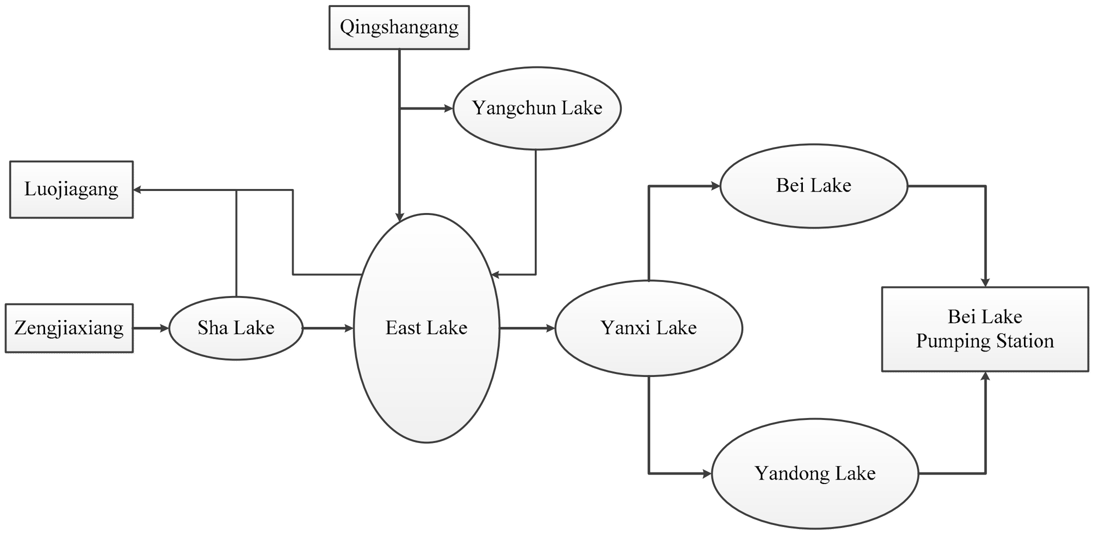


Figure 4. Diversion channels of the East Lake Network.

The objective functions are shown as:

$$\begin{cases} \mathbf{I} = f_1(\mathbf{q}_z, \mathbf{q}_q, \mathbf{q}_l) \\ C = f_2(\mathbf{q}_z, \mathbf{q}_q, \mathbf{q}_l) \\ Q = f_3(\mathbf{q}_z, \mathbf{q}_q, \mathbf{q}_l) \end{cases} \tag{17}$$

where **I** is the water quality index vector including TP, TN, and COD information, which is obtained by the water quality model. C is the total amount of water, Q is the economic cost, $\mathbf{q}_z$ is the inflow vector of the Zengjiaxiang channel, $\mathbf{q}_q$ is the inflow vector of the Qinshanggang channel, and $\mathbf{q}_l$ is the outflow vector of the Luojiagang channel. Then the outflow of Jiufengqu and Beihu pumping stations follows:

$$\mathbf{q}_j = \mathbf{q}_b = \mathbf{q}_z + \mathbf{q}_q - \mathbf{q}_l \tag{18}$$

The initial water qualities of the six lakes are shown in Table 11. The maximum inflow of the Qinshangang channel is 30 m^3/s, the maximum inflow of the Zengjiaxiang channel is 10 m^3/s, the total diversion time is 30 days. The water quality of Yandong Lake has already reached the standard, so it is not considered in this paper. The problem of water diversion scheduling is to design the inflow of Zengjiaxiang, Qinshangang, Luojiaxiang every day, which improves the water quality as much as possible with minimum cost.

Since the water quality simulation is time-consuming and MOEA cannot be adapted to this problem, after some pretreatment, we have identified a number of feasible schemes. These feasible schemes are shown in Table 12. Then we applied these schemes to the water quality model. The water quality simulation results of the schemes are shown in Table 13. Since the water quality model may have a certain amount of error, we decided to determine a set of good schemes instead of one scheme.

Table 12. Feasible schemes.

