A Quantum-inspired Genetic Algorithm for Data Clustering

Jing Xiao, YuPing Yan, Ying Lin, Ling Yuan and Jun Zhang

***Abstract*—The conventional K-Means clustering algorithm must know the number of clusters in advance and the clustering result is sensitive to the selection of the initial cluster centroids. The sensitivity may make the algorithm converge to the local optima. This paper proposes an improved K-Means clustering algorithm based on Quantum-inspired Genetic Algorithm (KMQGA). In KMQGA, Q-bit based representation is employed for exploration and exploitation in discrete 0-1 hyperspace by using rotation operation of quantum gate as well as three genetic algorithm operations (Selection, Crossover and Mutation) of Q-bit. Without knowing the exact number of clusters beforehand, the KMQGA can get the optimal number of clusters as well as providing the optimal cluster centroids after several iterations of the four operations (Selection, Crossover, Mutation, and Rotation). The simulated datasets and the real datasets are used to validate KMQGA and to compare KMQGA with an improved K-Means clustering algorithm based on the famous Variable string length Genetic Algorithm (KMVGA) respectively. The experimental results show that KMQGA is promising and the effectiveness and the search quality of KMQGA is better than those of KMVGA.**

I. Introduction

TEXT clustering plays an important role in many research areas, such as information retrieval, text summarization, topic detection, and data mining. Generally speaking, conventional text clustering algorithms can be grouped into two main categories, namely hierarchical clustering algorithms and partitional clustering algorithms. A hierarchical clustering algorithm outputs a dendrogram, which is a tree structure showing a sequence of clusterings with each clustering being a partition of the dataset [1]. Unlike the hierarchical clustering algorithm, the partitional clustering algorithms partition the data set into a number of clusters, and the output is only a single partition of the data set. The majority of partitional clustering algorithms obtain the partition through the maximization or minimization of some criterion function. Recent researches show that the partitional clustering algorithms are well suited for clustering a large dataset due to their relatively low computational requirements [2]. And the time complexity of the partitional algorithms is almost linear, which makes them widely used [3].

Manuscript received Dec. 10, 2007.

Jing Xiao is with the Department of Computer Science, Sun Yat-Sen University, GuangZhou, China. Postcode: 510275. Phone: 8620-31457668 (e-mail: xiaoj2@mail.sysu.edu.cn).

YuPing Yan is with the School of Software, Sun Yat-Sen University, GuangZhou, China. (e-mail: yuson_yan@163.com).

Ying Lin is with the Department of Computer Science, Sun Yat-Sen University, GuangZhou, China. (email: issly@mail.sysu.edu.cn).

Ling Yuan is with the School of Computing, National University of Singapore, Singapore. (email: yuanling@comp.nus.edu.sg)

Jun Zhang is with the Department of Computer Science, Sun Yat-Sen University, GuangZhou, China. (email: junzhang@ieee.org).

Among the partitional clustering algorithms, the most famous one is K-Means clustering algorithm [4]. K-Means clustering algorithm must know the exact number of final clusters (K) and a criterion function to evaluate whether the final partition is good. Then the algorithm randomly generates K initial clusters centroids. After several iterations of this algorithm, text data can be classified into a certain cluster by the criterion function, which make text data is similar to each other in the same cluster. However, the traditional K-Means clustering algorithm has two drawbacks. One is the number of clusters must be known in advance, and the other is that the result is sensitive to the selection of initial cluster centroids and this may make the algorithm converge to the local optima. Different dataset has different number of clusters, which is difficult to know beforehand, and the initial clusters centroids are selected randomly, which will make the algorithm converge to the different local optima. Therefore, the conventional K-Means clustering algorithm does not work well in practice.

Due to these two drawbacks, more and more researches on K-Means clustering algorithm are to find the optimal number of clusters (K) and the best initial clusters centroids. Since the Q-bit representation in quantum computing can represent linear superposition of states probabilistically, it has a better characteristic of population diversity than other representations. In this paper, we propose an improved K-Means clustering algorithm based on Quantum-inspired Genetic Algorithm (KMQGA). KMQGA use Q-bit [5] as its chromosome's representation and Davies-Bouldin Rule Index [6] as criterion function. Every chromosome in KMQGA denotes clusters centroids of a partition. Different chromosomes can have different string length of Q-bits, that is, different chromosomes denote different partitions. Before performing the GA operations (Selection, Crossover, and Mutation) and the Quantum operation (Rotation), the Q-bit representation firstly have to be encoded into the binary representation, and then the real-coded representation. When the GA operations and the Quantum operation are conducted during the iterations of KMQGA, the chromosome may change its string length, that is, the clusters centroids of a partition are changed. After several iterations of KMQGA, the improved algorithm can find the optimal number of clusters as well as the initial clusters centroids, due to the string length variation of chromosome. The simulated datasets and the real datasets are used to verify the effectiveness of KMQGA and to compare KMQGA with a K-Means clustering algorithm based on the famous Variable string length Genetic Algorithm (KMVGA) [7, 8] respectively.

