CHAPTER 7

FAST GENETIC k-MEANS ALGORITHM IMPROVED FOR REDUCTION IN COMPUTATIONAL TIME

7.1 INTRODUCTION

An improved version of GKA the Fast GKA, was proposed with some improvement over it. One of the most important features is that it always converges to the global optima as GKA. Enhanced GKA, (Venkatalakshmi 2007) GKA and Fast GKA (FGKA) outperform the k-Means algorithm and conventional GA. FGKA maintains a population of Z coded solutions. Each solution \( a_0, a_1 \ldots a_N \) is coded as a string of length N. Given a solution \( S_z = a_0, a_1, \ldots a_N \), the legality ratio of \( S_z \) is given as \( e(S_z) \). The \( e(S_z) \) is equal to the number of non-empty clusters in \( S_z \) divided by the number of clusters. \( S_z \) is legal if \( e(S_z) = 1 \), and illegal otherwise. Then separate fitness function is used depending on whether \( S_z \) is legal or illegal. The fitness function is such that illegal chromosome have lower fitness such that they have lower probability for selecting for the next generation. The increase in performance of FGKA (Yi Lu et al 2004) is obtained by avoiding the conversion of invalid (empty) cluster to valid singleton cluster.

Flash GKA, which is developed by inspiration of FGKA, is found to consume less computation time than the FGKA. Here, each allele value corresponds to the cluster to which it belongs and the position represents the corresponding pixel. Each gene in the chromosome is initialized to any one of the cluster and the cluster center is initialized randomly. Due to random
initialization if invalid chromosome (i.e. chromosome with empty cluster) results, then it is given a low fitness value. If the chromosome is legal then sum-of-deviation based on difference is calculated. The sum of deviation of a chromosome is obtained by summing the TWCV (Total Within Cluster Variation) of each cluster. The fitness for the legal chromosome is obtained by scaling the sum of deviation with an exponential factor. In the selection, the chromosome with lowest fitness value is truncated and rest are retained. In order to maintain population size, the best chromosome is duplicated. Also in Flash GKA, difference based mutation is used. By the replacement of Euclidean distance by difference the computation time will be decreased. The reassignment step of KMO using the difference rather than Euclidean distance also increases the performance. Here, empty clusters are identified based on a mere count. If the count of data items in a cluster is zero, then it is identified as an empty cluster. Hence, Flash GKA can clearly distinguish an empty cluster and a cluster in which the data items overlap. The experimental results show that Flash GKA has less computation time than FGKA and it converges to the global optimum always.

7.2 FAST GENETIC k-MEANS CLUSTERING ALGORITHM

Each chromosome in Flash GKA consists of ‘N’ genes corresponding to ‘N’ patterns. Each pattern is a vector of ‘D’ dimensions recording the expression level of genes under ‘D’ monitored conditions at each time points. The objective of FGKA is to partition the N patterns in to k clusters such that the clusters have minimum TWCV. Each partitioning is represented by \( a_0, a_1, \ldots, a_N \) where \( a_n \) takes a value from \( \{1, 2, \ldots, k\} \).

FGKA maintains a population of Z coded solutions. The legality ratio of a solution Sz which represents the number of non-empty clusters divided by k is given by \( e(Sz) \). If \( e(Sz)=1 \) then it is a legal chromosome.
otherwise illegal. FGKA starts with initialization phase which randomly initializes each allele to any one of the clusters. Due to random initialization, empty clusters may result which are undesirable. However FGKA permits empty cluster but assign their TWCV as $+\infty$ and its fitness value as low. This flexibility avoids the overhead of illegal string elimination. The selection used is proportional selection. The mutation operator performs the function of shaking the algorithm out of a local optimum, and moving it towards the global optimum. In FGKA a distance based mutation is used. In order to speed up the convergence process, one step of the classical k-Means algorithm, called as k-Means operator is used. In mutation when a solution encoded by $a_0, a_1, \ldots, a_N$ is given, a random allele is selected say $a_n$ and it is replaced by $a_n'$ where $a_n'$ is the cluster number whose centroid is closest to $X_n$ in Euclidean distance. Here, $X_n$ is the intensity of the pixel corresponding to the position of the allele $a_n$. To avoid illegal string $d(X_n, C_k)$ is set to $+\infty$ if the $k^{th}$ cluster is empty.