No.	1–5 Days			6–10 Days			11–15 Days			16–20 Days			21–25 Days			26–30 Days		
	$\mathbf{q}_z$	$\mathbf{q}_q$	$\mathbf{q}_l$	$\mathbf{q}_z$	$\mathbf{q}_q$	$\mathbf{q}_l$	$\mathbf{q}_z$	$\mathbf{q}_q$	$\mathbf{q}_l$	$\mathbf{q}_z$	$\mathbf{q}_q$	$\mathbf{q}_l$	$\mathbf{q}_z$	$\mathbf{q}_q$	$\mathbf{q}_l$	$\mathbf{q}_z$	$\mathbf{q}_q$	$\mathbf{q}_l$
1	10	0	10	10	30	40	10	30	20	10	30	20	10	30	20	10	30	20
2	10	0	10	10	30	40	5	22.5	13.8	5	22.5	13.8	5	22.5	13.8			
3	10	0	10	10	30	40	10	30	40	10	30	20	10	30	20	10	30	20
4	10	0	10	10	30	40	10	30	40	7.5	30	18.8	7.5	30	18.8	7.5	30	18.8
5	10	0	10	10	30	40	10	30	40	5	27.5	16.2	5	27.5	16.2	5	27.5	16.2
6	10	0	10	10	30	40	10	30	40	5	22.5	13.8	5	22.5	13.8	5	22.5	13.8
7	10	0	10	10	30	40	10	30	40	10	30	40	7.5	20	13.8			
8	10	0	10	10	25	35	5	25	15	5	25	15	5	25	15	5	25	15
9	10	0	10	10	25	35	10	20	15	10	20	15	10	20	15	10	20	15
10	10	0	10	10	25	35	5	20	12.5	5	20	12.5	5	20	12.5			
11	10	0	10	10	25	35	10	25	35	5	25	15	5	25	15			
12	10	0	10	10	25	35	10	25	35	10	25	35	7.5	27.5	17.5	7.5	27.5	17.5
13	10	0	10	10	25	35	10	25	35	10	25	35	5	25	15	5	25	15
14	10	0	10	5	30	35	10	22.5	16.2	10	22.5	16.2	10	22.5	16.2			
15	10	0	10	5	30	35	5	30	35	7.5	27.5	17.5	7.5	27.5	17.5			
16	10	0	10	5	30	35	5	30	35	7.5	27.5	17.5	7.5	27.5	17.5	7.5	27.5	17.5
17	10	0	10	5	30	35	5	30	35	5	30	17.5	5	25	15			
18	10	0	10	5	30	35	5	30	35	5	30	35	5	22.5	13.8	5	22.5	13.8

Table 13. Results of the schemes shown in Table 12.