The rest part of this paper is organized as follows: Section Ⅱ introduces some related work about optimization problems

978-1-4244-1823-7/08/$25.00©2008 IEEE

using the concept of quantum computing. Section Ⅲ briefly introduces the quantum computing .The details of KMQGA are described in Section Ⅳ. Experiments and results are presented in Section Ⅴ. We conclude this paper in Section Ⅵ.

II. Related Work

Quantum computing is a research area that includes concepts like quantum-mechanical computers and quantum algorithms. So far, many efforts on quantum computing have progressed actively due to its superiority to classical computers on many aspects. Also, there are some well-known quantum algorithms, such as Grover's database search algorithm [9] and Shor's quantum factoring algorithm [10]. During the past two decades, evolutionary computing has attracted more and more researchers and played an important part in the area of optimization. Recently, the work of merging quantum computing and evolutionary computing has stimulated the studies of its theories and applications. The most important work to classical computer was done by Han and Kim [11]. They proposed a quantum evolutionary algorithm which is used to solve combinational problem on classical electronic computer. In [12], a quantum-inspired genetic algorithm is proposed for flow shop scheduling problem with a single objective. And in [13], a hybrid quantum-inspired genetic algorithm is proposed for multi-objective flow shop scheduling problem. The concept of quantum computing algorithms are also applied to some other applications, such as solving the travelling salesman problem [14], image segmentation [15], and face detection [16].

III. Quantum Computing

Before describing KMQGA, we introduce the quantum computing briefly. Unlike the two-state (0/1) representation in conventional computing, the smallest information representation in quantum computing is called quantum bit (Q-bit) [17]. The state of Q-bit may be "0" state, "1" state or any superposition of the two. So the state of Q-bit can be represented in formula ①:

$$|\psi> = \alpha|0> + \beta|1> \qquad ①$$

Where α and β are complex numbers that specify the probability amplitudes of the corresponding states. Thus, $|\alpha|^2$ and $|\beta|^2$ denote probabilities that the Q-bit will be found in the "0" state and the "1" state, respectively. Normalization of the state to the unity guarantees in formula ②:

$$|\alpha|^2 + |\beta|^2 = 1 \qquad ②$$

Thus, a Q-bit can be represented as the linear superposition of the two conventional binary genes (0 and 1). A Q-bit individual as a string of m Q-bits is defined in formula ③:

$$\begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix} \begin{matrix} \alpha_2 \\ \beta_2 \end{matrix} \begin{matrix} \ldots\ldots \\ \ldots\ldots \end{matrix} \begin{bmatrix} \alpha_m \\ \beta_m \end{bmatrix} \qquad ③$$

If there is a string of m Q-bits, then the string can represent 2^m states at the same time. The state of a Q-bit can be changed by the operation with a quantum gate, such as the NOT gate, the rotation gate, etc. However, the superposition of "0" state and the "1" state must collapse to a single state in the action of observing a quantum state, that is, a quantum state have to be the "0" state or "1" state. In the evolutionary computing, Q-bit representation has a better characteristic of population diversity than other representations, because it can represent linear superposition of states probabilistically. Examples of Q-bit representation can be seen in [11].

Inspired by the concept of quantum computing and the representation of Q-bit, KMQGA is designed with this novel Q-bit representation, and Quantum Rotation operation is one of the various operations in the algorithm.

IV. KMQGA

In this section, we first propose the overall algorithm of KMQGA. Then the chromosome representation and the fitness function of this algorithm will be presented. Last, four main operations and an accessional operation (GA selection, GA crossover, GA mutation, Quantum rotation operation and Quantum catastrophe operation) will be interpreted.

A. The Overall Flowchart of KMQGA

KMQGA uses Davies-Bouldin (DB) Rule Index as its criterion function, which is also called fitness function, to evaluate the partition of a dataset is good or not. The overall flowchart of KMQGA is shown in Fig. 1.The algorithm begins with initializing the population randomly. Each chromosome (also called individual) in the population denotes a certain partition of a dataset. The chromosome is represented by the Q-bit representation at first. Then the Q-bit string will be collapsed into a certain state, which is a binary string. After the collapse operation, there is another operation to change this binary string into real-coded string. Each real number in the real-coded string denotes a pattern of the dataset. Then the algorithm run conventional K-Means clustering algorithm on each chromosome just one time and evaluates each chromosome by the DB rule index fitness function and calculates value and the selected probability in the later GA selection operation. The performance of a certain partition of a dataset is evaluated by the fitness value. Based on the selected probability, the algorithm can produce a new population through roulette selection and the elite selection. The GA crossover operation affects each chromosome in terms of the crossover probability. In the process of GA crossover operation, the length of each chromosome may be changed. Due to this change, the partition denoted by the chromosome is changed accordingly, and after several iterations of this algorithm, the better chromosome may be shown up. Then the GA mutation and Quantum rotation operation are performed, both of which may make the searching space of KMQGA more diversified. In order to avoid the degeneration of the chromosome and prematurity of the algorithm, KMQGA uses elite selection operation, adaptive probabilities of crossover operation and mutation operation, quantum catastrophe operation. The overall procedure of KMQGA is summarized as follows:

(a) Randomly generating an initial population using Q-bit representation;

(b) Collapsing the Q-bit representation into the binary representation, and then the real-coded representation. Using the DB fitness function to evaluate each chromosome;

(c) If the stopping criterion is satisfied, then output the best result; otherwise save the best chromosome and go to the following steps;
(d) If the prematurity criterion is satisfied, then go to step (f); otherwise go to step (e);
(e) Adjusting the probabilities of crossover and mutation as $\mathcal{P}_c$ and $\mathcal{P}_m$ respectively. Then go to step (g);
(f) Adjusting the probabilities of crossover and mutation as $\mathcal{P}_{cc}$ and $\mathcal{P}_{mm}$ respectively. Then go to step (g);
(g) Performing selection operation, crossover operation and mutation operation;
(h) If the catastrophe criterion is satisfied, then go on step (j); otherwise go to step (g);
(i) Look up table of rotation angle, and then perform rotation operation. Go back to step (b);
(j) Perform catastrophe operation. Go back to step (b).

KMQGA sets a maximal iteration number $\mathcal{N}_{\max}$ as its stopping criterion, which should be as big as possible in order to find the optimal solution. The prematurity criterion is satisfied when the best chromosome saved by the algorithm does not change after $n_{\max}$ iterations, where $n_{\max} < \mathcal{N}_{\max}$. That is to say, the best chromosome in current iteration is the same as the best one which is $n_{\max}$ iterations ago, then probabilities of the crossover ($\mathcal{P}_c$) and the mutation ($\mathcal{P}_m$) should be adjusted to $\mathcal{P}_{cc}$ and $\mathcal{P}_{mm}$ respectively for the sake of jumping out of the local trap. Since the larger probability of the crossover can exchange more information among a couple of chromosomes, and the larger probability of the mutation can enlarge the diversity of the population, the algorithm adjusts $\mathcal{P}_c$ and $\mathcal{P}_m$ to larger probabilities ($\mathcal{P}_{cc}$and $\mathcal{P}_{mm}$).The catastrophe criterion [12] is similar to the prematurity criterion and we think it is trapped in a local optima if the best solution does not vary in a number of consecutive generations. Then the best solution is reserved and the others will be replaced by randomly generated solutions. Another number $m_{\max}$ needs to be set to judge whether the catastrophe criterion is met, where $m_{\max} < \mathcal{N}_{\max}$.

The four probabilities ($\mathcal{P}_c, \mathcal{P}_m, \mathcal{P}_{cc}, \mathcal{P}_{mm}$) are real numbers less than 1.0, where $\mathcal{P}_{cc} > \mathcal{P}_c$, $\mathcal{P}_{mm} > \mathcal{P}_m$, $\mathcal{P}_{cc} = \mathrm{random}(\mathcal{P}_c, 1)$ and $\mathcal{P}_{mm} = \mathrm{random}(\mathcal{P}_m, 2 \times \mathcal{P}_m)$. Here, we set $\mathcal{P}_m < 0.5$.and $\mathrm{random}(a, b)$ is a function that generates a real number between a and b.

B. *Representation of chromosome in KMQGA*

Inspired by the concept of quantum computing, the Q-bit representation and the representation of Modified Variable string length Genetic Algorithm (MVGA) [18]; KMQGA employs a modified variable Q-bit string length representation. Before coding the chromosomes, the amount of texts to be clustered in dataset N should be known and the range of K (number of clusters) $[\mathrm{K}_{\min}, \mathrm{K}_{\max}]$ should be defined first, where $\mathrm{K}_{\min} \geq 2$ and $\mathrm{K}_{\max} \leq \mathrm{N}$. As long as the $\mathrm{K}_{\max}$ is not less than the exact number of clusters, the algorithm can get the optimal solution. However, if $\mathrm{K}_{\max}$is much greater than the exact number of clusters, the cost of both the computation time and the storage space will be high. Research shows that the optimal K is not larger than $\sqrt{\mathrm{N}}$[19]. Therefore, the range of K is set to $\left[2,\ \sqrt{\mathrm{N}} + 1\right]$.

Knowing the amount N, we are also aware of the number of binary string length (B) to denote a text pattern ID, that is, $2^{\mathrm{B}-1} < N \leq 2^{\mathrm{B}}$. For example, if the number of texts is 178, then $\mathrm{B} = 8$.

When coding the chromosome, a number k is randomly generated first, where $k \in [\mathrm{K}_{\min}, \mathrm{K}_{\max}]$. Thus, the Q-bit string length of this chromosome is $k \times \mathrm{B}$. The chromosome representation is presented as follows:

$$\mathcal{R}:\ \begin{bmatrix} \alpha_1 & | & \alpha_2 & | & \cdots\cdots & | & \alpha_{k\times\mathrm{B}} \\ \beta_1 & | & \beta_2 & | & \cdots\cdots & | & \beta_{k\times\mathrm{B}} \end{bmatrix}$$

Then the Q-bit string will be collapsed into a certain state, which is a binary string. That is, randomly generating a float number η, where $\eta \in (0,1)$. If $\eta \leq |\alpha_i|^2$, then the corresponding binary state is 0, otherwise is 1.

