Chapter IV

Application 1: Modification to Data Clustering Algorithms
CHAPTER 4

APPLICATION 1: MODIFICATION TO DATA CLUSTERING ALGORITHMS

4.1 INTRODUCTION

Clustering means the act of partitioning an unlabelled data set into groups of similar objects. Each group, called a “cluster” consists of objects that are similar between themselves and dissimilar to objects of other groups. In the past few decades, cluster analysis has played a central role in a variety of fields such as engineering (e.g., machine learning, artificial intelligence, pattern recognition, mechanical engineering, and electrical engineering), computer sciences (e.g., web mining, spatial database analysis, textual document collection, and image segmentation), life and medical sciences (e.g., genetics, biology, microbiology, palaeontology, psychiatry, and pathology), earth sciences (e.g., geography, geology, and remote sensing), social sciences (e.g., sociology, psychology, archaeology, and education), and economics (e.g., marketing and business) [15, 28, 31, 40, 44].

4.1.1 Data Clustering Algorithms

Data clustering algorithms can be hierarchical or partitional [28]. Within each of the types, there exists a wealth of subtypes and different algorithms for finding the clusters. In hierarchical clustering, the output is a tree showing a sequence of clustering, with each cluster being a partition of the data set [28]. Partitional clustering algorithms, on the other hand, attempt to decompose the data set directly into a set of disjoint clusters. They try to optimize certain criteria such as square-error function, and the criterion function may emphasize the local structure of the data, such as by assigning clusters to peaks in the probability density function, or the global structure. Typically, the global criteria involve minimizing some measure of dissimilarity in the samples within each cluster while maximizing the dissimilarity of different clusters [46]. The work described in this thesis focuses on the partitional clustering algorithms.
4.2 K-MEANS CLUSTERING ALGORITHM

The goal of data clustering is grouping data into a number of clusters and K-Means algorithm is the most popular clustering algorithm. In this section, the K-Means algorithm has been briefly described. Let \( X = \{x_1, x_2, \ldots, x_N\} \) be a set of \( N \) data and let each data vector be a \( p \)-dimensional vector. Let \( C = \{c_1, c_2, \ldots, c_K\} \) be a set of \( K \) clusters and \( K \) denotes the number of cluster centroids which is provided by the user.

The aim of the K-Means algorithm is to get a partition of \( X \), which can be called \( S \), whose subsets can be called \( S = \{S_1, S_2, \ldots, S_K\} \). In K-Means algorithm, \( K \) cluster centroid vectors are initialized randomly and then each data vector to the class is assigned with the closest centroid vector [46]. In this work, Euclidian metric has been used as a distance metric and is shown in the following equation (1):

\[
D(x_i, c_j) = \sqrt{\sum_{k=1}^{p} (x_{ik} - c_{jk})^2}
\]  

(1)

After all data are being grouped, the cluster centroid vectors are recalculated using the following equation (2):

\[
C_j = \frac{1}{n_j} \sum_{S \in S_j} x_i
\]

(2)

where \( n_j \) is the number of data vectors in subset \( S_j \). After the above process, the data to the new cluster centroids are reassigned and the process is repeated until a criterion is satisfied. To measure the goodness of the partition, a measure must be defined. A popular performance function for measuring goodness of the partition is the total Within-Cluster Variance or the total Mean-Square quantization Error (MSE), which is defined in the following equation (3):

\[
MSE(X, C) = \sum_{i=1}^{N} \min \{\|X_i - C_l\|^2 \mid l = 1, \ldots, K\}
\]

(3)

The major steps in K-Means algorithm are shown in Fig. 4.1.

Step 1: Decide on a value for \( K \), the number of clusters

Step 2: Initialize the \( K \) cluster centers (randomly, if necessary)

Step 3: Decide the class memberships of the \( N \) objects by assigning them to the nearest cluster centers
Step 4: Re-estimate the K cluster centers, by assuming the memberships found above are correct.

Step 5: If none of the N objects changed membership in the last iteration, exit and return the solution. Otherwise go to Step 3.

Fig. 4.1 K-Means algorithm

4.2.1 Drawbacks of K-Means

The algorithm is efficient at clustering large data sets because its computational complexity only grows linearly with the number of data points. Although K-Means algorithm is simple, straightforward and easy to be implemented and works fast in most situations, it suffers from two major drawbacks which make it inappropriate for many applications. One is sensitivity to initialization and the other is convergence to local optima. This research work has proposed two novel algorithms to overcome these drawbacks as well as to solve the clustering problem.

