IMPROVEMENT OF THE FIREFLIES-KMEANS CLUSTERING ALGORITHM

GRADUATE PROJECT REPORT

Submitted to the Faculty of
the Department of Computing Sciences
Texas A&M University-Corpus Christi
Corpus Christi, Texas

In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

By

Lubo Zhou
Fall 2017

Committee Members

Dr. Longzhuang Li
Committee Chairperson

Dr. Ajay Katangur
Committee Member

ABSTRACT

This project is to implement hybrid global cluster algorithms on high-performance Apache Spark platform. The algorithms combine present cluster algorithms and optimization algorithms for better performance. To be more specific, two kinds of variant fireflies algorithms are utilized to find global optima among several clustering possibilities to one dataset and use the global optima to initialize k-means algorithm. To improve the performance of fireflies algorithm, additional features are added into the algorithm. To test the performance of hybrid improved k-means algorithms, several experiments are performed on Apache Spark platform based on MapReduce model.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

BACKGROUND AND RATIONALE

Nowadays, big data related technologies are playing a dominant role in computer science industry due to the amount of data we are dealing with. Consequently, data mining or machine learning is one of the most important areas that researchers put strength into. Generally, data mining techniques can be roughly divided into two aspects - classification and cluster. Classification, or supervised machine learning, indicates that to make the computer recognize multiple known classes form organized data after proper training. On the other hand, cluster, or unsupervised machine learning, means to make computer automatically divide data into several groups without known classes. The patterns in the same group should be more similar than patterns in other groups.

Efficiency and accuracy are always the primary targets when improving an algorithm. With the development of cloud computing, most real-life approaches are utilizing parallel computing on high-performance clusters, which can greatly improve the speed of the algorithm. In this project, Apache Spark platform is utilized based on MapReduce model to parallelly run the k-means algorithm. To improve the accuracy of the k-means algorithm, the algorithm is combined with fireflies algorithms to reasonably choose a starting point for the k-means, rather than generate starting centroids randomly.

1.1 K-Means Algorithm

K-means algorithm is a well-known cluster algorithm that is especially popular and easy to apply. K stands for the number of clusters that are intended to be divided

from the data. To put the data into k clusters, the algorithm first randomly generates k points that have the same dimension as each sample in the dataset. Each point refers to the center of a cluster. After that, the algorithm calculates the distance between each sample and each center and assign the sample to the point that is closest to it. After finish calculating all samples in the dataset, the center of each cluster is recalculated as the mean value of the samples in one cluster. The entire process performs several times until all samples stay still in one cluster.

Although K-means algorithm is popular and widely used, it can be improved in many ways. The major drawback of k-means algorithm is that a cluster may be trapped in local optima, or it is unstable, in another word since the starting centroids are randomly generated. Therefore, the choice of starting centroids is critical to k-means. Instead of choosing starting centroids randomly, an optimization algorithm, such as fireflies algorithm used in this project, can be used to find an okay position for the k-means to start. This approach can greatly improve the stability of k-means. If the initiate centroids are generally fine to the clusters each time, k-means can perform very well and generate consistent cluster results.

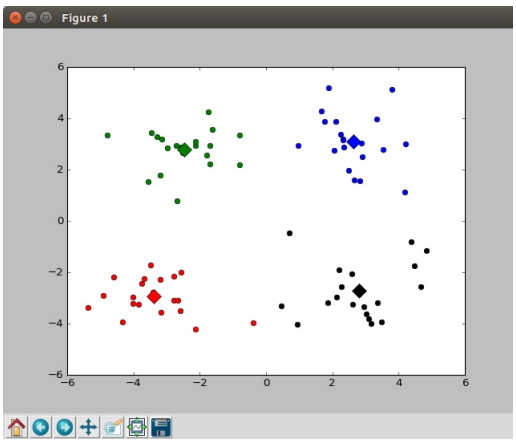

Figure 1. Correct Cluster

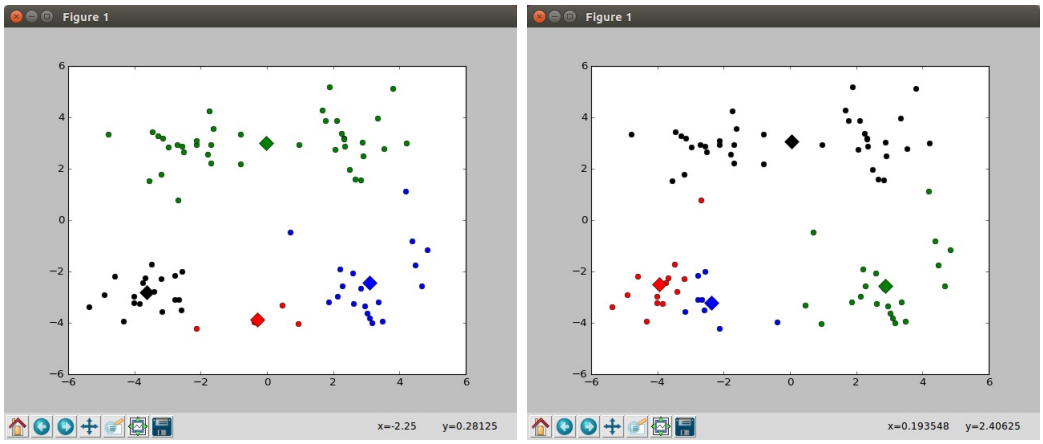

Figure 2. Incorrect cluster result (trapped into local optima)

1.2 Global Optima Algorithms

As the name states, optimization algorithms can find the optimized value among several instances. There are many kinds of optimization algorithms. For example, genetic algorithm [3], particle swarm optimization algorithm [5] and fireflies algorithm [10]. There are similar researches of improving cluster algorithms that combine optimization algorithms with clustering algorithms. Please find more details in related work section.

In this project, fireflies algorithm is utilized. The detailed approach is explained in next chapter.

1.2.1 Genetic Algorithm

The genetic algorithm is a famous optimization algorithm presented by John Holland [3] in 1975. The genetic algorithm is inspired by the mechanism of natural selection. In GA, each candidate to evolve indicates one solution, and the evolution process is also the process of optimization.

Pseudocode of Genetic Algorithm

```
//Initialization
01 generate α feasible solutions randomly;
02 save them in the population Pop;
//Loop until the terminal condition
03 for i = 1 to δ do
//Elitism based selection
04    number of elitism ne = α * β;
05    select the best ne solutions in Pop and save them in Pop1;
//Crossover
06    number of crossover nc = (α - ne) / 2;
07    for j = 1 to nc do
08       randomly select two solutions X_A and X_B from Pop;
09       generate X_C and X_D by one-point crossover to X_A and X_B;
10       save X_C and X_D to Pop2;
11       endfor
//Mutation
12    for j = 1 to nc do
13       select a solution X_j from Pop2;
14       mutate each bit of X_j under the rate γ and generate a new solution X_j' ;
15       if X_j' is unfeasible
16          update X_j' with a feasible solution by repairng X_j';
17       endif
18       update X_i with X_i' in Pop2;
19    endfor
//Updating
20    update Pop = Pop1 + Pop2 ;
21 endfor //Returning the best solution
22 return the best solution X in Pop;
```

Each candidate in population maintains a set of chromosomes or genotype, which is like the identification of the specific candidate. The optimization is produced by several iterations, and the population in each iteration is called a generation. In each generation, the algorithm evaluates the fitness value of each candidate, and the more fit candidates are randomly selected. The chromosomes of each candidate then change to learn the more fit ones, or randomly mutate to become a new generation. After that, the new generation is passed to next iteration. The algorithm stops when predetermined iteration time is achieved, or the optimization fitness value reached

the preset requirement.

As a classic optimization algorithm, genetic algorithm and fireflies algorithm shares many similarities. However, genetic algorithm contains more random factors like mutation. However, a standard fireflies algorithm does not include the random change of chromosomes, but randomly alter the degree of change when moving the fireflies.