After the collapse operation, there is another operation to change this binary string into real-coded string. The algorithm converts every B binary string into a real number. Each real number in the real-coded string denotes a pattern of the dataset.

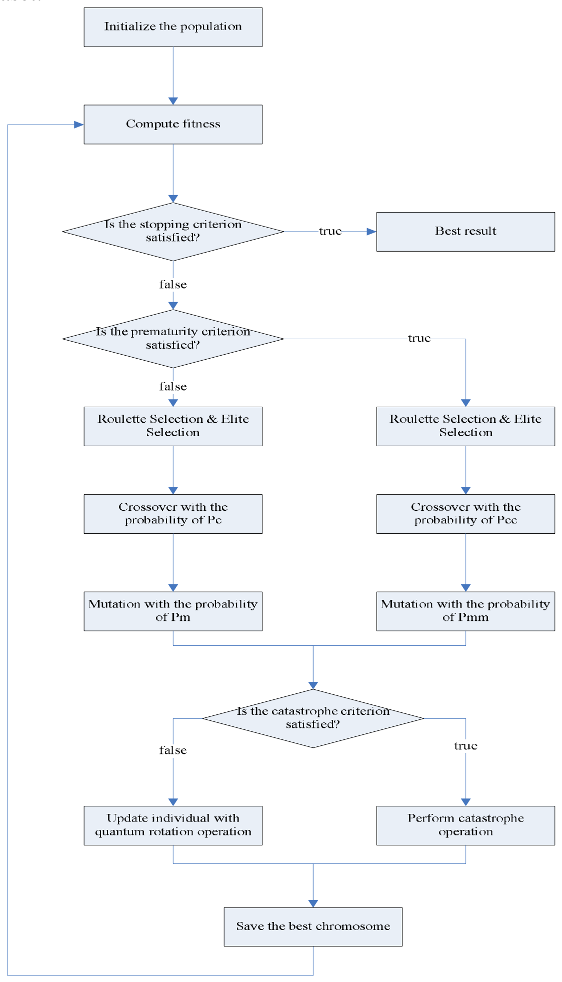
Initialize the population
Compute fitness
Is the stopping criterion satisfied?
true
Best result
false
Is the prematurity criterion satisfied?
true
false
Roulette Selection & Elite Selection
Roulette Selection & Elite Selection
Crossover with the probability of Pc
Crossover with the probability of Pcc
Mutation with the probability of Pm
Mutation with the probability of Pmm
Is the catastrophe criterion satisfied?
false
true
Update individual with quantum rotation operation
Perform catastrophe operation
Save the best chromosome

Fig. 1 The Overall Flowchart of KMQGA

C. Fitness function of KMQGA

Evaluating a certain partition should use a criterion function (also called fitness function). A good fitness function can give a high value to a good partition solution and give a low value to a bad partition. There are many fitness functions to evaluate clustering nowadays, such as Euclidean distance, Davies-Bouldin Rule Index, Dunn index [20], and Turi [21], S_DBW [22], CDBW [23].

Since the DB rule index can evaluate the distance of intra-cluster and the distance of inter-cluster, KMQGA employs DB as its fitness function. We describe DB as follows [24]:

$$S_i = \frac{1}{N_i}\sum_{X \in G_i} \|X - Z_i\|$$

$$Z_i = \frac{1}{N_i}\sum_{X \in G_i} X$$

Where G_i is the set the ith cluster, N_i is the amount of texts in the cluster G_i, Z_i is the centroids of G_i, $\|X - Z_i\|$ is the Euclidean distance between text pattern X and Z_i, S_i means the average cohesion of G_i.

$$d_{i,j} = \|Z_i - Z_j\|$$

Where $d_{i,j}$ denotes the distance between Z_i and Z_j.

$$R_i = \max_{j,j \neq i}\left\{\frac{S_i + S_j}{d_{i,j}}\right\}, i \neq j$$

$$DB = \frac{1}{K}\sum_{i=1}^{K} R_i$$

The smaller the DB value, the better the partition. So we let $\frac{1}{DB}$ be the KMQGA's fitness function.

$$\mathcal{F} = \frac{1}{DB}$$

Since this Q-bit representation cannot denote the expression of the text pattern ID directly, the Q-bit representation has to be collapsed to binary string representation first. A Q-bit string $\mathcal{R}$ with length $k \times B$ is firstly collapsed to a certain state (0-1 state) according to the probability amplitudes of the individual. For $i = 1,2,3,\ldots\ldots, k \times B$, we generate a random number η between $[0, 1]$. If α_i of chromosome $\mathcal{R}$ satisfies $|\alpha_i|^2 > \eta$, then set $\mathcal{R}_i$ to 0, otherwise set it to 1. After the binary string is constructed, we encode every $\mathcal{B}$-length binary string into a real number, which is a text pattern ID. That is to say, KMQGA first change the Q-bit string to binary string, and then convert it to real number. Thus, a Q-bit string with length $k \times B$ denotes a partition with k centroids. By doing this, the DB fitness function can evaluate every partition.