4.3 METAHEURISTIC OPTIMIZATION METHODS FOR DATA CLUSTERING – THE NEED

The key idea in metaheuristic is to create a population of candidate solutions to an optimization problem, which is iteratively refined by alteration and selection of good solutions for the next iteration. Candidate solutions are selected according to a fitness function, which evaluates their quality with respect to the optimization problem.

An important advantage of these algorithms is their ability to cope with local optima by maintaining, recombining, and comparing several candidate solutions simultaneously. In contrast, local search heuristics, such as the simulated annealing algorithm [38], only refine a single candidate solution and are notoriously weak in coping with local optima. Deterministic local search, which is used in algorithms like the K-Means [46], always converges to the nearest local optimum from the starting position of the search. In this research work, the metaheuristic algorithms EBHS and EBGHS proposed in chapter 3 have been employed for clustering process.
4.4 NOVEL ALGORITHMS FOR DATA CLUSTERING

This section proposes two novel data clustering algorithms by adopting the EBHS and EBGHS algorithms given in Chapter 3. These algorithms in combination with K-Means clustering algorithm give rise to new algorithms namely HSBEEK and GHSBEEK.

4.4.1 The Framework

The Harmony memory has been initialised as shown in Fig. 4.3 with HMS number of random solutions where each solution is a centroid vector. Each centroid vector contains D centers where D is the dimension of the dataset considered.

The steps involved in the strategy used in the design of the proposed algorithms are as follows:

1) Pre-processing:

   In this step, normalization is applied to prevent feature values in greater numeric ranges from dominating those in smaller numeric ranges, as well as to prevent numerical inconveniences during calculation. Each feature value can generally be linearly scaled to the range \([0,1]\) using the following equation (4):

   \[
   x'_i = \left( \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \right)
   \]  

   (4)

   where \(x'_i\) denotes the scaled value: where \(x_i\) is the actual value of attribute i, \(\max(x_i)\) represents the maximum value of the feature i in the dataset, and \(\min(x_i)\) denotes the minimum value of the feature i in the dataset. If the instance lacks the values of some features (i.e., an instance with a missing value), then it was removed.

2) Solution Representation:

   The solution (Centroid Vector) is represented as shown in Fig. 4.2. If the data set involves D features, then D variables must be adopted. Each variable is in the range from 0 to 1. That is, the Harmony vectors are limited between a lower
bound and an upper bound. Execute K-Means algorithm and calculate fitness using equation 3 for the randomly selected solution vectors (centroids) and store them in the Harmony Memory (HM) as shown in Fig 4.3.

<table>
<thead>
<tr>
<th>$c_{i1}$</th>
<th>$c_{i2}$</th>
<th>$c_{i3}$</th>
<th>...</th>
<th>$c_{iD}$</th>
</tr>
</thead>
</table>

**Fig. 4.2 The Centroid Vector $C_i$ with D centers**

| 1 | Centroid 1 | $fit \,(X_1)$ |
| 2 | Centroid 2 | $fit \,(X_2)$ |
| ... | ... | ... |
| HMS | Centroid $_{HMS}$ | $fit \,(X_{HMS})$ |

**Fig. 4.3 The Harmony Memory in Clustering Process**

3) **Improvisation of a New Centroid:**
   (i) Select a vector solution randomly from HM and mutate it by the neighbouring centroids based on HMCR and Pitch adjust the mutated vector based on PAR

(ii) Create a New vector solution based on Random function value

Execute K-Means and calculate fitness for New solution vector

4) **Harmony memory Updation**

Replace the Worst solution vector with New Solution vector if the fitness value of new solution vector is better than Worst one.
5) **Termination criteria**

**Case 1:**

If \( \text{abs}(\text{MSE}(\text{Best Centroid})) < \text{minimum threshold value} \) is satisfied, then the process ends by returning the best harmony vector as the optimal solution. Otherwise, the next iteration will run.

**Case 2:**

If maximum number of iterations, NI is reached, then the process ends by returning the best harmony vector as the optimal solution. Otherwise, the next iteration will run.