1.2.2 Particle Swarm Optimization Algorithm

Particle swarm optimization was originally presented by J. Kennedy and R. Eberhart [5]. This optimization algorithm starts from random values, and try to find optimization value in iterations. PSO also use fitness value to evaluate the quality of solutions, but with a simpler rule than the genetic algorithm. PSO does not include the crossover and mutation of GA but searches for the optimization value to reach an optimized solution. PSO is easier to apply, along with the advantage of accurate and fast, and PSO does not include too many parameters to test on.

Pseudocode of Particle Swarm Optimization Algorithm

```
Initialize particles population
do
   for each particle p with position x_p do
      calculate fitness value f(x_p)
      if f(x_p) is better than pbest_p then
         pbest_p = x_p
      endif
   endfor
   Define gbest_p as the best position found so far by any of p's neighbors
   for each particle p do
      v_p = computeVelocity(x_p, pbest_p, gbest_p)
      x_p = updatePosition(x_p, v_p)
   endfor
while (Max iteration is not reached or a stop criterion is not satisfied)
```

PSO is inspired by the activity of birds when they search for food. Each bird

is called a particle in PSO. Each particle maintains the coordinate information and fitness value, just like a firefly in fireflies algorithm. For each iteration, the algorithm finds the particle with highest fitness value or the bird who is closest to food. The other particles in population follow the best particle and search around it. Each particle remembers the best place it reached. The algorithm also finds out the best fitness among all particles for each iteration. The particles move based on these two values. Each particle also holds a value of speed to determine the direction and length they are moving in space, and the speed value changes dynamically based on the experience of itself and other particles.

1.2.3 Fireflies Algorithm

The fireflies algorithm was originally proposed by Xinshe Yang [10]. As an optimization algorithm, fireflies algorithm simulates the movement activity of fireflies. Each firefly instance has a brightness or fitness value. The algorithm evaluates all fireflies and decides the brightness of each one. After that, the algorithm moves all fireflies iteratively. The movements of fireflies are based on the brightness. Darker firefly is attracted by brighter ones and moves toward it. After one round of movement, the algorithm reevaluates the brightness of each firefly and move them again. Several generations of movement are performed and finally obtain the global optima, the best firefly.

Pseudocode of Fireflies Algorithm
Initialize algorithm parameters:
MaxGen: the maximum number of generations
Objective function of f(x), where x = (x1,.....,xd)T
Generate initial population of fireflies or xi (i=1, 2,..., n)
Define light intensity of Ii at xi via f (xi)
While (t < MaxGen)
  For i = 1 to n (all n fireflies)
  For j=1 to n (all n fireflies)
    If (Ij > Ii), move firefly i towards j; end if
    Evaluate new solutions and update light intensity;
  End for j;
  End for i;
  Rank the fireflies and find the current best;
End while;
Post process results and visualization;
End procedure;

In this project, three kinds of fireflies algorithms are implemented as the initializer of the improved k-means algorithm. In the approaches, each firefly stands for one possibility of cluster results. The brightness of fireflies is evaluated based on how good the clusters are. Therefore, the best firefly can be a good starting position for the k-means.

1.3 MapReduce Model

To implement the algorithms in parallel, specific computation model is needed. In this project, MapReduce model is utilized to perform parallel computing on Spark platform. In general, the model handles computation tasks in map function and reduce function. The dataset is split into pieces and broadcast to all mappers into clusters. All the pieces together are called RDD, as one special data structure. The mappers then map old data to processed data by map function. After that, reduce function is called to collect map results from all distributed hosts.

For the k-means algorithm, specifically, the entire process of the map and

reduce can be briefly described as follow. Map function requires a set of centroids, index or offset of samples and value of samples as input parameters. The map function calculates the distance between the input samples with all centroids, and assign the samples to closest cluster and output the result. After that, a combine function sums up the values of samples in one cluster and count the total number of them. Finally, the reduce function take the outputs of combine function from all hosts and calculate the mean value of each cluster as a new center after summing up the values of the same cluster. The entire process can repeat several times until all samples stay still in a cluster.

1.4 Related Work

There are lots of similar approaches utilized ether parallel platforms or optima algorithms on cluster algorithm. The major idea, take k-means as an example, is to initialize k-means with several considered k centroids, rather than take random centroids.

Tahereh Hassanzadeh and Mohammad Reza Meybodi [2] proposed a hybrid approach that also combines fireflies algorithm with the k-means algorithm, however, in a different strategy. In their algorithm, to initialize the centroids for the k-means algorithm, the firefly need to be defined to higher dimensions to match the k-means dataset divisions. For example, if the dataset that needs to be clustered in two dimensions, can the dataset need to be divided into two clusters, the firefly then has $2 \text{ X } 2 = 4$ dimensions. The result optima point or the left best firefly becomes a point at [x1, y1, x2, y2], indicating the two centroids for the k-mean algorithm. Different type of dataset can be initialized with different spaces, and the fireflies in the space are all randomly generated to fill the space up. This approach can generally improve

the stability of k-means by giving certain spread centroids. However, for a complexly distributed data set, this method may not be able to give positive results since the beginning centroids have a chance to fall into local optima.

Similarly, K. Krishna and M. Narasimha Murty [6] proposed a hybrid optimization k-means utilizing genetic algorithm. In the approach, each allele in the population is initialized with a random cluster number, and the number changes during the involution iterations, which is very alike our fireflies k-means algorithm. However, the genetic algorithm calculates k-means for every iteration. Fireflies k-means divide optimization and cluster into two stages.

Gilberto Viana de Oliveira and Murilo Coelho Naldi [1] also proposed improved k-means algorithm using map reduce model. This approach gives k-means the ability to evolve itself so that k value is not necessary to provide. Their approach is entirely base on map reduce, rather than utilizing the parallel platform for speed up. For our hybrid approach, we can implement it on both local and parallel platform.

N.M. Abdul Latiff [7] and his team implemented a hybrid k-means cluster algorithm combing with Backtracking Search Optimization Algorithm to cluster protocols that eliminate energy inefficiencies in wireless sensor network, which obtained a good result. Jixiong Hu [4] and his group also made an innovation to combine fruit fly optimization algorithm with k-means to improve the cluster performance. They propose a hybrid fruit fly optimization and differential evolution by utilizing modified smell concentration judgment value, and to replace the stochastic search by the differential vector. The team of Caiquan Xiong [9] presented an interesting approach to improve the stability of k-means when performing text cluster. In their approach, they calculate the density of each data point from the dataset to

be clustered. By comparing the density of each data point, the isolated points are found and removed. After that, they randomly choose starting centroids from the non-isolated points. This approach is simple and straightforward, which inspired me a lot in improving my algorithms. For the fireflies k-means algorithm in the report, we also evaluate the solution of the cluster by calculating the sum of squared error of the data points. However, the sum of squared error helps us to determine how good one solution is, rather than the quality of data points.

CHAPTER 2

FIREFLIES K-MEANS ALGORITHM

This project includes three hybrid k-means algorithm, which are Fireflies K-means Algorithm (FK), Probabilistic Fireflies K-means Algorithm (PFK) and Greedy Fireflies K-means Algorithm (GPFK). In the three algorithms, FK was proposed by Kazunori Mizuno and his group [8], and we proposed the other two approaches to improve his original idea.

The general structure of all these three algorithms can be divided into two stages. The first part is the algorithm to find the globally optimal solution, which is either fireflies algorithm, probabilistic fireflies algorithm or greedy probabilistic fireflies algorithm. After that, the k-means algorithm uses the optimal solution to generate the starting k centroids and start the clustering process. Since the algorithm is designed for Spark platform, the algorithm is modified as MapReduce structure. The intermediate values are stored in memory to increase the speed of platform, which is the design improvement of Spark.

2.1 Algorithm Mechanism

The basic idea of Kazunori's FK is to treat each firefly as a solution of the cluster. To be more specific, the fireflies are generated in a multi-dimensional space, and the number of dimensions is equal to the length of data set to be clustered. Each dimension represents a line of data, or a data point, in the dataset. Therefore, each firefly's coordinate in the space indicates the cluster assignment for the entire data set, and one value on one axis of one direction is 0 to k-1, indicating the cluster index assignment for each data point.