D. Selection Operation in KMQGA

Roulette selection and elite selection are employed in KMQGA. Roulette selection runs according to the selected probability. Usually, the chromosome with high fitness value will be chosen to the next iteration with a high probability. However, in order to avoid the optimum chromosome with good fitness value is not selected occasionally; KMQGA performs elite selection after the roulette. Elite selection guarantees that the best chromosome in a certain generation will not be lost in the evolutionary process. The basic idea of elite selection is: if a certain individual in the former generation is better than the best individual in the current generation, then some individuals in the current generation will be replaced by the better ones in the former generation. The pseudo code of elite selection is expressed as follows:

```
For the $t$th generation, the population is
$P(t) = \{P(t,1), P(t,2), \ldots, P(t,N)\}$;
and $F_{max}(t)$ is the best individual in the $t$th generation.
$F_{max}(t) = \max\{F(P(t,1)), F(P(t,2)), \ldots, F(P(t,N))\}$;
$F_{max}(t+1) = \max\{F(P(t+1,1)), F(P(t+1,2)), \ldots, F(P(t+1,N))\}$;
if $F_{max}(t) > F_{max}(t+1)$ then
    replicate
        $\{P(t,k) \mid F(P(t,k)) > F_{max}(t+1), P(t,k) \in P(t)\}$;
    replace randomly
    $\{P(t+1,j) \in P(t+1)\}$ with $\{P(t,k) \mid F(P(t,k)) > F_{max}(t+1), P(t,k) \in P(t)\}$;
end if
```

E. Crossover Operation in KMQGA

KMQGA uses a special crossover operation which can change the length of parent-chromosomes. For each chromosome, the crossover point is randomly chosen according to its own string length. For example, there are two chromosomes ($\mathcal{R}_1$ and $\mathcal{R}_2$) with length 8 and 5 respectively:

$$\mathcal{R}_1: \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & \beta_8 \end{bmatrix}$$

$$\mathcal{R}_2: \begin{bmatrix} \alpha'_1 & \alpha'_2 & \alpha'_3 & \alpha'_4 & \alpha'_5 \\ \beta'_1 & \beta'_2 & \beta'_3 & \beta'_4 & \beta'_5 \end{bmatrix}$$

A random integer between 1 and its length is generated as the crossover point for each chromosome. For example, the crossover point of $\mathcal{R}_1$ and $\mathcal{R}_2$ are 6 and 2 respectively. Then, the new chromosomes ($\mathcal{R}'_1$ and $\mathcal{R}'_2$) are shown as follows:

$$\mathcal{R}'_1: \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha'_3 & \alpha'_4 & \alpha'_5 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta'_3 & \beta'_4 & \beta'_5 \end{bmatrix}$$

$$\mathcal{R}'_2: \begin{bmatrix} \alpha'_1 & \alpha'_2 & \alpha_7 & \alpha_8 \\ \beta'_1 & \beta'_2 & \beta_7 & \beta_8 \end{bmatrix}$$

Now, we can see both of the chromosomes' lengths are changed. Due to this change, the partition solution denoted by the chromosome is changed accordingly. Thus, the search space is larger and the optimal solution could be found.

F. Mutation Operation in KMQGA

For the diversity, mutation is employed. Based on the mutation probability, mutation point is generated randomly according to the chromosome's length. For example, there is a chromosome ($\mathcal{R}_3$) with the length of 7.

$$\mathcal{R}_3: \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 \end{bmatrix}$$

The mutation point is a number between 1 and its string length 7. If $\mathcal{R}_3$ mutation point is 5, then KMQGA change the position of α_5 and β_5. The new chromosome is $\mathcal{R}'_3$ is shown as follows:

$$\mathcal{R}'_3: \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \beta_5 & \alpha_6 & \alpha_7 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 & \alpha_5 & \beta_6 & \beta_7 \end{bmatrix}$$

G. Rotation Operation in KMQGA

A rotation gate $U(\theta)$ is employed to update a Q-bit individual as follows:

$$\begin{bmatrix} \alpha'_i \\ \beta'_i \end{bmatrix} = U(\theta_i) \times \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \times \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \qquad ④$$

where $\begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}$ is the ith Q-bit and θ_i is the rotation angle of each Q-bit toward either 0 or 1 state depending on its sign.