**4.4.2 The Clustering Algorithm - HSBEEK**

The formation of HSBEEK algorithm is depicted as a two step process.

**Step 1:** In HS, the update of the HS memory highly depends on the past search experiences which slow down the search process. So, the exploitation phase of the HS method has been refined and the selection process used has been enhanced. The food source exploitation feature of the Artificial Bee Colony method [32] is employed to improve the fitness of the solution candidates in HM. The following pseudo code in Fig. 4.4 illustrates the ABC inspired HS algorithm (EBHS).

```
Begin

Define the Objective function \( f(x), x=(x_1,x_2, ...,x_d) \), where \( d \) is the dimension of the data set

Set the Parameters Harmony Memory Size (HMS), Harmony Memory Considering Rate (HMCR), Pitch Adjustment Rate (PAR), Number of Iterations (NI) and, the minimum threshold value(\( \varepsilon \))

Initialize the HM with a set of random solutions

(ii) Improvise a new Harmony:

\( \text{for} \ I = 1 \text{ to } D \text{ do (where } D \text{ represents Dimension) } \)

\( \text{if (rand >HMCR)}, \)

\( \text{begin} \)

Choose an existing harmony \( (X_a) \) randomly

Mutate it by its neighbouring pitch value within limits and name it as \( X_{\text{new}} \)

Evaluate the Objective function value of \( X_{\text{new}} \)

\( \text{if } X_{\text{new}} \text{ is better than } X_a, \text{ then replace } X_a \text{ with } X_{\text{new}} \text{ endif} \)

End
```
if (rand>PAR)
    Adjust the pitch randomly within limits
endif
end
else
    generate new harmony $X_{new}$ via randomization
endif
if the objective function value of $X_{new}$ is better than the worst harmony $X_w$
    replace $X_w$ with $X_{new}$
endif
endfor
if the termination criteria is met find the current best solution and exit
else proceed with the next iteration endif
end

Fig. 4.4 The Pseudo code of ABC inspired Harmony Search Algorithm (EBHS)

This new approach modifies the pitch adjustment step of HS and this modification allows this algorithm work efficiently and speeds up the convergence process.

Step 2: While the ABC inspired HS algorithm (EBHS) has been used as a global search strategy across the whole solution space, the K-Means algorithm has been used as a local strategy for improving solutions. EBHS together with K-Means algorithm forms a novel clustering algorithm HSBEEK and the following Pseudo code in Fig. 4.5 illustrates it.

(i) Initialize the Parameters Harmony Memory Size (HMS), Harmony Memory Considering Rate (HMCR), Pitch Adjustment Rate (PAR), Number of Iterations (NI) and the minimum threshold value ($\theta$)
(ii) Select initial centroids from the original data set
(iii) Initialize the Harmony Memory (HM) with initial centroids
(iv) for i = 1 to HMS
    begin
        MSE_centroid[i] = Call_K_Means (HM(centroid[i]))
    Next-for
(v) Improvise a new Harmony: //Finding the new centroid/
    for i = 1 to D do (where D represents Dimension)
        if (rand > HMCR)
begin
Randomly select a center of a centroid from HM, RCi
Mutate it by its neighbouring centers within limit, MCi
MSE_RCI = Call_K_Means(RCi)
MSE_MCI = Call_K_Means(MCi)
if MSE_MCI < MSE_RCI
    NC[i] = MCi
else
    NC[i] = RCi
endif
if (rand>PAR)
    Pitch adjust the NC[i] within limit
endif
end
else
    NC[i] = Randomly generated center element
endif
Next-for
MSE_NC = Call_K_Means(NC)
(vi) Update the harmony memory:
    MSE_WC = Call_K_means(WC)// WC is the worst centroid in HM
    if MSE_NC < MSE_WC then replace WC in HM with NC
(vii) Check the Stopping Criterion :
Case 1:
    Select the best Centroid in HM, BC
    MSE_BC = Call_K_Means(BC)
    if abs(MSE_BC < ε))
        return BC and exit
    else
        go to step(v)
    endif
Case 2:
    if maximum number of iterations NI is reached, the process ends by returning the best centroid as the optimal solution. Otherwise, the next iteration will run
endif
Fig. 4.5 The Pseudo code of the Clustering Algorithm (HSBEEK)
4.4.3 The Clustering Algorithm - GHSBEEK

In GHS, the update of the Harmony memory highly depends on the past search experiences. Unfortunately, this inherent shortcoming limits the search ability of the GHS method. The food source exploitation feature of the Artificial Bee Colony method is employed to improve the fitness of the solution candidates in the HM. While the ABC inspired GHS algorithm can be used as a global search strategy across the whole solution space, the K-Means algorithm has been used as a local strategy for improving solutions. The following Pseudo code shown in Fig. 4.6 illustrates the proposed algorithm GHSBEEK.