In the fireflies' stage of FK, the process can be divided into three steps, which are initialization, evaluation, and movement. Initialization is processed only ones when the program starts, and generate the preset number of fireflies. After that, the predetermined number of iterations of movement and evaluation is executed. Finally, the best firefly is passed to the k-means algorithm as the initial solution.

Pseudocode of Fireflies K-means Algorithm

```
t = 0, s* = φ, γ= 1.0;
   // initialize the generation counter, best solution, and the absorption coefficient
p^(0) = InitializeFA();
   // generate the initial firefly population
while (t < T) do
   α^(t) = AlphaNew();
   Evaluate(p^(t), f(s));
   // evaluate the solution, s
   OrderFA(p^(t), f(s));
   //sort in order of fitness values
   s* = FindTheBestFA(p^(t), f(s));
   // a best solution, s*, at the t-th generation is maintained
   p^(t+1) = MoveFA(p^(t));
   // update of the position of fireflies
   t = t + 1;
end while
ApplyKmeans(s*);
// excecution of k-means for the best solution of firefly algorithm.
```

2.2 Data Structure of Firefly

The Firefly class includes the following members. Variable dimensions, which is the length of the entire data set, in this case. Boundary, the range of the values on each axis, which is k in our approach. Fitness, or brightness, is the value to determine how bright one firefly is, the brightest firefly is certainly the best one and is the most attractive one, which draws other fireflies toward him. Cluster array,

similar to the chrom of the genetic algorithm, the coordinate value of each firefly, which is a long array that has the same length as the data set to be clustered. Each element in the cluster array is the assigned cluster index for the corresponding data point. The entire array exists as a point indicating the position of the firefly in the multi-dimensional space.

2.3 Initialization of Firefly

Initialization is the first step in FK. During generation, each firefly creates all class members and randomly assign an integer from 0 to k-1 to each element of the cluster array, which means determined a random position in the multi-dimensional space. So, each data point in the data set is randomly assigned to one cluster, and the one firefly becomes a solution of the cluster. The number of fireflies to be created is predetermined before the program start. The algorithm then appends all initialed fireflies to a population list, to be evaluated.

2.4 Evaluation of Firefly

After the initialization, evaluation and movement steps are called for several iterations. Evaluation step is to decide the fitness or brightness of one firefly calculated by on the cluster array. To be more specific, the fitness is decided according to the similarity of data points that assigned to one cluster.

In this part, to calculate the similarity of data points in one cluster, we see each data point as a point in a space with the dimension that same to the number of attributes of one line of date. For example, if there are 5 attributes in one line of data, the space is, therefore, a 5-dimensional space. And the values of attributes become the coordinate of the point in the space. For each cluster, the algorithm

calculates the center point based on all points in the cluster. After that, for the data points that in a same cluster, the algorithm calculates the Euclidean distance between each of them and the center point of the cluster and add the distance values together, which equals the sum of squared error of one cluster, or WSSSE, standing for Within Set Sum of Squared Error. After calculating all cluster's sum of squared error and add them up, the fitness value is the reciprocal of the total sum of squared error.

$$f(s) = \sum_{k=1}^{K} \sum_{x_i \in C_k} D(x_i, x_k) \qquad (1)$$

$$D(x_i, x_k) = \sqrt{\sum_{j=0}^{N-1} (x_{ij} - xkj)^2} \qquad (2)$$

$$x_{kj} = \sum_{x_i \in C_k} \frac{x_{ij}}{|C_k|} \qquad (3)$$

As showed in formula (1), (2) and (3), the fitness value f(s) equals to the sum of distance. The variable x means the data points from the data set. Therefore, the closer the data points are to each other, the shorter the total distance is, or the more similar the data points are in one cluster, the brighter the firefly is. Since the fitness of one firefly is determined by the cluster array of itself, we can also say that the brightness of the Firefly is based on its position in its multi-dimensional space.

In addition, to reduce the size of one firefly, the firefly instance does not contain the entire data set to be clustered, but read the data cached on Spark and calculate its fitness by the data attributes.

2.5 Movement of Firefly

The evaluation step is followed by the movement step. Since each firefly has its own fitness value, for each firefly, the algorithm makes it compare with all other fireflies. If there is any other firefly that is brighter, the darker firefly moves toward the brighter one. After each firefly's movement, the algorithm checks the values of cluster array to see if any of them is beyond the boundary and fix it in range if they do pass the boundary.

There are 3 parameters that configure the movement behavior of fireflies. If we represent the attractiveness of source firefly as β, then β is affected by r, the distance between two fireflies, γ, the coefficient of light absorption at the source, and $\beta 0$, the coefficient of attraction when the distance r equals 0. The calculation of β follows the formula below.

$$\beta(r) = \beta_0 e^{-\gamma r^2} \tag{4}$$

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d}(x_{ik} - x_{jk})^2} \tag{5}$$

After the β value calculated, the movement amount of the darker firefly is the error between two fireflies times β, and plus an uniform random number from -1 to 1. The random number also times a coefficient α. The entire movement behavior is determined by the following formula.

$$s_i = s_i + \beta(r) = \beta_0 e^{-\gamma r_{ij}^2}(s_j - s_i) + \alpha\epsilon_i \tag{6}$$

Therefore, the 3 parameters we need to give before running the algorithm is α, β, and γ, standing for the random moving distance coefficient, the attractiveness

coefficient, and the light absorption coefficient. For experiments in next chapter, we used different parameter combinations for the three different fireflies k-means algorithms.

The movement of Firefly is to update the coordinate value, the cluster array, toward the better-assigned solution since the brighter firefly represents a better-assigned solution. Therefore, during the entire moving process, all the cluster solutions become better assigned.

Table 1. Data Structure and Movement of Firefly

data ID	1	2	.....	N-1	N
Bright FF	3	2	.....	1	3
Si[j]	1	3	.....	2	2

After Movement

Si[j]	2	2	.....	1	3

One of the methods to test whether a solution of the cluster is good or bad is the sum of squared error. Smaller the value is, closer the data points are to each other in one cluster, which means the groups are smaller. For the experiments in this project, the sum of squared error is the major measurement of the experiment results. The following figure shows the sum of squared error during iterations of fireflies algorithm running on a random data set. We can clearly see the drop of both the best firefly's sum of squared error and the average sum of squared error of cluster solutions.

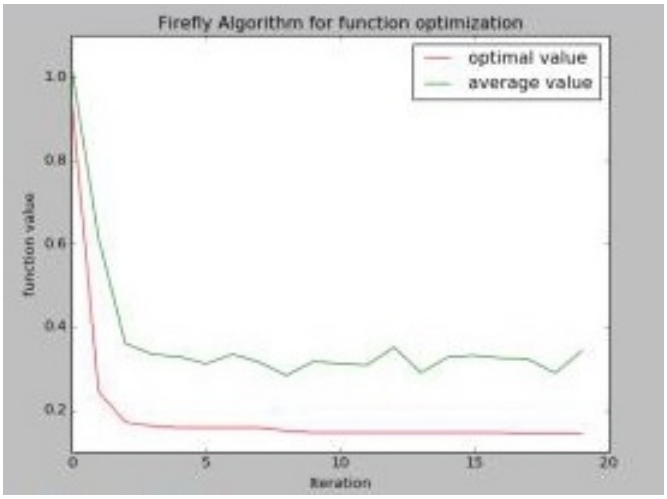

Figure 3. Changing of the best and average value of sum of squared error

2.6 Cluster Stage

After predetermined certain iterations, the firefly with the highest fitness, or the smallest sum of squared error becomes the best cluster solution and is passed to k-means algorithm for the cluster stage. K-means algorithm takes the whole data set to be clustered, and assign each data points with a cluster index based on the assignment of best firefly. K centroids are initiated by calculating the mean value of the points in same clusters, and the algorithm can iterate several times to do its standard cluster process until the moving distance of each data points is smaller than a preset value. The cluster results of k-means after fireflies algorithm can generally be stable, since the groups are already roughly shaped, and the initial centroids are separated into the groups.