7.3 **FLASH GENETIC k-MEANS CLUSTERING ALGORITHM**

In the Flash GKA, the objective is to partition the $N$-given data input items into $k$ clusters such that it minimizes the TWCV. In the Flash GKA, the actual problem is coded into a chromosome of length $N$. Each chromosome is considered as a solution. Each allele value in the chromosome corresponds to the cluster to which it belongs and the position of the allele corresponds to the position of the input data. Let $a_0, a_1, \ldots, a_N$ represent a chromosome such that $a_n$ represent the pixel at $n^{th}$ position and the value of $a_n$ can be any number from $\{1,2\ldots k\}$ which represent the cluster number to which it belongs. The population size is maintained as a constant in Flash GKA. The following steps are performed in Flash GKA.
7.3.1 Initialization

In the initialization step the alleles are randomly initialized to any one of the cluster. Here, the cluster centers are also initialized randomly. Each allele in the chromosome is initialized to any one value from \{1, 2 \ldots K\}. Due to the nature of random initialization, illegal chromosomes may result (i.e. chromosome with empty clusters). These illegal chromosomes are undesirable. They are however permitted in Flash GKA and given a very small value as fitness to prevent them from selection to next generation.

7.3.2 Fitness Function

In the Flash GKA the goal is to minimize the sum of deviation, hence a high fitness value is associated with a chromosome of lower sum of deviation. In the Flash GKA the illegal chromosomes (i.e. chromosome with empty cluster) are permitted. But it is actually undesirable. To prevent such illegal chromosomes from selection for future generation they are given a very low value (say zero) as fitness. The fitness for legal chromosome (chromosome with non-empty cluster) is computed by scaling the sum of deviation by an exponential factor. The fitness is obtained by equation (7.1). The sum of deviation is obtained by summing up the TWCV of each cluster as shown in equation (7.2). The TWCV is obtained by summing up the difference between the pixels in the cluster and the corresponding cluster centre. Here the scaling factor chosen is \(\exp(-0.0005)\) since its value is 1. Since the objective is to minimize TWCV, it is a minimization problem. In order to convert the minimization problem to maximization problem a negative sign is included in the fitness as \(-0.0005\).
In case of Fast GKA, fitness for illegal chromosomes (chromosome with empty cluster) is obtained by multiplying $e^{S_2 \times F_{\text{min}}}$ where $F_{\text{min}}$ is minimum fitness value of the legal string. In Flash GKA for illegal chromosomes fitness is assigned as zero. In fitness calculation, the empty clusters are identified by a mere count. In each chromosome the number of elements in the $k^{\text{th}}$ cluster is obtained by counting the number of occurrence of $k$.

### 7.3.3 Selection Operator

In Flash GKA, the selection operator retains the best chromosomes for future generations and prevents the worst chromosome from entering into the next generation.

- From the fitness calculated for each chromosome the minimum value is found.
- The worst chromosome corresponding to minimum fitness value is truncated and the other entire chromosomes are added to mating pool for future generation.
- In order to maintain population size the best chromosome corresponding to the maximum fitness value is duplicated.

In Flash GKA, the selection used is proportional selection. The selection and the fitness used in Flash GKA were experimentally found to consume less computation than FGKA.
7.3.4 k-Means Operator

The KMO used in Fast GKA is based on Euclidean distance whereas in Flash GKA it is based on difference. Since the reassignment sub-step is based on the difference rather then the distance, it consumes less time than the distance based reassignment. Due to KMO illegal chromosome (i.e. chromosome with empty cluster) may result. They are retained as such, since Flash GKA permits chromosomes with empty clusters to prevail. But however during the fitness calculation the illegal chromosome will be detected and assigned a zero value as fitness.