(i) \textit{Initialize the Parameters} Harmony Memory Size (HMS), Harmony Memory Considering Rate (HMCR), Pitch Adjustment Rate (PAR), Number of Iterations (NI) and, the minimum threshold value ($\mathcal{E}$)

(ii) Select initial centroids from the original data set

(iii) Initialize the Harmony Memory (HM) with initial centroids

(iv) for $i = 1$ to HMS

\hspace{0.5cm}begin

\hspace{1cm}MSE\textsubscript{centroid}[i] = Call\_K\_Means(HM(centroid[i]))

\hspace{0.5cm}Next-for

(v) Improvise a new Harmony: //Finding the new centroid/

\hspace{0.5cm}for $i = 1$ to D do (where D represents Dimension)

\hspace{1cm}if (rand $>$ HMCR)

\hspace{1.5cm}begin

\hspace{2cm}Randomly select a center of a centroid from HM, RC\textsubscript{i}

\hspace{2cm}Mutate it by its neighbouring centers within limit, MC\textsubscript{i}

\hspace{2cm}MSE\_RC = Call\_K\_Means(RC)

\hspace{2cm}MSE\_MC = Call\_K\_Means(MC)

\hspace{1.5cm}if MSE\_MC $<$ MSE\_RC

\hspace{2cm}NC\textsubscript{ij} = MC\textsubscript{i}

\hspace{1.5cm}else

\hspace{2cm}NC\textsubscript{ij} = RC\textsubscript{i}

\hspace{1cm}endif

\hspace{0.5cm}endif
if (rand>PAR)
    NC[i] = GBC    //where GBC is the global best centroid in the HM i.e the
    centroid with minimum MSE value //
endif

end

else
    NC[i] = Randomly generated center element
endif

Next-for

MSE_NC = Call_K_Means(NC)

(vi) Update the harmony memory:
    MSE_WC= Call_K_means(WC)// WC is the worst centroid in HM
    If MSE_NC < MSE_WC then replace WC in HM with NC

(vii) Check the Stopping Criterion : 
    Case 1:
        Select the best Centroid in HM, BC
        MSE_Bc = Call_K_Means(BC)
        if (MSE_Bc < ε)
            return BC and exit
        else
            go to step(v)
        endif
    
    Case 2:
        if maximum number of iterations NI is reached, the process ends by returning the best
        centroid as the optimal solution. Otherwise, the next iteration will run

endif

Fig. 4.6 The Pseudo code of the Clustering Algorithm (GHSBEEK)

4.5 DATA CLUSTERING – PERFORMANCE EVALUATION OF HSBEEK

This section discusses about the performance evaluation of the novel data clustering algorithm HSBEEK.

4.5.1 Experimental Setup

This algorithm has been implemented using MATLAB 7.0 and data sets were selected from the UCI machine learning repository [88].
The performance evaluation of the proposed HSBEEK approach for clustering on six different data sets selected from the UCI machine learning repository has been done and its results have been compared with the results of the K-Means, PSO, and HS clustering algorithms.

For HS algorithm, the following parameters were set to the values as recommended in [15]: the size of the harmony memory = 10, HMCR = 0.9, PAR = 0.3, BW = 0.01. The maximum improvisation number was set as 10000 for all test problems. For the proposed algorithm, the same parameter setting has been maintained. The standard PSO has been used. In this algorithm, the inertia weight $\omega$ has been varied from 0.9 to 0.7 linearly with the iterations and the acceleration factors $c_1$ and $c_2$ have been kept as 2.0 [5].

4.5.2 The Datasets applied

Motorcycle data (N = 133, D= 2, K = 4): the Motorcycle benchmark consists of a sequence of accelerometer readings through time following a simulated motorcycle crash during an experiment to determine the efficacy of crash helmets.