2.7 The Problem of FK Algorithm

Although Kazunori and his group proposed a practical approach to initiate k-means, there is a major drawback on his FK algorithm. The main problem with this approach is that he treats each different cluster indexes as a continuous axis. It is easier for one data point to change its index from 1 to 2, but harder from 1 to 4, for example. However, this is unreasonable. Changing between different indexes should have the same distance or same difficulty. In another word, a cluster index should be a compound object formed from several elements, rather than a single value, so that the index can change towards one significant element, to change the composition of its index. In this way, the changing behavior of all cluster index can share the same difficulty.

In addition, in Kazunori's FK approach, the index number can turn to decimals during the moving process but keep as integers, which is meaningless. An index number is meaningful only when it is an integer, indicating a corresponding cluster. The fireflies in the intermediate stage during the iterations fail to represent meaningful cluster solution since the index numbers represent nothing as decimals. In the end of fireflies algorithm stage, the best firefly's cluster index values are rounded to integers. Therefore, it is hard to say that the optimization process of FK is helpful to really generate the best valuable firefly.

To fix these two problems, we improved standard FK and proposed two variant algorithms, PFK and GPFK.

CHAPTER 3

IMPROVEMENT OF FK ALGORITHM

3.1 PFK Algorithm

PFK, probabilistic fireflies k-means algorithm, is an improved algorithm based on FK. The main purpose of PFK is to propose a reasonable way to express and evolve cluster index, rather than to treat index number as a continues to value. To do so, we changed the data structure of Firefly. To be more specific, the cluster array of each firefly instance is changed to a two-dimensional array to hold more information, and the initialization and movement steps have to make some changes to fit the new data structure. The pseudocode of PFK is as follows.

Pseudocode of Probabilistic Fireflies K-means Algorithm
$p^{(0)}$ = InitializeFA();
  // generate the initial firefly population
t = 0
while (t < T) **do**
  $\alpha^{(t)}$ = AlphaNew();
  CalculateFinalClusters($p^{(t)}$.clusterChannels);
  Evaluate($p^{(t)}$.finalClusters)
  // evaluate the fireflies by finalCluster indexes
  s* = FindTheBestFA($p^{(t)}$.fitness);
  $p^{(t+1)}$ = MoveFA($p^{(t)}$.clusterChannels);
  // update of the position of fireflies by changing the clusterChannels
  t = t + 1;
end while
ApplyKmeans(s*);
// excecution of k-means for the best solution of firefly algorithm.

3.1.1 Data Structure of Firefly in PFK

In FK algorithm, the Firefly class has a cluster array member indicating the cluster assignment. However, the Firefly class in PFK algorithm has, instead of one

array, a two-dimensional array to represent the index numbers. The length of the array is still same to the length of data set to be clustered. The height of the array is same as the value of k. For each element indicating one data point in this array, there are k percentage values in the element to determine which cluster index should be assigned to this data point. The sum of the k percentage values is 1. Each percentage value represents a cluster index, and the highest percentage value can dominant the element to make it belong to the corresponding cluster. For easy description, we call this array in PFK as cluster channel array, and each element in cluster channel array is called a cluster channel. Each cluster channel contains k percentage values to determine with cluster the data point is assigned to.

All the percentage values in cluster channel array are randomly generated when constructing the firefly instance, and another array called final cluster array is filled with the fixed cluster index number based on the percentages. The final cluster array, which is like the cluster array in FK algorithm, is used to calculate the fitness value, but the movement step only alters the values of cluster channel in cluster channel array and has nothing to do with the final cluster array, which is only recalculated based on cluster channel array after the movement.

3.1.2 Initiation of Firefly in PFK

When initializing, the algorithm generates predetermined numbers of fireflies to append to population list. Each firefly's constructor fills the two-dimensional cluster channel array with random percentage number, and create the single-dimensional final cluster array. The constructor does not calculate the final cluster array, but fill it with zeros.

Generally, the time cost of initialization step of PFK is k times to the time

cost of FK, since, for each cluster index, a data set length array needs to be created while only one array with the same length is needed in FK. However, the cluster channel array in PFK makes more sense for representing cluster indexes.

3.1.3 Evaluation of Firefly in PFK

PFK also uses WSSSE to calculate the fitness value of each firefly as showed in formula (1). In the evaluation step, the algorithm calculates the final cluster array for each firefly in the population list based on their cluster channel array, and then use the final cluster array to perform the same calculation in FK to obtain the fitness value of each firefly. The time cost of evaluation in PFK is k * n more than the time cost in FK, where n is the length of data set to be clustered.

3.1.4 Movement of Firefly in PFK

The movement of fireflies in PFK follows the same rules showed in formula (6) as in FK. Meanwhile, for each movement activity between two fireflies, the calculation is performed k times, since the cluster channel array in PFK is k high. The movement behavior does not alter the final cluster array in one firefly, but changing the percentage values in cluster channel instead. After each movement activity between two fireflies, the algorithm restrains the percentage values in charging of determine cluster index for one data point and make the sum of these values to be 1. This restraining step replaces the boundary checking at the end of the movement. The time cost of movement in PFK is also k times to which in FK.

Table 2 is an example of the movement activity from one iteration in PFK when k is 4. In the table, best FF is the elite firefly with the showed cluster channel array and final cluster array. Si[j] stands for a random firefly in the population that

Table 2. Data Structure and Movement of Firefly in PFK

data ID		1	2	.....	N-1	N
Bright FF	Cluster Channels	[0.3, 0.2, 0.4, 0.1]	[0.6, 0.1, 0.2, 0.1]	.....	[0.03, 0.07, 0.7, 0.2]	[0.1, 0.55, 0.3, 0.05]
	Final Clusters	2	0	.....	2	1
Si[j]	Cluster Channels	[0.5, 0.1, 0.1, 0.3]	[0.2, 0.3, 0.25, 0.25]	.....	[0.4, 0.2, 0.3, 0.1]	[0.1, 0.3, 0.2, 0.4]
	Final Clusters	0	1	.....	0	3
After Movement						
Si[j]	Cluster Channels	[0.4, 0.15, 0.3, 0.15]	[0.5, 0.2, 0.2, 0.1]	.....	[0.1, 0.1, 0.6, 0.2]	[0.1, 0.5, 0.3, 0.1]
	Final Clusters	0	0	.....	2	1

is moving toward best FF. In the two fireflies, the elements in final cluster arrays are decided by the cluster channels in cluster channel arrays. Take the first column as an example. The cluster channel of data point 1 in best FF is [0.3, 0.2, 0.4, 0.1]. Therefore, the final cluster of this data point in best FF is 2, the third cluster when k equals 4. In Si[j], the cluster channel of data point 1 is [0.5, 0.1, 0.1, 0.3], so the final cluster is 0, the first cluster index since the first percentage value in this cluster channel is the biggest. The biggest percentage value in a cluster channel decides the cluster index number in the corresponding final cluster array. When Si[j] moves to best FF, the percentage values of Si[j] change close to the values of best FF. We can see from the table that for data point 1, the cluster channel of Si[j] changes to [0.4, 0.15, 0.3, 0.15], which gets closer to the cluster channel of best FF. However, the final cluster of data point 1 in Si[j] does not change after this movement behavior since 0.4 is still the biggest value in its cluster channel. On the other hand, the final cluster of data point 2 changes after the movement. This example shows that each cluster number for each data point gets an equal opportunity to change from one to another.

3.1.5 The Pros and Cons of PFK Algorithm

The advantage of PFK is from the new data structure of Firefly. To be more specific, the two-dimensional cluster channel array allows a firefly to use k percentage

values to represent the cluster index assignment for one data point. The movement step makes a firefly move the percentage values closer to a better one. The cluster assignment may change if the proportion of one percentage value changed to be the highest value but may remain the same if the percentage value of the current index is still dominant after movement. The cluster numbers keep as integers to be meaningful all the time.

However, the drawback of PFK is also obvious. The total time cost of PFK's optimization stage is about k times to the time cost of FK since the initialization and movement step all cost k times more. This feature makes PFK a very time-consuming algorithm when dealing with big data, especially the ones with too many categories. Therefore, it is crucial to make some change to the algorithm strategy to reduce the time cost. Since movement step is performed repeatedly and is the time-consuming part of the entire algorithm, we modified the movement strategy and proposed another new approach, GPFK.