The rotation operation used in KMQGA is to adjust the probability amplitudes of each Q-bit. According to the rotation operation in ④, a quantum-gate $U(\theta_i)$ is a function of $\theta_i = s(\alpha_i, \beta_i) \times \Delta\theta_i$, where $s(\alpha_i, \beta_i)$ is the sign of θ_i which determines the direction and $\Delta\theta_i$ is the magnitude of rotation angle [13]. The lookup table of $\Delta\theta_i$ is shown in Table Ⅰ, where b_i and r_i are the ith bits of the best solution b and a binary solution of r, respectively. In particular, if the length of b is not the same as the length of r, then additional correcting performance will be employed. In this paper, we increase/delete the r′s Q-bits randomly if the length of r is less/more than the length of b. Because b is the best solution in KMQGA, the use of quantum-gate rotation is to emphasize the searching direction toward b.

TABLE Ⅰ
THE LOOKUP TABLE OF ROTATION ANGLE

r_i	b_i	$\mathcal{F}(r) < \mathcal{F}(b)$	$\Delta\theta_i$	$s(\alpha_i, \beta_i)$			
				$\alpha_i\beta_i > 0$	$\alpha_i\beta_i < 0$	$\alpha_i = 0$	$\beta_i = 0$
0	0	False	0	0	0	0	0
0	0	True	0	0	0	0	0
0	1	False	0	0	0	0	0
0	1	True	0.05л	-1	+1	±1	0
1	0	False	0.01л	-1	+1	±1	0
1	0	True	0.025л	+1	-1	0	±1
1	1	False	0.005л	+1	-1	0	±1
1	1	True	0.025л	+1	-1	0	±1

H. Catastrophe operation in KMQGA

To avoid premature convergence, a catastrophe operation [12] is used in KMQGA. We consider that it is trapped in local optima if the best solution does not change in certain consecutive generations. Then the best solution is reserved and the others will be replaced by solutions randomly generated.

V. EXPERIMENTAL EVALUATION OF KMQGA

In this section, we will test the performance of the proposed KMQGA. First, we test KMQGA's correctness using three simulated datasets. Then four datasets from famous UCI machine learning repository [25] are carried out to compare the performance and the effectiveness of KMQGA with those of KMVGA. In both simulated data and real data experiments, we set the size of population as 100, $\mathcal{P}_c = 0.9$, $\mathcal{P}_m = 0.01$, $n_{\max} = 15$, $m_{\max} = 25$. We set $\mathcal{N}_{\max} = 100$ and $\mathcal{N}_{\max} = 300$ in simulated data and real data respectively.

A. Simulated datasets

We generate three simulated datasets (sds1, sds2, sds3), randomly. There are three, four and eight clusters in sds1, sds2, sds3 respectively. Each cluster in simulated datasets has 100 data vectors. To get the direct vision from the coordinate, we define each data vector as two dimensions. Details of clusters' ranges of sds1, sds2 and sds3 are given in Table Ⅱ, Table Ⅲ, and Table Ⅳ respectively.

TABLE Ⅱ
DETAILS OF CLUSTERS' RANGES IN SDS1 DATASET

Coordinate	*Range of sds1*		
	Cluster1	Cluster2	Cluster3
X	[0,20]	[40,60]	[80,100]
Y	[0,20]	[40,60]	[80,100]

TABLE Ⅲ
DETAILS OF CLUSTERS' RANGES IN SDS2 DATASET

Coordinate	*Range of sds2*			
	Cluster1	Cluster2	Cluster3	Cluster4
X	[0,20]	[40,60]	[80,100]	[0,20]
Y	[0,20]	[40,60]	[80,100]	[80,100]

TABLE Ⅳ
DETAILS OF CLUSTERS' RANGES IN SDS3 DATASET

Coordinate	*Range of sds3*							
	Cluster1	Cluster2	Cluster3	Cluster4	Cluster5	Cluster6	Cluster7	Cluster8
X	[0,20]	[40,60]	[80,100]	[80,100]	[0,20]	[180,200]	[180,200]	[180,200]
Y	[0,20]	[40,60]	[80,100]	[0,20]	[180,200]	[0,20]	[80,100]	[180,200]

After a certain range is given, these data vectors are generated in uniform probability distribution with the given range. We run KMQGA with the three simulated datasets for ten times, and each time we get the number of clusters (K). Details of the results are given in Table Ⅴ:

TABLE Ⅴ
DETAILS OF RESULTS WHICH KMQGA RUNS

Dataset	*K in Dataset*	*KMQGA results*										
		1	2	3	4	5	6	7	8	9	10	Average
Sds1	3	3	3	3	3	3	3	3	3	3	3	3
Sds2	4	4	4	4	4	4	4	4	4	4	4	4
Sds3	8	8	8	8	8	8	8	8	8	8	8	8

KMQGA also obtains the best initial clusters centroids of each dataset. One of the ten runs experiments' results are shown as follows. Fig. 2, Fig. 3, Fig. 4 are the partitions solution of sds1, sds2, sds3 respectively.