7.3.5 Mutation Operator

In Flash GKA, a difference based mutation is used. The mutation operator performs the function of shaking the algorithm out of a local optimum, and moving it toward global optimum. During mutation an allele \(a_n\) is replaced by another allele \(a_n'\) for \(n=1,2,…N\) and \(a_n'\) can take a value from \{1,2,…k\} with the probability defined as \(P_{m}\).

\[
P_{mut} = \frac{1.5 \times \text{Diff}_{\text{max}}(X_n) - \text{Diff}(X_n, C_k) + 0.5}{\sum_{k=1}^{\text{max}[\text{abs}(X_n - C_1), \text{abs}(X_n - C_2), \text{abs}(X_n - C_3), \ldots, \text{abs}(X_n - C_k)]}}
\]  \hspace{1cm} (7.3)

\[
\text{Diff}(X_n, C_i) = \text{abs}(X_n - C_i)
\]  \hspace{1cm} (7.4)

\[
\text{Diff}_{\text{max}}(X_n) = \text{max}[\text{abs}(X_n - C_1), \text{abs}(X_n - C_2), \text{abs}(X_n - C_3), \ldots, \text{abs}(X_n - C_k)]
\]  \hspace{1cm} (7.5)

A bias of 0.5 is added in equation (7.3) to avoid divide by zero error. The \(\text{Diff}_{\text{max}}(X_n)\) and \(\text{Diff}(X_n, C_k)\) are defined in equations (7.4) and
respectively. Then a random number is generated between \((0, 1)\). If that random number is less than \(P_m\) then mutation is performed by randomly generating an allele and assigning it to some other random cluster. As a result of mutation, empty clusters can be formed which are detected later by a mere count during the fitness calculation where such illegal chromosomes are assigned a low fitness value. The above mutation operator ensures that an arbitrary solution, including the global optimum, might be generated by the mutation from the current solution with a positive probability. Second, it encourages that each \(X_n\) is moving towards a closer cluster with a higher probability.

All the above operators such as fitness calculation, Selection operator, and KMO and Mutation operator are performed in each generation until the termination condition is reached. There are two methods of termination one is iterating the above operators for a predefined number of generations or iterating the above operators until the cluster centers remain constant between generations. The termination condition used in Flash GKA is the iteration of the above operator for a predefined number of generations.

### 7.4 EXPERIMENTAL STUDY

For the experimental purpose of the Flash GKA, the following 5\(\times\)5 array is considered, which easily explains the procedure

\[
\begin{bmatrix}
37 & 30 & 255 & 255 & 255 \\
15 & 27 & 255 & 255 & 255 \\
22 & 30 & 255 & 255 & 255 \\
255 & 255 & 255 & 255 & 255 \\
255 & 255 & 255 & 255 & 255 \\
\end{bmatrix}
\]
In this experiment, Flash GKA tries to find k globally optimum cluster centers such that it minimizes TWCV. Here the value of k is assigned as 5 and N as 25 which correspond to number of pixels. The following genetic operators are applied.

### 7.4.1 Initialization

In the initialization the chromosomes are randomly initialized. The population size is selected as 20 and it is maintained as constant. Each allele is randomly initialized to anyone of the values from \{1, 2...5\}. The allele values represent the cluster number to which it belongs. The randomly initialized cluster centers are shown in Figure 7.1. Due to random nature of initialization one or more clusters can become empty in any chromosome. One such invalid chromosome is shown in Figure 1. It is illegal since none of the 25 pixels is assigned to the cluster 3. The invalid chromosome is permitted in Flash GKA but they are undesirable. In order to prevent such undesirable chromosomes, they are given a very low fitness value.