Iris data (N =150, D =4, K = 3): this data set is with 150 random samples of flowers from the iris species setosa, versicolor, and virginica. From each species there are 50 observations for sepal length, sepal width, petal length, and petal width measured in cm.

Wine data (N =178, D=13, K =3): There are 178 instances with 13 numeric attributes in wine data set. All attributes are continuous. There is no missing attribute value.

Wisconsin Breast Cancer (N = 683, D=9, K = 2): This data set consists of 683 objects characterized by nine features namely, clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. There are two categories in the data: malignant (444 objects) and benign (239 objects).

Contraceptive Method Choice (N=1473, D =10, K = 3): This data set is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The samples are
married women who were either not pregnant or do not know if they were at the time of interview. The problem is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socioeconomic characteristics.

Ripley’s glass ($N = 214$, $D=9$, $K = 6$): For this data set, data were sampled from six different types of glass: building windows float processed (70 objects), building windows non-float processed (76 objects), vehicle windows float processed (17 objects), containers (13 objects), tableware (9 objects), and headlamps (29 objects), each with nine features, which are refractive index, sodium, magnesium, aluminium, silicon, potassium, calcium, barium, and iron.

4.5.3 Discussion on Experiment Results

For every data set, each algorithm has been applied 30 times individually with random initial solutions. The efficiencies of the algorithms are compared according to their quality and speed of convergence.

4.5.3.1 Evaluation of Quality

For evaluating the quality of clustering results the metric intra-cluster distance has been used. Table 4.1 summarizes the intra-cluster distances, as defined in equation (3) of this chapter, obtained from all algorithms for the data sets considered. The average, best, and worst solution of fitness from 30 simulations, and standard deviation are presented in Table 4.1. Figures from 4.7 to 4.12 show the search progress of the best values found by four algorithms over 30 runs for six datasets.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Criteria</th>
<th>HSBEEK</th>
<th>HS</th>
<th>PSO</th>
<th>K-Means</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor Cycle</td>
<td>Average</td>
<td>2.095e+003</td>
<td>2.210e+003</td>
<td>2.976e+003</td>
<td>3.318e+004</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>2.060e+003</td>
<td>2.060e+003</td>
<td>2.077e+003</td>
<td>3.187e+004</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>2.224e+003</td>
<td>2.934e+003</td>
<td>3.053e+003</td>
<td>3.658e+004</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>1.189e+001</td>
<td>1.198e+001</td>
<td>1.549e+001</td>
<td>2.623e+001</td>
</tr>
<tr>
<td>Iris</td>
<td>Average</td>
<td>0.912e+002</td>
<td>0.931e+002</td>
<td>0.957e+002</td>
<td>1.260e+002</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>0.904e+002</td>
<td>0.904e+002</td>
<td>0.921e+002</td>
<td>1.067e+002</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>0.935e+002</td>
<td>0.947e+002</td>
<td>1.053e+002</td>
<td>1.725e+002</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>1.997e-000</td>
<td>1.754e+000</td>
<td>1.7629000</td>
<td>1.736e+001</td>
</tr>
<tr>
<td>Wine</td>
<td>Average</td>
<td>1.615e+003</td>
<td>1.628e+003</td>
<td>1.685e+004</td>
<td>1.773e+003</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>1.603e+003</td>
<td>1.606e+003</td>
<td>1.606e+004</td>
<td>1.703e+003</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>1.697e+003</td>
<td>1.698e+003</td>
<td>1.863e+004</td>
<td>1.886e+003</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>1.917e-001</td>
<td>1.146e+000</td>
<td>1.128e+001</td>
<td>1.926e+001</td>
</tr>
<tr>
<td>WBC</td>
<td>Average</td>
<td>3.010e+003</td>
<td>3.043e+003</td>
<td>3.087e+003</td>
<td>3.161e+003</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>3.002e+003</td>
<td>3.031e+003</td>
<td>3.032e+003</td>
<td>3.045e+003</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>3.110e+003</td>
<td>3.113e+003</td>
<td>3.203e+003</td>
<td>3.274e+003</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>1.420e-001</td>
<td>1.957e-001</td>
<td>1.453e+000</td>
<td>2.632e+002</td>
</tr>
<tr>
<td>CMC</td>
<td>Average</td>
<td>5.697e+003</td>
<td>5.729e+003</td>
<td>5.752e+003</td>
<td>5.877e+003</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>5.694e+003</td>
<td>5.694e+003</td>
<td>5.694e+003</td>
<td>5.842e+003</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>5.727e+003</td>
<td>5.799e+003</td>
<td>5.878e+003</td>
<td>5.934e+003</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>4.290e-002</td>
<td>1.397e+000</td>
<td>4.053e+001</td>
<td>4.632e+001</td>
</tr>
<tr>
<td>Glass</td>
<td>Average</td>
<td>2.177e+002</td>
<td>2.249e+002</td>
<td>2.522e+002</td>
<td>2.269e+002</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>2.093e+002</td>
<td>2.123e+002</td>
<td>2.157e+002</td>
<td>2.291e+002</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>2.463e+002</td>
<td>2.532e+002</td>
<td>2.554e+002</td>
<td>2.991e+002</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>7.832e-001</td>
<td>1.269e-000</td>
<td>1.461e+000</td>
<td>1.247e+001</td>
</tr>
</tbody>
</table>

