

Analysis of Gene Expression Data: Application of Quantum-Inspired Evolutionary Algorithm to Minimum Sum-of-Squares Clustering

Wengang Zhou, Chunguang Zhou, Yanxin Huang, and Yan Wang

College of Computer Science and Technology, Jilin University,
Changchun 130012, P.R. China
wgzhou@email.jlu.edu.cn

Abstract. Microarray experiments have produced a huge amount of gene expression data. So it becomes necessary to develop effective clustering techniques to extract the fundamental patterns inherent in the data. In this paper, we propose a novel evolutionary algorithm so called quantum-inspired evolutionary algorithm (QEA) for minimum sum-of-squares clustering. We use a new representation form and add an additional mutation operation in QEA. Experiment results show that the proposed algorithm has better global search ability and is superior to some conventional clustering algorithms such as k-means and self-organizing maps.

1 Introduction

In the field of bioinformatics, clustering algorithms have received renewed attention due to the breakthrough of microarrays data. Microarrays experiments allow for the simultaneous monitoring of the expression patterns of thousands of genes. Since a huge amount of data is produced during microarray experiments, clustering methods are essential in the analysis of gene expression data. The goal is to extract the fundamental patterns inherent in the data and to partition the elements into subsets referred to as clusters. Two criteria should be satisfied: homogeneity - elements in the same cluster are highly similar to each other; and separation - elements from different clusters have low similarity to each other. In gene expression, elements are usually genes. The vector of each gene contains its expression levels under each of the monitored conditions.

Several clustering algorithms have been proposed for gene expression data analysis, such as hierarchical clustering [1], self-organizing maps [2], k-means [3], and some graph theoretic approaches: HCS, CAST, CLICK [4], MST [5], and many others. In general, these approaches can not be compared directly, since there is no unique measure quality.

In this paper, we propose a novel evolutionary algorithm so called quantum-inspired evolutionary algorithm (QEA) for minimum sum-of-squares clustering. Experiment results show that the proposed algorithm is superior to conventional clustering algorithms such as k-means and self-organizing maps.

2 Minimum Sum-of-Squares Clustering

Clustering can be considered as a combinatorial optimization problem [6], in which an assignment of data vectors to clusters is desired, such that the sum of distance square of the vectors to their cluster mean (centroid) is minimal. Let P_k denote the set of all partitions of X with $X = \{x_1, \dots, x_n\}$ denoting the data set of vectors with $x_i \in R^m$ and C_i denoting the i_{th} cluster with mean \hat{x}_i . Thus the objective function becomes:

$$\min_{p \in P_k} \sum_{i=1}^K \sum_{x_j \in C_i} d^2(x_j, \hat{x}_i) \quad (1)$$

$$\text{with } \hat{x}_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j \quad (2)$$

where $d(\cdot, \cdot)$ is the Euclidean distance in R^m . Alternatively we can formulate the problem as the search for an assignment p of the vectors to the clusters with $C_i = j \in \{1, \dots, n\} | p[j] = i$. Thus the objective function becomes:

$$\min_p \sum_{i=1}^n d^2(x_i, \hat{x}_{p[i]}) \quad (3)$$

This combinatorial optimization problem is called the minimum sum-of-squares clustering (MSSC) problem and has been proven to be NP-hard [7]. The k-means algorithm is a heuristic approach which minimizes the sum-of-squares criterion provided an initial assignment of centroids. It can in fact be regarded as a local search algorithm for this hard combinatorial optimization problem. This fact demonstrates the importance of effective global optimization methods from the field of evolutionary computation.

Since the novel quantum-inspired evolutionary algorithm has better global search ability and has been shown to be effective as compared to the conventional genetic algorithm [8], the application of QEA to the MSSC appears to be promising.

3 Quantum-Inspired Evolutionary Algorithm

Quantum-inspired evolutionary algorithm (QEA) is based on the concept and principles of quantum computing such as a quantum bit and superposition of states [9]. Like other evolutionary algorithms, QEA is also characterized by the representation of the individual, the evaluation function and the population dynamics. The smallest unit of information stored in a two-state quantum computer is called a quantum bit or qubit [10]. A qubit may be in the '1' state, in the '0' state, or in any superposition of the two. The state of a qubit can be represented as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (4)$$

where α and β are complex numbers that specify the probability amplitudes of the corresponding states. $|\alpha|^2$ gives the probability that the qubit will be found in the ‘0’ state and $|\beta|^2$ gives the probability that the qubit will be found in the ‘1’ state. Normalization of the states to unity guarantees the following equation:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{5}$$

A number of different representations can be used to encode the solutions onto individuals in evolutionary computation. Inspired by the concept of quantum computing, QEA uses a new representation, called a Q-bit, for the probabilistic representation that is based on the concept of qubits, and a Q-bit individual as a string of Q-bits. A Q-bit is defined as the smallest unit of information in QEA, which is defined with a pair of numbers (α, β) as $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$, where α and β satisfy the equation (2). A Q-bit individual is defined as:

$$\begin{bmatrix} \alpha_1|\alpha_2| \dots |\alpha_m \\ \beta_1|\beta_2| \dots |\beta_m \end{bmatrix} \tag{6}$$

The state of a Q-bit can be changed by the operation with a quantum gate, such as NOT gate, Rotation gate, and Hadamard gate, etc. Rotation gate is often used to update the Q-bit as follows:

$$\begin{bmatrix} \alpha'_i \\ \beta'_i \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \tag{7}$$

where $|\alpha_i|^2 + |\beta_i|^2 = 1, i = 1, 2, \dots, m$.