![Randomly initialized cluster](INVALID_CHROMOSOME)

*Figure 7.1 Randomly initialized cluster*
7.4.2 Fitness Function

The objective of Flash GKA is to minimize the sum of deviation. The fitness for each chromosome is computed using Equation 7.1 is tabulated in Table 7.1. In Table 7.1 a very low fitness value is assigned to generation one compared to other generations. In generation 1 the value of fitness is very low due to random initialization of pixels to any cluster. The cluster centers start converging towards global optima from 40th generation onwards. During the fitness calculation invalid chromosomes i.e. chromosomes with empty cluster remain. These chromosomes are actually undesirable and they may be created during the initialization, KMO, Mutation step. Such illegal chromosomes are shown in Figure 7.1 and it is given a very low fitness value such as zero. In Table 7.1 chromosome 1 and 4 in generation 2 are assigned with fitness zero. The chromosome 1 and 4 are assigned zero fitness due to the presence of empty cluster. In order to understand the rate of convergence a plot of fitness against number of generation is shown in Figure 7.2.

![Figure 7.2 Plot of fitness against number of generations](image)
Selection

In the selection operator, the best chromosomes based on fitness are retained. From Table 7.1 chromosome 5 during generation 1 corresponds to the worst chromosome with the lowest fitness value 0.1119 and the chromosome 19 with fitness value 0.4521 is the best chromosome in that generation. In case if there are more than one chromosome with the highest fitness value in that generation then the chromosome which occurs first is considered as the best chromosome. Similarly, if there is more than one chromosome with the lowest fitness value in that generation then the chromosome which occurs first is considered as the worst chromosome. Based on the Flash GKA, the selection operator truncates the chromosome 5 in the generation 1 and retains the rest. In order to maintain the population size as a constant the best chromosome 19 in generation 1 is duplicated. Finally chromosome selected for the next generation are copied into the mating pool.
7.4.4 k-Mean Operator

In Flash GKA KMO, one step of k-Means algorithm is used instead of crossover. It is illustrated in Figure 7.3. The chromosome before applying KMO is shown in Figure 7.3(a). The cluster center for cluster 3 is obtained by taking the average of the pixels in that cluster and it is found to be 29 for that cluster is shown in Figure 7.3(b). Similarly, the cluster centers are found for all the 5 clusters. Then the clusters are reassigned based on the new center such that the difference between the pixels and the cluster centers are minimum than the other cluster centers. The chromosome applied after KMO is shown in Figure 7.3(c). The KMO is introduced to increase the rate of convergence but however as a result of KMO some invalid chromosomes with empty clusters may result. In order to alleviate these problems the fitness function detects such illegal chromosomes and assigns its fitness as zero.

7.4.5 Mutation

In the mutation operation, the probability of mutation $P_m$ is calculated using equation 6.3. Then a random number $r$ is generated from the interval $(0, 1)$. If this $r$ is less than the $P_m$, then mutation is performed. In mutation, a randomly generated allele is removed from the corresponding cluster and assigned to some other randomly generated cluster. As a result of mutation, empty clusters may be formed which are later detected as illegal chromosome during fitness calculation.
(a) Chromosome before applying KMO with cluster centre cen1=255
cen2=22 cen3=28 cen4=37 cen5=15

CLUSTER 3

New cluster centers: Cen1=255
Cen2=22 Cen3= 29 Cen4=37
Cen5=15

Average of all pixels in cluster 3

(b) Calculation of new cluster center

(c) Chromosome after applying KMO

Figure 7.3 Illustration of KMO

The cluster centers of the Flash GKA with and without mutation are tabulated in Tables 7.2 and 7.3 respectively. Here the termination condition is a predefined number of generations which is taken as 50. The above operators fitness, selection, KMO, mutation are repeated for 50 generations and finally it was found that the cluster center converges towards the global optima. A relative plot of Flash GKA with Fast GKA is shown in Figure 7.4. It shows that Flash GKA is faster than Fast GKA.
7.5 EXPERIMENTAL RESULTS AND CONCLUSION

The Flash GKA which is faster than Fast GKA is applied to a Remote sensed image. The image consists of three categories: vegetation area, water body and wasteland as shown in Figure 7.5. Based on the above algorithm, three random values are generated and initialized as cluster centers. Then the above operators are applied in an iterative manner for 50 generations. The final cluster centers are found to converge to global optimum. The clustered image is shown in Figure 7.6. The cluster centers obtained using Flash GKA are 51, 103 and 182. The time elapsed to find the clustered image is 4.625 seconds. FGKA is applied for the same problem, for the same number of generations and it is found that convergence took more time. Flash GKA converges to the global optima in less time and it is proved to be effective in case of clustering remote sensed images. An entire design flow of Flash GKA is shown in Figure 7.7.