From the values in Table 4.1, it has been concluded that the results obtained by HSBEEK are clearly better than the other algorithms for all data sets; HS is better than PSO; the K-Means is the worst for all data sets.
For Motorcycle data set, the optimum value of the fitness function for HSBEEK and HS is 2.060e+003. From the values of the standard deviation, it is observed that the HSBEEK algorithm is performing better than the other methods. The standard deviation value of HSBEEK, which is less than 1 represents that the algorithm is converged to the global optimum most of the times.

For Iris data set, HSBEEK and HS provide the optimum values and small standard deviation when compared to those of obtained by other methods. The average values of the fitness function for HSBEEK and HS are 0.912e+002 and 0.931e+002 respectively; the standard deviations for HSBEEK and HS algorithms are less than 1 which indicates that HSBEEK and HS are converged to the global optimum most of the times.

For Wine data set, the results of HSBEEK algorithm have outperformed other methods. It has converged to Global optimum most of the times compared to other methods.

For Wisconsin breast cancer data set, the averages of the fitness for HSBEEK and HS are 3.010e+03 and 3.043e+003; the standard deviations of the fitness for the HSBEEK and HS algorithms are 1.420e-001 and 1.957e-001 respectively. It means that the HSBEEK and HS algorithms are able to converge to the global optimum in all of the runs, while K-Means and PSO were trapped at local optimum solutions.

For CMC data set, the best global solution, the worst global solution, the average value and the standard deviation of the fitness function for HSBEEK are 5.694e+003, 5.727e +003, 5.697e +003, and 4.550e −002, respectively and the average of fitness is identical to the best global solution. It is evident from the values that the HSBEEK algorithm outperforms the other methods.

For Ripley’s glass data set, Table 4.1 shows that the average, best, worst, and standard deviation values of the fitness function for HSBEEK algorithm are much smaller than those of the other four methods and is able to produce the same partition of the data points in all runs. From the results discussed above, it can be concluded that, for all data sets, HSBEEK outperforms the other four methods.
4.5.3.2 Speed of Convergence

The second criterion for evaluating these algorithms is their convergence rate to optimal solution. Figures from 4.7 to 4.12 illustrate the convergence behaviours of the proposed HSBEEK and the other three existing algorithms. It is obvious from Fig. 4.7 to Fig. 4.12 that HSBEEK takes more time to reach the optimal solution and K-Means converges more quickly. This is because the K-Means algorithm might be trapped in local optima. Although the K-Means algorithm is more efficient than HSBEEK in terms of execution time, the HSBEEK generates much better quality clustering than the K-Means algorithm.

From Fig. 4.7 to Fig. 4.12, it has been observed that HSBEEK follows a smooth curve from its initial vectors to final optimum solution and has fewer sharp moves when compared to other algorithms. Another noteworthy point in those figures is that the final value of fitness for HSBEEK is the lowest among other algorithms. The sequence of other algorithms with respect to their fitness values are Harmony Search, Particle Swarm Optimization and K-Means. From this result, it can be concluded that the cluster produced by HSBEEK has best quality.