3.2 GPFK Algorithm

GPFK utilizes the same data structure of PFK to represent one cluster index by k percentage numbers. Therefore, the initialization and evaluation steps are the same to those in PFK. However, the movement step of GPFK is a different strategy. Generally, to reduce the time cost of movement, the fireflies move only to the best firefly, ignoring the other better ones. This strategy follows the classic greedy algorithm design, so the algorithm is called greedy probabilistic fireflies k-means. During the specific movement process, the firefly follows two evolution principles, which are random value fluctuation and random static segment. The pseudocode of GPFK is as follow.

Pseudocode of Greedy Probabilistic Fireflies K-means Algorithm

```
p^(0) = InitializeFA();
   // generate the initial firefly population
t = 0
while (t < T) do
   α^(t) = AlphaNew();
   CalculateFinalClusters(p^(t).clusterChannels);
   Evaluate(p^(t).finalClusters)
   // evaluate the fireflies by finalCluster indexes
   s* = FindTheBestFA(p^(t).fitness);
   p^(t) = s*;
   foreach(firefly in p^(t)) do
      segment = CalculateStaticSegment(t);
      p^(t+1).append(MoveFA(firefly.clusterChannels - segment));
   end foreach
   // update of the position of fireflies
   // the segment that holds still increases while t increases
   t = t + 1;
end while
ApplyKmeans(s*);
// excecution of k-means for the best solution of firefly algorithm.
```

3.2.1 Random Value Fluctuation

In each iteration, after the evaluation step, the algorithm traverses all fireflies and determine the best firefly with highest fitness value. The other fireflies then all move to the point of the elite firefly. In another word, all fireflies become the best one. After that, the fireflies randomly change their cluster channel percentage values to find a better place that may lead to higher fitness values. In a more visual explanation, the fireflies all gather to the brightest one and randomly roam around in the multi-dimensional space. The movement distance of each firefly in GPFK also follows the same rule of formula (6) as in FK. These random movements may cause a firefly to reach a higher position and become better than the previous elite firefly, or result in an opposite status. This random movement activity is called random

value fluctuation of a firefly. After all fireflies' movement, the algorithm performs the evaluation step and decide a new brightest firefly.

3.2.2 Random Static Segment

Since all fireflies move from the same starting position, it is preferable to make each firefly goes toward disparate directions. To do so, we import the second principle, random static segment. This principle means when moving a firefly, to randomly hold certain cluster channels in cluster channel array still, while other elements fluctuating around. In another word, there are always certain directions the firefly will not go, and each firefly moves toward distinct directions, which increase the stability of the algorithm since fireflies move from the best one.

In addition, along with the iterations of the algorithm, the best firefly becomes closer and closer to the ultimate cluster solution, so the size of the segment in cluster channel array that holds still increases, and the parts that are going to fluctuate becomes less and less. More iterations the algorithm performs fewer index changes. This mechanism we applied is to make the algorithm more stable and time efficient since the amount of data requires calculation reduces trough time.

For example, when the iteration is less than 10 times, we set up a random mode RM equals to 3. For each cluster channel in cluster channel array of the firefly that is ready to move, before changing its values, we generate a random integer number R from 1 to 3 and then calculate the remainder of R % RM. If the remainder is 0, the cluster channel continues to perform the movement, otherwise, it passes the movement for this round. Therefore, each cluster channel gets 1/3 chance to perform the movement, which also means that generally, only one-third of the cluster channels move. When the iteration is more than 10 times but less than 50 times, for another

example, the random mode RM change to 4. Only one-fourth of the cluster channels move in this case. More iterations are executed, bigger the RM is, and lesser the cluster channels move in that round.

Theoretically, the GPFK algorithm should be able to stop automatically without a preset iteration time, and the final solution should be a fine clustered result. However, it needs too many iterations to reach the result. It is more practical to use GPFK only to give a decent starting position to initialize k-means and let k-means to finish the cluster stage. The experiments in next chapter indicate that k-means algorithm is a considerable stable cluster algorithm when the starting centroids are properly arranged.

CHAPTER 4

EVALUATION AND RESULTS

To test the performance of FK, PFK, and GPFK, we did several experiments to evaluate the three algorithms. in the following sections in this chapter, we first evaluate the general mechanism and property of each algorithm using random and small data sets. After that, we tested the algorithms on four small data sets and one big data set to evaluate the performance and stability of them.

4.1 General Mechanism Evaluation

In the general mechanism evaluation, we first generated a random 100 lines dataset, values from 0 to 3, to observe the initial centroids that each algorithm can provide for k-mans after the fireflies optimization stage. In this test, we set k as 4. We can clearly see from the figures below that the centroids are all close to the center of data points at the beginning since the cluster indexes are randomly assigned.

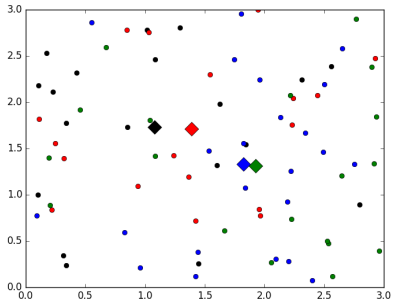

Figure 4. Positions of Centroids After Random Cluster Assignment

After that, we let FK, PFK, and GPFK perform optimization stage as we designed. For the three parameters, α, β and γ, we use 0.5, 1.0 and 1.0 on FK,

0.06, 1.0 and 0.000001 on PFK, 0.75, 1.0 and 0.000001 on GPFK. Since the three algorithms perform in the different mechanism, it is not fair to force them to use the exact same parameters. These values can reach a fine performance for each of the algorithms after tests. The parameters we used on FK is also same to what Kazunori [8] uses in his experiments. After 100 iterations, the following optimization results of FK, PFK, and GPFK, from left to right, gives an obvious observation of the performance of each algorithm. PFK and GPFK performed as better initiators since the starting centroids they gave are more spread out and fit potential clusters.

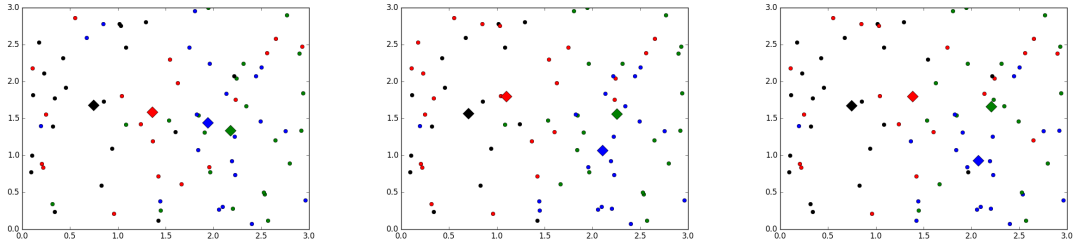

Figure 5. Positions of Centroids After Optimization

Comparing with the final fine clustered result after k-means, the result of PFK and GPFK are closer to the real cluster assignment.

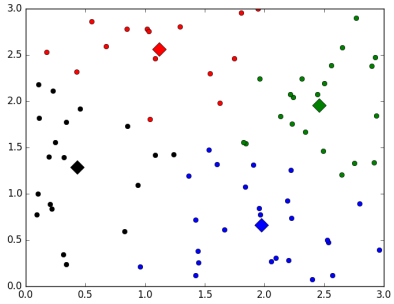

Figure 6. Real Clusters

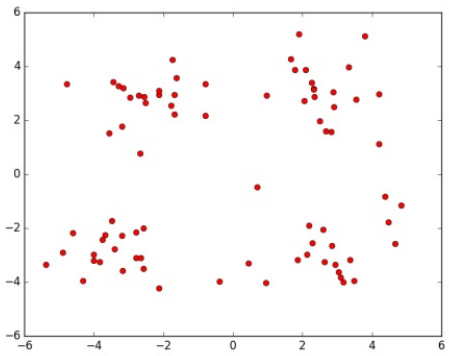

Figure 7. Data Points Distribution of 80 lines Data

To further observe the ability of PFK and GPFK, we tested the two algorithms on another dataset with 80 lines of data points. The boundaries between each cluster in this dataset are more clear and obvious so that we can directly find out which fireflies algorithm can give better starting centroids.