![Figure 7.4 Comparison of flash GKA with fast GKA](image-url)
Table 7.2 Cluster centers with mutation

<table>
<thead>
<tr>
<th>Generation</th>
<th>Cen1</th>
<th>Cen2</th>
<th>Cen3</th>
<th>Cen4</th>
<th>Cen5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>255.0000</td>
<td>255.0000</td>
<td>15.0000</td>
<td>255.0000</td>
<td>255.0000</td>
</tr>
<tr>
<td>2</td>
<td>255.0000</td>
<td>255.0000</td>
<td>138.0000</td>
<td>170.5000</td>
<td>198.7500</td>
</tr>
<tr>
<td>3</td>
<td>255.0000</td>
<td>255.0000</td>
<td>59.4286</td>
<td>255.0000</td>
<td>255.0000</td>
</tr>
<tr>
<td>4</td>
<td>255.0000</td>
<td>27.0000</td>
<td>26.8333</td>
<td>255.0000</td>
<td>27.0000</td>
</tr>
<tr>
<td>5</td>
<td>255.0000</td>
<td>15.0000</td>
<td>22.0000</td>
<td>32.3333</td>
<td>27.0000</td>
</tr>
<tr>
<td>6</td>
<td>255.0000</td>
<td>15.0000</td>
<td>22.0000</td>
<td>32.3333</td>
<td>27.0000</td>
</tr>
<tr>
<td>49</td>
<td>255.0000</td>
<td>37.0000</td>
<td>15.0000</td>
<td>22.0000</td>
<td>29.0000</td>
</tr>
<tr>
<td>50</td>
<td>255.0000</td>
<td>37.0000</td>
<td>15.0000</td>
<td>22.0000</td>
<td>29.0000</td>
</tr>
</tbody>
</table>

Table 7.3 Cluster centers without mutation

<table>
<thead>
<tr>
<th>Generation</th>
<th>Cen1</th>
<th>Cen2</th>
<th>Cen3</th>
<th>Cen4</th>
<th>Cen5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>255.0000</td>
<td>255.0000</td>
<td>255.0000</td>
<td>22.0000</td>
<td>255.0000</td>
</tr>
<tr>
<td>2</td>
<td>255.0000</td>
<td>200.5000</td>
<td>215.0000</td>
<td>178.7778</td>
<td>198.7500</td>
</tr>
<tr>
<td>3</td>
<td>255.0000</td>
<td>255.0000</td>
<td>255.0000</td>
<td>26.8333</td>
<td>255.0000</td>
</tr>
<tr>
<td>4</td>
<td>255.0000</td>
<td>255.0000</td>
<td>255.0000</td>
<td>26.8333</td>
<td>255.0000</td>
</tr>
<tr>
<td>49</td>
<td>255.0000</td>
<td>37.0000</td>
<td>15.0000</td>
<td>22.0000</td>
<td>29.0000</td>
</tr>
<tr>
<td>50</td>
<td>255.0000</td>
<td>37.0000</td>
<td>15.0000</td>
<td>22.0000</td>
<td>29.0000</td>
</tr>
</tbody>
</table>
Figure 7.5 Original image

Figure 7.6 Clustered image
Start

Input image and number of clusters (n)

Initialize the cluster centers randomly

For Num_gen = 1 to 50

If invalid cluster result?

No: Compute Fitness Using Equation 7.1

Selection

Perform KMO

Mutation

Output the clustered Image

Yes: Assign fitness as zero

Stop

Figure 7.7 Design flow of the clustering process