Fig. 4.7 Comparing the convergence of HSBEEK with other approaches for Motorcycle dataset
Fig. 4.8 Comparing the convergence of HSBEEK with other approaches for Iris dataset

Fig. 4.9 Comparing the convergence of HSBEEK with other approaches for Wine dataset
Fig. 4.10 Comparing the convergence of HSBEK with other approaches for WBC dataset

Fig. 4.11 Comparing the convergence of HSBEK with other approaches for CMC dataset
4.6 DATA CLUSTERING – PERFORMANCE EVALUATION OF GHSBEEK

This Section discusses about the performance evaluation of the novel data clustering algorithm HSBEEK.

4.6.1 Experimental Setup

The GHSBEEK algorithm has been implemented using MATLAB 7.0 and datasets were selected from the UCI machine learning repository [88]. The performance evaluation of the proposed HSBEEK approach for clustering on six different data sets selected from the UCI machine learning repository has been done and its results have been compared with the results of the K-Means, PSO, and HS clustering algorithms.

For GHS algorithm, the following parameters were set to the values as recommended in [54]: Size of the harmony memory = 15, HMCR = 0.9, PAR = 0.3, BW = 0.01. The maximum improvisation number was 10000 for all test problems. For GHSBEEK also the same parameter setting has been maintained. The standard PSO has been used. In this algorithm, the inertia weight $\omega$ varies from 0.9 to 0.7 linearly with the iterations and the acceleration factors $c1$ and $c2$ have been kept as 2.0 [5].
4.6.2 The datasets applied

The same datasets which have been used to evaluate the performance of the HSBEEK algorithm have been used again to evaluate the GHSBEEK algorithm.

4.6.3 Discussion on Experiment Results

For every dataset, each algorithm has been applied 30 times individually with random initial solutions. The efficiencies of the algorithms are compared according to their quality and speed of convergence.

4.6.3.1 Evaluation of Quality

For evaluating the quality of clustering results, the metric intra-cluster distance has been used. Table 4.2 summarizes the intra-cluster distances, as defined in equation (3) of this chapter, obtained from all algorithms for the data sets considered. The average, best, and worst solution of fitness from 30 simulations, and standard deviation are presented in Table 4.2.

From the values in Table 4.2, it has been concluded that the results obtained by GHSBEEK are clearly better than the other algorithms for all data sets; GHS is little better than PSO; K-Means is the worst for all data sets.

The Comparisons of the convergence of the proposed GHSBEEK based clustering with GHS in terms of total Mean-Square quantization Error for all the data sets have been shown in graphs from Fig. 4.13 to Fig. 4.18.

It has been observed that GHSBEEK follows a smooth curve from its initial vectors to final optimum solution and has fewer sharp moves when compared to GHS. It has also been observed that the final value of fitness value for GHSBEEK is lower than that of GHS. From these results, it can be concluded that the cluster produced by GHSBEEK has best quality.
Table 4.2 Comparison of intra-cluster distances of GHSBEEK, GHS, PSO and K-Means