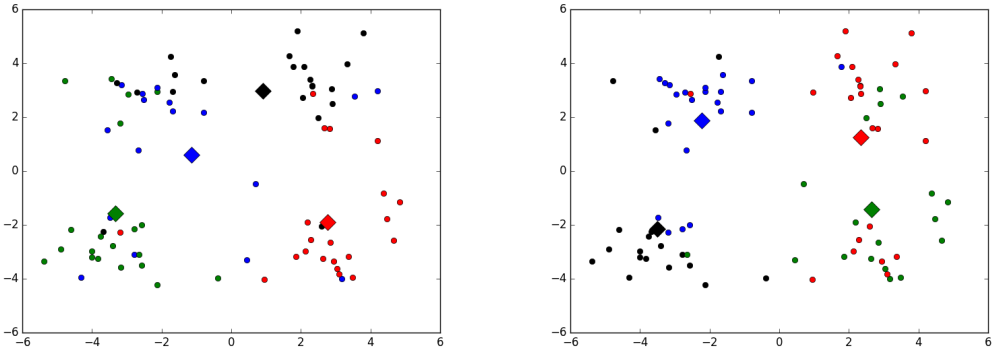

Figure 8. Optimization Results of PFK and GPFK

From the above figures, we can clearly see that the data was roughly clustered after the optimization stages of PFK and GPFK, and the starting centroids are separated into four clusters. Although one of the centroids of PFK is a little deviated to the group, the starting position is already properly arranged for the k-means.

Therefore, the result of k-means can be consistent using these initiators.

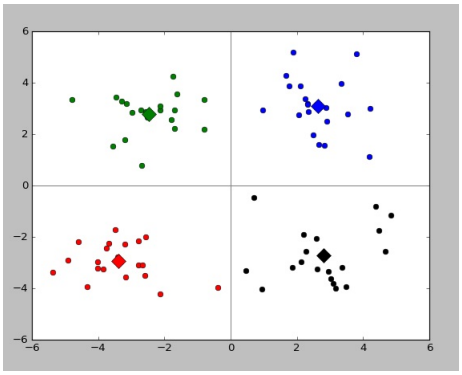

Figure 9. Real Clusters

4.2 Real Data Experiments

To evaluate the performance of FK, PFK, and GPFK in details, we did further experiments on these algorithms using real-life data. All the experiments below were processed on the High-Performance Computing platform of Texas A & M University-Corpus Christi. Six nodes were used for Spark parallel computing for each experiment, each node contains two Xeon E5-2680v2 processors, 256GB of DDR4 memory, and 1 TB of local disk. The nodes are interconnected with a Mellanox FDR InfiniBand in one level, non-blocking, topology. The version of Spark platform we use is 2.0.0, and the version of Python used for coding the experiment programs is 2.6.6. The parameters of each algorithm maintain the same as below.

Table 3. Parameters for fireflies algorithms

	α	β	γ
FK	0.50	1.0	1.0
PFK	0.06	1.0	0.000001
GPFK	0.75	1.0	0.000001

4.2.1 Small Data Experiments

In the experiments below, we use real data from the UCI Machine Learning Repository. The dataset we use are "Iris data set", "Glass data set", "Cancer-Int data set", and "Haberman data set", which have the benchmarks below.

Table 4. Data Sets for Experiments

Number of	Data lines	Attributes	Clusters
Iris	150	4	3
Glass	214	9	6
Cancer-Int	699	9	2
Haberman	306	3	2

The iris data set is consisted of three kinds of flowers, Setosa, Versicolour, and Virginica. The attributes refer to sepal length, sepal width, petal length, and petal width. The glass dataset includes six types of glass for different usage, building windows float processed, building windows non-float processed, vehicle windows float processed, containers, tableware, and headlamps. The attributes are chemical elements a kind of glass can contain, RI, Na, Mg, Al, Si, K, Ca, Ba, and Fe. The cancer-int data set is the data to show the nine properties of breast cancer patients and be classified as benign or malignant. Haberman data set is the records of the patients who experienced surgery of breast cancer, and been classified to whether the patient died in five years after the surgery or survived five years and longer.

For each dataset, we performed FK, PFK, and GPFK, and recorded the results of experiments. To test the stability of each algorithm, we performed each experiment 100 times and calculated the average values. To evaluate the performance, we recorded three values as the measurement, which is the sum of squared errors, PCA (percentage of correct answers), and time cost (in seconds). The sum of squared errors is the most convictive measurement since it tells how close the groups

are. PCA, on the other hand, can evaluate how practical the algorithm is for real-life usage. In all the experiments on small data sets, we set the population of Firefly as 20, and the number of iterations as 100.

Table 5. Small Data Experiment Results

	Just K-means			
	iris	glass	haberman	cancer
Sum of squared error	8966.5	18714.5	595155.0	1415005.9
PCA	0.874	0.321	0.441	0.959
Time cost (second)	4.75	6.11	5.32	5.48
	Fireflies K-means			
	iris	glass	haberman	cancer
Sum of squared error	7554.0	17171.2	570838.0	1415005.9
PCA	0.882	0.364	0.463	0.959
Time cost (second)	38.01	45.90	42.03	50.69
	Probabilistic Fireflies K-means			
	iris	glass	haberman	cancer
Sum of squared error	6998.0	16936.1	562042.9	1415005.9
PCA	0.887	0.383	0.478	0.959
Time cost (second)	55.01	64.85	76.64	126.47
	Greedy Probabilistic Fireflies K-means			
	iris	glass	haberman	cancer
Sum of squared error	6997.4	16287.6	561782.1	1415005.9
PCA	0.887	0.432	0.479	0.959
Time cost (second)	38.42	35.66	38.81	49.68

From the results, we can see that for each experiment, the average value of the sum of squared error from FK's results is the highest among the three algorithms, which means the clustering result is not as good as the other two. We can also see that GPFK can produce the results with the lowest sum of squared error by making the final groups tighter. The PCA of PFK and GPFK is higher than FK as well. The reason that PFK and GPFK perform better is that they make k-means more stable by providing decent initiate centroids. From the 100 records of each experiment, it is clear that k-means algorithm makes consistency results with PFK and GPFK as initiators, but make more faulty cluster results using FK, since the starting centroids

given by FK may be improper.