No.	Sha Lake			Yangchun Lake			East Lake			Yandong Lake			Bei Lake			Water	Cost
	TP	TN	COD	TP	TN	COD	TP	TN	COD	TP	TN	COD	TP	TN	COD		
1	0.108	2.386	17.569	0.066	0.997	10.602	0.16	2.136	19.43	0.198	3.294	30.023	0.165	3.907	32.88	9072	226.8
2	0.126	2.942	22.051	0.066	0.998	10.773	0.172	2.23	20.895	0.2	3.366	30.644	0.161	3.912	32.93	5724	143.1
3	0.108	2.386	17.569	0.066	0.997	10.602	0.161	2.14	19.438	0.2	3.366	30.566	0.161	3.917	32.943	9072	226.8
4	0.115	2.593	19.413	0.066	0.997	10.602	0.162	2.161	19.668	0.2	3.365	30.566	0.161	3.917	32.943	8748	218.7
5	0.12	2.751	20.453	0.066	0.997	10.602	0.165	2.177	19.95	0.2	3.367	30.581	0.161	3.917	32.943	8100	202.5
6	0.12	2.751	20.453	0.066	0.997	10.602	0.168	2.209	20.321	0.2	3.369	30.607	0.161	3.917	32.943	7452	186.3
7	0.116	2.624	19.634	0.066	0.998	10.773	0.168	2.211	20.482	0.202	3.627	32.489	0.144	3.877	32.597	6804	170.1
8	0.123	2.865	21.321	0.066	0.997	10.602	0.17	2.222	20.575	0.199	3.301	30.12	0.165	3.907	32.881	7128	178.2
9	0.108	2.386	17.569	0.066	0.997	10.602	0.168	2.221	20.431	0.2	3.306	30.184	0.165	3.907	32.882	7128	178.2
10	0.126	2.942	22.051	0.066	0.998	10.773	0.175	2.26	21.26	0.2	3.369	30.684	0.161	3.912	32.93	5184	129.6
11	0.122	2.827	21.174	0.066	0.998	10.773	0.17	2.217	20.659	0.201	3.469	31.382	0.155	3.906	32.87	6048	151.2
12	0.113	2.536	18.893	0.066	0.997	10.602	0.165	2.185	19.988	0.201	3.473	31.379	0.156	3.91	32.882	7992	199.8
13	0.117	2.668	19.823	0.066	0.997	10.602	0.167	2.2	20.24	0.201	3.474	31.383	0.156	3.91	32.882	7560	189
14	0.115	2.62	19.627	0.066	0.998	10.773	0.169	2.23	20.719	0.2	3.366	30.648	0.161	3.912	32.93	6156	153.9
15	0.126	2.932	22.261	0.066	0.998	10.773	0.169	2.208	20.578	0.201	3.468	31.368	0.155	3.906	32.87	6480	162
16	0.122	2.818	21.328	0.066	0.997	10.602	0.167	2.203	20.236	0.2	3.367	30.583	0.161	3.917	32.943	7992	199.8
17	0.134	3.204	24.488	0.066	0.998	10.773	0.172	2.207	20.752	0.202	3.627	32.488	0.144	3.877	32.597	6264	156.6
18	0.133	3.137	23.92	0.066	0.997	10.602	0.171	2.22	20.632	0.201	3.473	31.375	0.156	3.91	32.882	7344	183.6

In this paper, the goal is to use the DHS-KFC method to determine the good schemes. According to the requirements, we need to divide the schemes into five categories, including excellent, good, average, fair, and poor. Using the kernel cluster method explained in Section 3, we input the scheme results shown in Table 13 as the samples for the clustering, including three water quality parameters (COD, TP, TN) of five lakes, water diversion quantity, and economic cost. The weight vector of water quality, water diversion quantity, and economic cost is defined in Equation (23) according to the advice given by experts.

In this case study, the parameter $m = 1.7$, and the minimum objective function value is 0.225269. The cluster results are shown in Table 14. The cluster center **v**, membership matrix **u**, and weight vector **w** are shown as:

$$\begin{aligned}\mathbf{w} = [&0.040.040.040.040.040.040.040.040.04\\ &0.040.040.040.040.040.040.20.2]\end{aligned} \tag{19}$$

$$\begin{aligned}\mathbf{v} = &\left[\begin{matrix} 0.077 & 0.102 & 0.087 & 1 & 0.084 & 0.001 & 0.087 & 0.120 & 0.030 \\ 0.459 & 0.461 & 0.417 & 1 & 0 & 0 & 0.457 & 0.459 & 0.388 \\ 0.601 & 0.600 & 0.660 & 1 & 0.979 & 1 & 0.784 & 0.741 & 0.764 \\ 0.761 & 0.554 & 0.769 & 1 & 0.915 & 0.916 & 0.775 & 0.563 & 0.696 \\ 0.223 & 0.243 & 0.230 & 1 & 0.032 & 0.067 & 0.750 & 0.669 & 0.467 \end{matrix}\right.\\ &\left.\begin{matrix} 0.364 & 0.198 & 0.118 & 0.981 & 0.846 & 0.995 & 1 & 0.946 \\ 0.533 & 0.307 & 0.322 & 0.772 & 0.883 & 0.878 & 0.648 & 0.645 \\ 0.626 & 0.352 & 0.391 & 0.752 & 0.808 & 0.953 & 0.185 & 0.191 \\ 0.937 & 0.914 & 1 & 0 & 0.107 & 0 & 0.313 & 0.286 \\ 0.645 & 0.098 & 0 & 0.88 & 0.773 & 1 & 0.48 & 0.479 \end{matrix}\right]\end{aligned} \tag{20}$$