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Criteria</th>
<th>GHSBEEK</th>
<th>GHS</th>
<th>PSO</th>
<th>K-Means</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor Cycle</td>
<td>Average</td>
<td>2.080e + 003</td>
<td>2.080e + 003</td>
<td>2.976e + 003</td>
<td>3.318e + 004</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>2.060e + 003</td>
<td>2.060e + 003</td>
<td>2.077e + 003</td>
<td>3.187e + 004</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>2.152e + 003</td>
<td>2.572e + 003</td>
<td>3.053e + 003</td>
<td>3.658e + 004</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>1.189e + 001</td>
<td>1.198e + 001</td>
<td>1.549e + 001</td>
<td>2.623e + 001</td>
</tr>
<tr>
<td>Iris</td>
<td>Average</td>
<td>0.908e + 002</td>
<td>0.915e + 002</td>
<td>0.957e + 002</td>
<td>1.260e + 002</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>0.904e + 002</td>
<td>0.904e + 002</td>
<td>0.921e + 002</td>
<td>1.067e + 002</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>0.917e + 002</td>
<td>0.939e + 002</td>
<td>1.053e + 002</td>
<td>1.725e + 002</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>1.997e – 001</td>
<td>1.754e + 000</td>
<td>1.7629 + 000</td>
<td>1.736e + 001</td>
</tr>
<tr>
<td>Wine</td>
<td>Average</td>
<td>1.300e + 003</td>
<td>1.538e + 003</td>
<td>1.685e + 004</td>
<td>1.773e + 003</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>1.227e + 003</td>
<td>1.446e + 003</td>
<td>1.606e + 004</td>
<td>1.703e + 003</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>1.528e + 003</td>
<td>1.652e + 003</td>
<td>1.863e + 004</td>
<td>1.886e + 003</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>1.917e – 002</td>
<td>1.146e + 000</td>
<td>1.128e + 001</td>
<td>1.926e + 001</td>
</tr>
<tr>
<td>WBC</td>
<td>Average</td>
<td>3.002e + 003</td>
<td>3.034e + 003</td>
<td>3.087e + 003</td>
<td>3.161e + 003</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>3.001e + 003</td>
<td>3.008e + 003</td>
<td>3.032e + 003</td>
<td>3.045e + 003</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>3.007e + 003</td>
<td>3.110e + 003</td>
<td>3.203e + 003</td>
<td>3.274e + 003</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>1.420e -001</td>
<td>1.957e - 001</td>
<td>1.453e + 000</td>
<td>2.632e + 002</td>
</tr>
<tr>
<td>CMC</td>
<td>Average</td>
<td>5.317e + 003</td>
<td>5.440e + 003</td>
<td>5.752e + 003</td>
<td>5.877e + 003</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>5.314e + 003</td>
<td>5.528e + 003</td>
<td>5.694e + 003</td>
<td>5.842e + 003</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>5.332e + 003</td>
<td>5.536e + 003</td>
<td>5.878e + 003</td>
<td>5.934e + 003</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>5.300e -003</td>
<td>1.002e + 000</td>
<td>4.053e + 001</td>
<td>4.632e + 001</td>
</tr>
<tr>
<td>Glass</td>
<td>Average</td>
<td>1.954e + 002</td>
<td>2.053e + 002</td>
<td>2.522e + 002</td>
<td>2.269e + 002</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>1.942e + 002</td>
<td>2.103e + 002</td>
<td>2.157e + 002</td>
<td>2.291e + 002</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>2.015e + 002</td>
<td>2.127e + 002</td>
<td>2.554e + 002</td>
<td>2.991e + 002</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>1.005e – 000</td>
<td>1.489e + 000</td>
<td>1.461e + 000</td>
<td>1.247e + 001</td>
</tr>
</tbody>
</table>
Fig. 4.13 Comparing the convergence of GHSBEEK with GHS for Motorcycle dataset

Fig. 4.14 Comparing the convergence of GHSBEEK with GHS for Iris dataset
Fig. 4.15 Comparing the convergence of GHSBEEK with GHS for Wine dataset

Fig. 4.16 Comparing the convergence of GHSBEEK with GHS for WBC dataset
Fig. 4.17 Comparing the convergence of GHSBEEK with GHS for CMC dataset

Fig. 4.18 Comparing the convergence of GHSBEEK with GHS for Glass dataset
4.6.3.2 Comparing the results of HSBEEK and GHSBEEK

The results of PSO and K-Means in Table 4.2 are almost same as the results shown in Table 4.1 as there is no change in the experimental setup. The observation also reveals the fact that the standard deviations for HSBEEK, GHSBEEK algorithms are less than 1 for all datasets which indicates that they are converged to the global optimum most of the times. It is also observed from Tables 4.1 and 4.2 that GHSBEEK provides better performance when compared with HSBEEK for all datasets considered.

4.7 SUMMARY

In this chapter, two novel algorithms HSBEEK and GHSBEEK each with the time complexity of $O(KND^2)$, where K is the number of clusters, N is number of instances and D is the number of features, for solving data clustering problem have been proposed. The performance of Harmony Search and Global-best Harmony Search algorithms has been increased by employing the food source exploitation feature of the ABC algorithm which improves the members of the Harmony Memory based on their fitness values and hence improves the convergence rate of the HS and GHS. The exploitation process has been carried out in a controlled way so that the better harmony vectors enjoy the higher selection probability. These actions enabled the speedy update of harmony memory with better solutions and hence caused the search process to move rapidly towards the goal. This enhancement also avoids the problem of getting trapped into the local minima, as the chance of selecting the same harmony vector repeatedly has been minimized. This results in two optimization algorithms which can be used for solving multivariable, multimodal function optimization. These algorithms, in combination with the K-Means clustering algorithm have shown significant improvements in the performance in terms of solution quality and convergence speed compared to other optimization algorithms in the data clustering process.