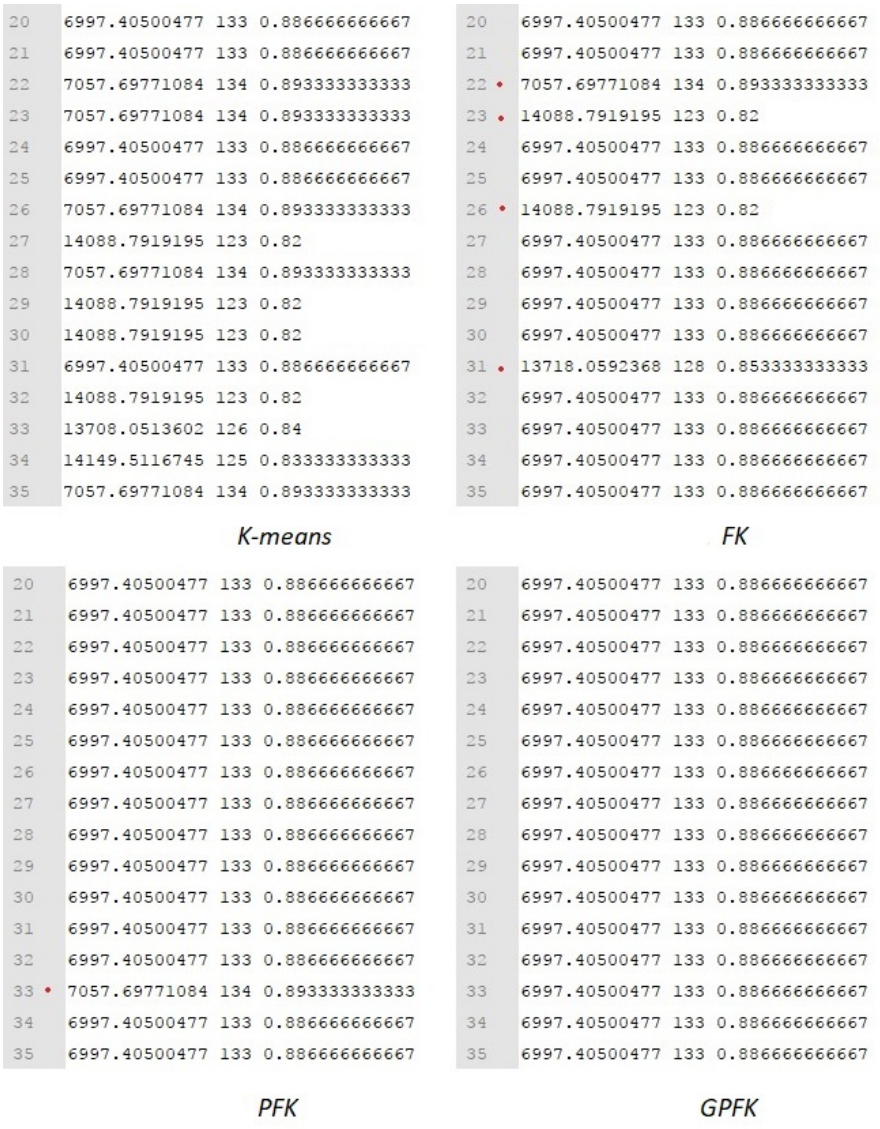

Figure 10. Records of Iris Experiments

Take parts of the experiment records of iris data as an example, the cluster results of GPFK are all consistent, maintaining the sum of squared error as 6997.4, and the results of PFK only appeared one distinct result. FK, on the other hand, made several mal cluster results, but still better than pure k-means, which fail to

give consistent cluster solutions. Therefore, it is safe to say that GPFK and PFK are more stable cluster algorithms.

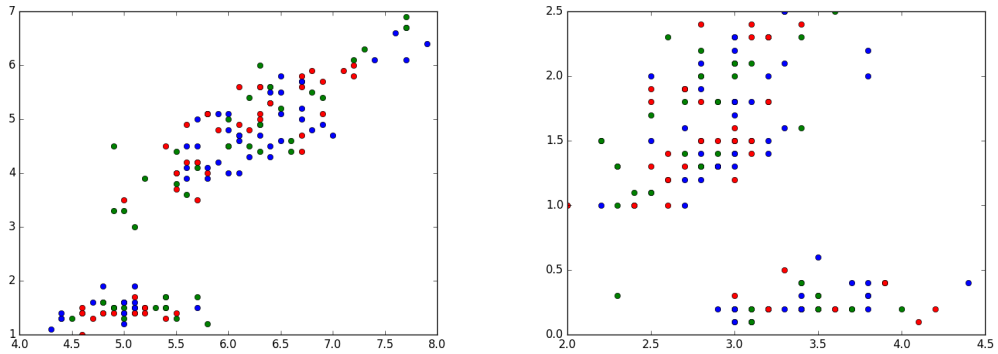

Figure 11. Cluster distribution where clusters are randomly assigned to each data

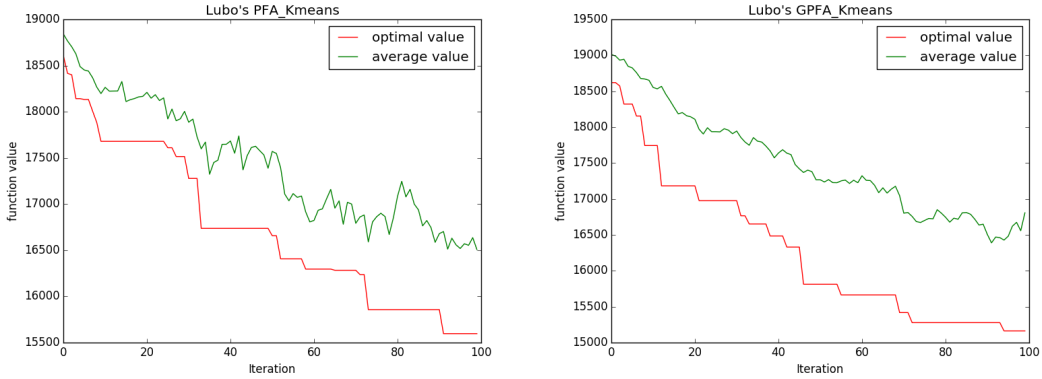

Figure 12. Sum of squared error of PF and GPF during iterations - average and best

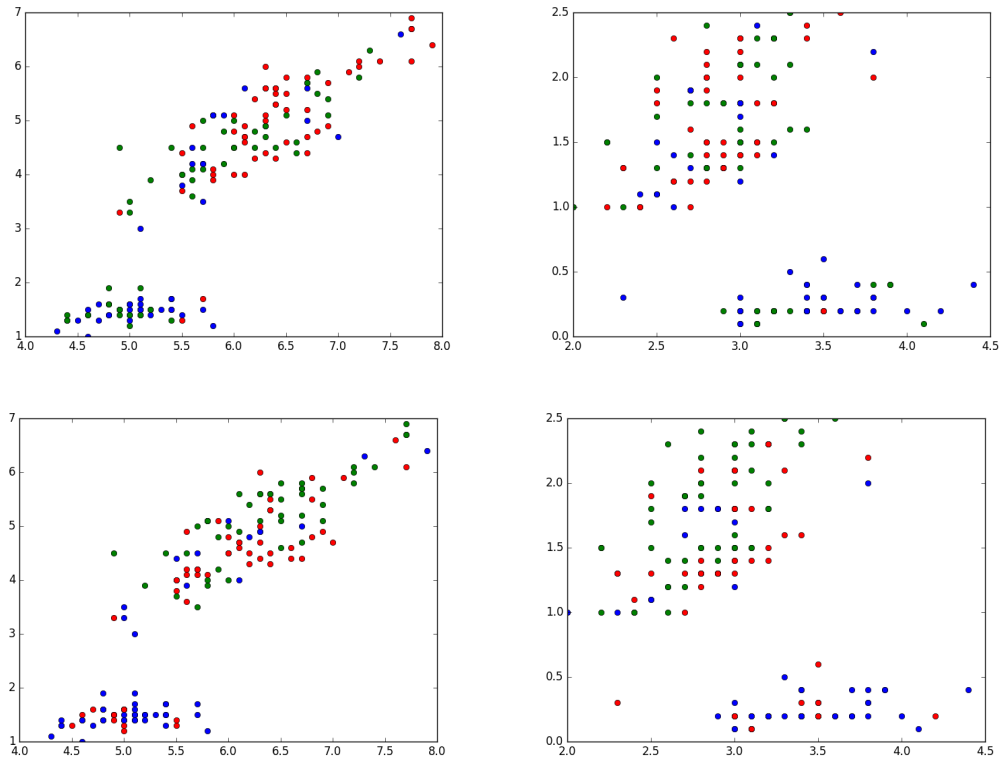

Figure 13. Cluster distribution after PF (top) and GPF(bottom) is executed

The comparison of PFK and GPFK on iris data shows that GPFK performs better than PFK. GPFK can make smaller groups since the sum of squared error dropped to a lower value after 100 iterations, which can provide a better starting position for the k-means.