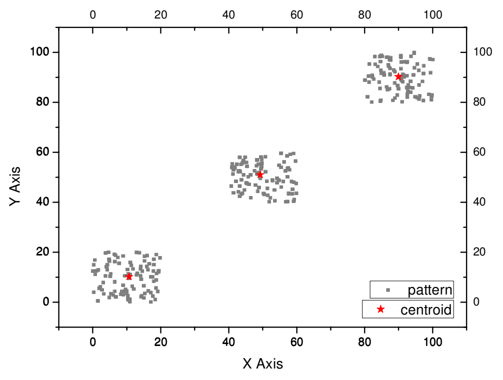

Fig. 2 sds1 dataset

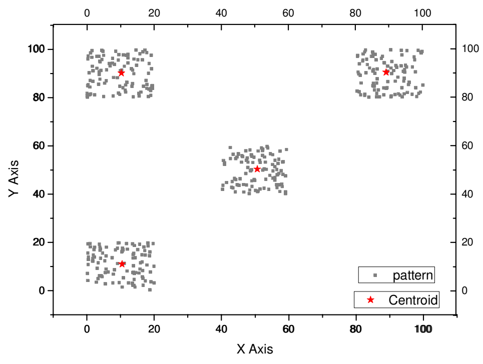

Fig. 3 sds2 dataset

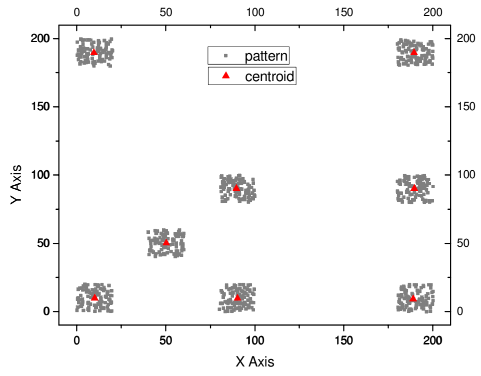

Fig. 4 sds3 dataset

We can see from Table Ⅴ that KMQGA can get the exact K of each dataset in all 10 experiments. In Fig. 2, Fig. 3, Fig. 4, the initial clusters centroids obtained by KMQGA are reasonable, since each centroids are almost in the center of corresponding data vectors from a visual point of view. Thus, the correctness of KMQGA is proved by the simulated datasets in our experiment.

B. *Real datasets (UCI datasets)*

The four real datasets from UCI machine learning repository are Glass, Wine, SPECTF-Heart and Iris. The details of the four datasets, which can be found in the .names file of every dataset's fold [25], are summarized in Table Ⅵ.

TABLE Ⅵ
DETAILS OF UCI REAL DATASETS

Dataset	*Number of Instances*	*Number of Attributes*	*Number of clusters*
Glass	214	9	2
Wine	178	13	3
SPECTF-Heart	187	44	2
Iris	150	4	3

For each dataset, KMQGA and KMVGA run 10 times respectively. And the average results are shown as follows in Table Ⅶ:

TABLE Ⅶ
COMPARING RESULTS BETWEEN KMQGA AND KMVGA

Dataset	*KMQGA*		*KMVGA*	
	Number of clusters	Best fitness value	Number of clusters	Best fitness value
Glass	2	63.5291	2	36.0576
Wine	3	10.6751	12	4.0309
SPECTF-Heart	2	10.9353	2	6.816
Iris	3	13.9861	6	4.8024

We can see from the results from Table Ⅶ that, for all the four real datasets, KMQGA obtains the exact K. But the KMVGA just gets the optimal result in Glass dataset and SPECTF-Heart dataset and cannot get the exact K in another two datasets. For the best fitness value, KMQGA gets the higher value than KMVGA does. In Glass dataset, KMQGA's best fitness value is 1.7619 times higher than KMVGA's. In Wine dataset, KMQGA's best fitness value is 2.6483 times higher than KMVGA's. In SPECTF-Heart dataset, KMQGA's best fitness value is 1.6044 times higher than KMVGA's. In Iris dataset, KMQGA's best fitness value is 2.9123 times higher than KMVGA's. Therefore, we can say that KMQGA is better than KMVGA in real datasets in our experiments.

From the results of both our simulated datasets and the UCI real datasets, the correctness of KMQGA is proved. And the performance, the effectiveness and the stability of KMQGA are better than those of KMVGA.

VI. CONCLUSION

In this paper, we propose an improved K-Means clustering algorithm based on Quantum-inspired Genetic Algorithm (KMQGA). This algorithm employs Q-bit representation and the concept of quantum computing. Four main operations and an accessional operation (GA selection, GA crossover, GA mutation, Quantum rotation operation and Quantum catastrophe operation) are performed to search the optimal partition solution of a certain dataset. And the Q-bit string length can be changed in crossover operation. Due to this change, the partition denoted by a chromosome is changed, too. Thus, the algorithm's searching space is large enough to get the optimal solution after several iterations of evolution. The simulated datasets in our experiment proved the correctness of KMQGA, and the UCI real datasets are performed to compare the difference between KMQGA and KMVGA in the performance and the effectiveness. The experiment results show that KMQGA is better than the KMVGA. Our future work is to investigate how to explore the search space using small number of individuals (even using only one individual).