$$\begin{aligned}\mathbf{u} = &\left[\begin{matrix} 0.935 & 0.001 & 0.973 & 0.858 & 0.064 & 0.010 & 0.021 & 0.034 & 0.028 \\ 0.035 & 0.003 & 0.015 & 0.102 & 0.869 & 0.842 & 0.044 & 0.377 & 0.125 \\ 0.005 & 0.982 & 0.002 & 0.005 & 0.006 & 0.006 & 0.077 & 0.044 & 0.018 \\ 0.004 & 0.006 & 0.001 & 0.004 & 0.004 & 0.003 & 0.813 & 0.018 & 0.008 \\ 0.019 & 0.006 & 0.007 & 0.028 & 0.054 & 0.137 & 0.042 & 0.525 & 0.819 \end{matrix}\right.\\ &\left.\begin{matrix} 0.010 & 0.004 & 0.127 & 0.029 & 0.015 & 0.013 & 0.021 & 0.005 & 0.059 \\ 0.026 & 0.014 & 0.697 & 0.815 & 0.042 & 0.044 & 0.923 & 0.012 & 0.525 \\ 0.873 & 0.915 & 0.019 & 0.015 & 0.824 & 0.780 & 0.005 & 0.035 & 0.106 \\ 0.048 & 0.045 & 0.017 & 0.012 & 0.040 & 0.115 & 0.003 & 0.935 & 0.084 \\ 0.040 & 0.019 & 0.137 & 0.127 & 0.077 & 0.047 & 0.046 & 0.011 & 0.223 \end{matrix}\right]\end{aligned} \tag{21}$$

Using the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) model [22,23], the similarity values of the cluster centers are shown as:

$$\mathbf{CC} = \begin{bmatrix} 0.950 & 0.592 & 0.13 & 0.052 & 0.816 \end{bmatrix} \tag{22}$$

The schemes in cluster I are considered better than the others.

Table 14. Results of the proposed method.

No.	Classification	No.	Classification	No.	Classification
1	I	7	IV	13	II
2	III	8	V	14	III
3	I	9	V	15	III
4	I	10	III	16	II
5	II	11	III	17	IV
6	II	12	II	18	II

5. Conclusions

The main contributions of this work are (i) a novel modification of the harmony search algorithm is proposed to deal with the optimization problem; (ii) a kernel fuzzy clustering algorithm with the harmony search algorithm is proposed; and (iii) the methodology is adopted for a water diversion scheduling assessment in East Lake.

In order to show the performance of the proposed differential harmony search algorithm, several tests have been conducted to compare with other methods. Results show that the modification can effectively improve the convergence of the harmony search algorithm. Finally, the combination of KFC and DHS is applied to the water diversion scheduling assessment in East Lake. It is efficient in clustering multi-dimensional data; the result shows that the methodology is reasonable and reliable. However, simulation results show that DHS still has the drawback of local convergence when dealing with some functions. Our work in the future will be to overcome this shortcoming.

Acknowledgments: This work is supported by The State Key Program of National Natural Science of China (Grant No. 51239004). We also acknowledge the entire development team, without whose help this research could not have been achieved.

Author Contributions: Yu Feng wrote the DHS-KFC algorithm. Muhammad Tayyab performed the experiments. Final checks were done by Jianzhong Zhou. All authors have read and approved the final manuscript.