However, the time cost records from the experiment results also show that PFK is very time-consuming comparing to others. This is because of the more calculation caused by the bigger sized firefly data structure. However, GPFK runs faster than PFK since it only learns from the best firefly, and the time cost is basically same to FK, or even slightly faster in some cases. Therefore, the best situation using PFK is when the requirement of stability is more important than time-efficiency, and

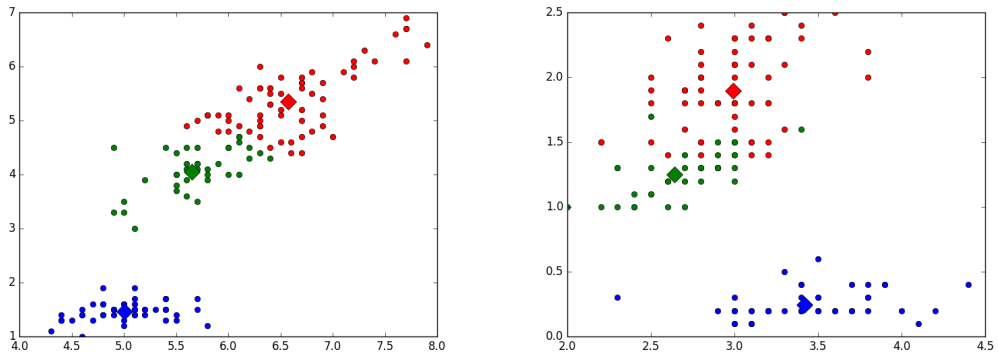

Figure 14. Correct cluster distribution

GPFK is a solution better than FK in every aspect.

4.2.2 Big Data Experiments

For the big data experiments, we use a record of watch accelerometer, which is also from UCI Machine Learning Repository. This series of data set recorded Human Activity Recognition (HHAR) from Smartphones and Smartwatches. The watch accelerometer data set we use contain about 1.5 million lines of data, and we take three attributes of float number from each line for cluster experiments. In the experiments below, we take the sum of squared error as the only measurement of cluster results, and time cost as well. For each algorithm, we perform the experiment two times. In each experiment, the population of Firefly and the iteration time are both set to 10.

The average of the sum of squared error shows that the clustering ability of GPFK is the best among all four algorithms. The results of two experiments on GPFK gave very similar result, which shows the high stability of GPFK. PFK also can reach a lower value of the sum of squared error than the k-means, which means

Table 6. Big Data Experiment Results

	Just K-means		Fireflies K-means	
	Experiment 1	Experiment 2	Experiment 1	Experiment 2
Sum of squared error	8722858.2	8720817.7	8721279.7	8534506.9
Time cost (second)	904.16	905.21	12460.19	12410.83
Average sum of squared error	8721838.0		8627893.3	
Average time cost (second)	904.69 (0.25 hour)		12435.51 (3.45 hour)	
	Probabilistic Fireflies K-means		Greedy Probabilistic Fireflies K-means	
	Experiment 1	Experiment 2	Experiment 1	Experiment 2
Sum of squared error	8534301.6	8534467.4	8534273.3	8534258.1
Time cost (second)	20886.50	22219.12	13980.91	12465.27
Average sum of squared error	8534384.5		8534265.7	
Average time cost (second)	21552.81 (5.99 hour)		13223.09 (3.67 hour)	

it is reasonable to use PFK as an improvement for the k-means. FK, however, fail to significantly outperform k-means on its cluster result. The time cost of FK and GPFK is basically at the same level. Although the time cost of GPFK is greater than k-means, accuracy and stability are more of concern, especially when dealing with big data.

4.3 Discussion

From the experiments above, we can see that the key to reaching a high accuracy when improving k-means algorithm is to maintain the stability. To do so, optimization algorithms need to provide k-means consistent proper starting centroids by pre-cluster the data. K-means algorithm is an accurate cluster algorithm with

the decent starting position.

It is also critical to choose proper parameters for the algorithms. In the experiments, the algorithms, FK, PFK, and GPFK have distinct characters, so the parameters need to be modified for each one. The movement of fireflies is reflected in updating the percentage values representing an index, so the α value tend to be smaller than ordinary fireflies algorithm. For GPFK, on the other hand, the α tend to be higher since all fireflies move from one point and want to make more obvious changes. Similarly, γ values appear to be much lower in PFK and GPFK than FK, since the percentage values are smaller and they are preferable to move without resistance. Instead, in FK and ordinary fireflies algorithms, the optimized value may drop too fast and fall into local optima if the γ value is too small.

CHAPTER 5

FUTURE WORK

Although PFK and GPFK can successfully improve the stability of k-means, time cost is the major issue because of the bigger sized data structure of Firefly. Therefore, it is critical to improve the algorithm and make it more time efficient. There are several ideas we can work on in the future to make it run faster and be more practical.

First, we can evaluate if it is possible to let Firefly only change one percentage value, instead of changing all k values when moving on one index assignment, to save a lot of time and remaining stable. Since it is the highest percentage value in a cluster channel that decides the index number, it could be practical to make darker firefly learn from the highest cause. However, we still need to execute more experiments to observe the effect of this change.

In addition, we can work on the random static segment mechanism to make it more intelligent. The algorithm should be able to automatically change the size and directions of the segment that is not moving based on the sum of squared error, the results of previous iteration and other related factors. In this way, the algorithm can reach a better ability and alter fewer parts in final cluster array at the same time. The algorithm can run faster when the calculation amount is reduced. Right now, the random static segment is only decided by the time of iteration, which is too simple and straightforward.

Moreover, the choice of parameters for fireflies algorithm is a key to its performance. It would be a huge improvement if we can let the algorithm decide the parameters by itself based on the type and size of data set, the form of attributes and

so on. The performance of GPFK can reach a new level if the automation of choosing random static segment and parameters can be realized. These are the aspects that we need to focus on in the future.

CHAPTER 6

CONCLUSION

In this project, we first implemented the fireflies k-means hybrid cluster algorithm on Spark platform and evaluated the performance of the original algorithm by both small and big data experiments. After that, we made several improvements to the original FK algorithm and presented two new approaches, probabilistic fireflies k-means algorithm and greedy probabilistic fireflies k-means algorithm. Per the experiment results between the three algorithms, a safe conclusion can be drawn that GPFK can outperform original FK by being more stable and accurate. The time cost of GPFK is at the same level with FK.

The main improvement we performed on the algorithm includes the new data structure of Firefly, the random value fluctuation, and random static segment. These improvements help GPFK to represent cluster assignment in a more reasonable way and optimize the movement strategy of fireflies. These innovations make GPFK an ideal and practical approach for data cluster jobs when stability and accuracy are in priority.

In the future, we will focus on reducing the time cost of GPFK by changing the movement strategy and the choosing of the static segment. In addition, the automation of parameter selection is also an aspect to work on.

REFERENCES

[1] de Oliveira, G. V., and Naldi, M. C. Scalable fast evolutionary k-means clustering. In *Intelligent Systems (BRACIS), 2015 Brazilian Conference on* (2015), IEEE, pp. 74–79.

[2] Hassanzadeh, T., and Meybodi, M. R. A new hybrid approach for data clustering using firefly algorithm and k-means. In *Artificial Intelligence and Signal Processing (AISP), 2012 16th CSI International Symposium on* (2012), IEEE, pp. 007–011.

[3] Holland, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[4] Hu, J., Wang, C., Liu, C., and Ye, Z. Improved k-means algorithm based on hybrid fruit fly optimization and differential evolution. In *2017 12th International Conference on Computer Science and Education (ICCSE)* (Aug 2017), pp. 464–467.

[5] Kennedy, J., and Eberhart, R. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on* (Nov 1995), vol. 4, pp. 1942–1948 vol.4.

[6] Krishna, K., and Murty, M. N. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 29*, 3 (1999), 433–439.

[7] Latiff, N. M. A., Malik, N. N. N. A., and Idoumghar, L. Hybrid backtracking search optimization algorithm and k-means for clustering in wireless sensor networks. In *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)* (Aug 2016), pp. 558–564.

[8] Mizuno, K., Takamatsu, S., Shimoyama, T., and Nishihara, S. Fireflies can find groups for data clustering. In *Industrial Technology (ICIT), 2016 IEEE International Conference on* (2016), IEEE, pp. 746–751.

[9] Xiong, C., Hua, Z., Lv, K., and Li, X. An improved k-means text clustering algorithm by optimizing initial cluster centers. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD)* (Nov 2016), pp. 265–268.

[10] Yang, X.-S. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.