REFERENCES

[1] Leung Y, Zhang J, and Xu Z, "Clustering by space-space filtering," *IEEE Trans Pattern Anal Mach* Intell 22(12): 1396-1410, 2000

[2] M. Steinbach, G. Karypis, and V. Kumar, "A Comparison of Document Clustering Techniques," in *TextMining Workshop, KDD*, 2000

[3] Ajith Abraham, Swagatam Das, and Amit Konar, "Document Clustering Using Differential Evolution". in *2006 IEEE Congress on Evolutionary Computation*, 2006

[4] J. A. Hartigan, “Clustering Algorithms,” John Wiley and Sons, Inc., New York, NY, 1975

[5] Narayanan, A. and Moore, M, “Quantum-inspired genetic algorithms,” in*1996 IEEE Congress on Evolutionary Computation*, 1996

[6] Davies Bouldin, “A cluster separation measure,” *IEEE Trans Pattern Anal Mach* Intell 1(2), 1979

[7] Sanghamitra Bandyopadhyay and Ujjwal Mauilk, “Nonparametric Genetic Clustering: Comparison of Validity Indices,” *IEEE Transactions on System, Man*, and Cybernetics-Part C Applications and Reviews, Vol. 31, No. 1,2001

[8] Ujjwal Mauilk and Sanghamitra Bandyopadhyay, “Performance Evaluation of Some Clustering Algorithms and Validity Indices,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 12, 2002.

[9] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proc. 28th ACM Symp. Theory Comput.*, Philadelphia, PA, pp. 212–221, 1996.

[10] P. W. Shor, “Algorithms for quantum computation: Discrete logarithms and factoring,” in *Proc. 35th Symp. Found. Comput.* Sci., Los Alamitos,CA, pp. 20–22, 1994.

[11] K. H. Han and J. H. Kim, “Quantum-inspired evolutionary algorithm for a class of combinatorial optimization,” *IEEE Trans. Evol. Comput.*, vol. 6,no. 6, pp. 580–593, Dec. 2002.

[12] L. Wang, H. Wu, F. Tang, and D. Z. Zheng, “A hybrid quantum-inspired genetic algorithm for flow shop scheduling,” *Lecture Notes in Computer Science*, vol. 3645. Berlin, Germany: Springer-Verlag, pp. 636–644, 2005

[13] Bin-Bin Li and Ling Wang, “A Hybrid Quantum-Inspired Genetic Algorithm for Multiobjective Flow Shop Scheduling,” *IEEE Transactions on System, Man and Cybernetics*, PART B: CYBERNETICS, VOL. 37, NO. 3, pp. 576-591, JUNE 2007

[14] Talbi. H, Draa. A, and Batouche. M, “A new quantum-inspired genetic algorithm for solving the travelling salesman problem,” in *2004 IEEE International Conference on Industrial Technology*, Volume 3, pp:1192 – 1197, Dec. 2004.

[15] Benatchba Karima, Koudil Mouloud, Boukir Yacine, Benkhelat,Nadjib, “Image segmentation using quantum genetic algorithms,” in *IEEE 2006 - 32nd Annual Conference on Industrial Electronics*, pp:3556 – 3563, Nov. 2006.

[16] Jun-Su Jang, Kuk-Hyun Han, Jong-Hwan Kim, “Face detection using quantum-inspired evolutionary algorithm,” in *2004 IEEE Congress on Evolutionary Computation*, 2004

[17] T.Hey, “Quantum computing : An introduction,” *Computing & Control Engineering Journal*, Piscataway, NJ: IEEE Press, vol. 10, no. 3,pp. 105-112, June 1999

[18] Wei Song, Soon Cheol Park, “Genetic Algorithm-based Text Clustering Technique: Automatic Evolution of Clusters with High Efficiency,” in *Proceedings of the Seventh International Conference on Web-Age Information Management Workshops*, 2006

[19] S.L. Yang, Y.S. Li, X.X. Hu, and PAN Puo-yu, “Optimization study on k value of K-Means algorithm,” *System Engineering-Theory & Practice*, 26(2):97-101, 2006

[20] Dunn JC, “Well separated clusters and optimal fuzzy partitions,” *J Cybern* 4:95-104, 1974

[21] Turi RH, “Clustering-based colour image segmentation,” PhD Thesis, Monash University, Australia, 2001

[22] Halkidi M and Vazirgiannis M, “Clustering validity assessment: finding the optimal partitioning of a data set,” in *Proceedings of ICDM conference*, CA, USA, 2001

[23] Halkidi M and Vazirgiannis M, “Clustering validity assessment using multi representative,” in *Proceedings of SETN conference,* Thessaloniki, Greece, 2002

[24] Zhang Dingxue, Liu Xinzhi, and Guan Zhihong, “A Dynamic Clustering Algorithm Based on PSO and Its Application in Fuzzy Identification,” in *Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'06)*, 2006

[25] A. Asuncion,D.J. Newman. UCI Machine Learning Repository [Online].Available:http://www.ics.uci.edu/~mlearn/MLRepository.html