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

References

1. Bianchi, L.; Dorigo, M.; Gambardella, L.M.; Gutjahr, W.J. A survey on metaheuristics for stochastic combinatorial optimization. *Nat. Comput.* **2009**, *8*, 239–287. [CrossRef]
2. Blum, C.; Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *J. ACM Comput. Surv.* **2001**, *35*, 268–308. [CrossRef]
3. Almasi, M.H.; Sadollah, A.; Mounes, S.M.; Karim, M.R. Optimization of a transit services model with a feeder bus and rail system using metaheuristic algorithms. *J. Comput. Civ. Eng.* **2015**, *29*, 1–4. [CrossRef]
4. Yazdi, J.; Sadollah, A.; Lee, E.H.; Yoo, D.; Kim, J.H. Application of multi-objective evolutionary algorithms for the rehabilitation of storm sewer pipe networks. *J. Flood Risk Manag.* **2015**. [CrossRef]
5. Yoo, D.G. Improved mine blast algorithm for optimal cost design of water distribution systems. *Eng. Optim.* **2014**, *47*, 1602–1618.
6. Eskandar, H.; Sadollah, A.; Bahreininejad, A. Weight optimization of truss structures using water cycle algorithm. *Int. J. Optim. Civ. Eng.* **2013**, *3*, 115–129.
7. Sadollah, A.; Eskandar, H.; Yoo, D.G.; Kim, J.H. Approximate solving of nonlinear ordinary differential equations using least square weight function and metaheuristic algorithms. *Eng. Appl. Artif. Intell.* **2015**, *40*, 117–132. [CrossRef]
8. Glover, F.W.; Kochenberger, G.A. *Handbook of Metaheuristics*; Springer: New York, NY, USA, 2003; pp. 293–377.
9. Geem, Z.W.; Kim, J.H.; Loganathan, G. A new heuristic optimization algorithm: Harmony search. *Simulation* **2001**, *76*, 60–68. [CrossRef]
10. Wang, Y.; Liu, Y.; Feng, L.; Zhu, X. Novel feature selection method based on harmony search for email classification. *Knowl. Based Syst.* **2014**, *73*, 311–323. [CrossRef]
11. Zammori, F.; Braglia, M.; Castellano, D. Harmony search algorithm for single-machine scheduling problem with planned maintenance. *Comput. Ind. Eng.* **2014**, *76*, 333–346. [CrossRef]
12. Mahdavi, M.; Fesanghary, M.; Damangir, E. An improved harmony search algorithm for solving optimization problems. *Appl. Math. Comput.* **2007**, *188*, 1567–1579. [CrossRef]
13. Omran, M.G.H.; Mahdavi, M. Global-best harmony search. *Appl. Math. Comput.* **2008**, *198*, 643–656. [CrossRef]
14. Alatas, B. Chaotic harmony search algorithms. *Appl. Math. Comput.* **2010**, *216*, 2687–2699. [CrossRef]
15. Jain, A.K.; Dubes, R.C. *Algorithms for Clustering Data*; Prentice hall Englewood Cliffs: Upper Saddle River, NJ, USA, 1988.
16. Girolami, M. Mercer kernel-based clustering in feature space. *IEEE Trans. Neural Netw.* **2002**, *13*, 780–784. [CrossRef] [PubMed]

17. Shen, H.; Yang, J.; Wang, S.; Liu, X. Attribute weighted mercer kernel based fuzzy clustering algorithm for general non-spherical datasets. *Soft Comput.* **2006**, *10*, 1061–1073. [CrossRef]
18. Yang, M.-S.; Tsai, H.-S. A gaussian kernel-based fuzzy c-means algorithm with a spatial bias correction. *Pattern Recognit. Lett.* **2008**, *29*, 1713–1725. [CrossRef]
19. Ferreira, M.R.P.; de Carvalho, F.A.T. Kernel fuzzy c-means with automatic variable weighting. *Fuzzy Sets Syst.* **2014**, *237*, 1–46. [CrossRef]
20. Graves, D.; Pedrycz, W. Kernel-based fuzzy clustering and fuzzy clustering: A comparative experimental study. *Fuzzy Sets Syst.* **2010**, *161*, 522–543. [CrossRef]
21. Xing, H.-J.; Ha, M.-H. Further improvements in feature-weighted fuzzy c-means. *Inf. Sci.* **2014**, *267*, 1–15. [CrossRef]
22. Chen, Z.P.; Yang, W. An magdm based on constrained fahp and ftopsis and its application to supplier selection. *Math. Comput. Model.* **2011**, *54*, 2802–2815. [CrossRef]
23. Singh, R.K.; Benyoucef, L. A fuzzy topsis based approach for e-sourcing. *Eng. Appl. Artif. Intell.* **2011**, *24*, 437–448. [CrossRef]