The angle parameters used for rotation gate are shown in Table 1. Where x_i and b_i are the i_{th} bit of the best solution b and the binary solution x respectively; $f(\cdot)$ is the objective function defined in Eq. (3).

Table 1. The angle parameters used for rotation gate

x_i	b_i	$f(x) \geq f(b)$	$\Delta\theta_i$	x_i	b_i	$f(x) \geq f(b)$	$\Delta\theta_i$
0	0	false	θ_1	1	0	false	θ_5
0	0	true	θ_2	1	0	true	θ_6
0	1	false	θ_3	1	1	false	θ_7
0	1	true	θ_4	1	1	true	θ_8

The structure of QEA is described in the following:

Procedure QEA

```

begin
 $t \leftarrow 0$ 
initialize  $Q(t)$ 
make  $P(t)$  by observing the states of  $Q(t)$ 
evaluate  $P(t)$ 
store the best solutions among  $P(t)$  into  $B(t)$ 
    
```

```

while (not termination-condition) do
begin
 $t \leftarrow t + 1$ 
make  $P(t)$  by observing the states of  $Q(t - 1)$ 
evaluate  $P(t)$ 
update  $Q(t)$  using Q-gates
store the best solutions among  $B(t - 1)$  and  $P(t)$  into  $B(t)$ 
store the best solution  $b$  among  $B(t)$ 
if (global migration condition)
then migrate  $b$  to  $B(t)$  globally
else if (local migration condition)
then migrate  $b_j^t$  in  $B(t)$  to  $B(t)$  locally
end
end

```

A migration in QEA is defined as the copying of b_j^t in $B(t)$ or b to $B(t)$. A global migration is implemented by replacing all the solutions in $B(t)$ by b and a local migration is implemented by replacing some of the solutions (local group) in $B(t)$ by the best one of them. The local group size is often set as $\max(\text{round}(n/5), 1)$, where n is the population size.

4 The Gene Expression Data Sets

The first data set denoted as HL-60 is described in the work by Tomayo [11] and it contains data from macrophage differentiation experiments. The data set consists of 7229 genes and expression levels at 4 time points including 0, 0.5, 4 and 24 hours. We apply a variation filter which discards all genes with an absolute change in maximum and minimum expression level less than 30. The number of genes which pass the filter is 2792. The vectors are then normalized to have mean 0 and variance 1.

The second data set denoted as Yeast is described in the work by Cho [12]. It consists of 6602 yeast genes including some reduplicate ones measured at 17 time points over two cell cycles. The 90-minute and 100-minute time points are excluded because of difficulties with scaling. Afterwards, we use a variation filter to discard all genes with an absolute expression level change less than or equal to 100, and an expression level of $\max/\min < 2.0$ or $\min = 0$. The number of genes that pass the filter is 2947. Again, the vectors are normalized to have mean 0 and variance 1.

The number of clusters for the clustering is taken from Ref. [11] for the HL-60 data set and from Ref. [12] for the Yeast data set, the cluster number is 12 and 30 respectively.

5 QEA for Clustering Gene Expression Data

In our experiments, we compare quantum-inspired evolutionary algorithm (QEA) with the multi-start k-means (MS+k-means) and self-organizing maps

(SOM) algorithms. Multi-start k-means produces the best solution by running k-means algorithm many times with randomly generating initial centroid. QEA and multi-start k-means algorithms are implemented in Matlab 6.5. The self-organizing maps algorithm is available in the software package Gene Cluster 2.0. In the experiments, the default values of Gene Cluster are used.

5.1 Initialization

The k-means algorithm is first run ten times and so ten initial solutions are produced for each data set. Each solution is composed of n mean vectors and n is the number of clusters for each data set. Given the mean vectors, the cluster memberships can be calculated by calculating the nearest distance of each gene vector with all mean vectors. Therefore it is not necessary to store the cluster memberships in the Q-bit individuals.

5.2 Representation and Fitness Function

The representation used in QEA is straightforward. There are $10 \times n$ mean vectors in all for the ten initial solutions. Then we number each mean vector as $1, 2, 3, \dots, 10 \times n$ and put them into the set V where $V = (v_1, v_2, \dots, v_{10 \times n})$. So $10 \times n$ Q-bits are used in each Q-bit individual. A Q-bit individual has the following form in the t_{th} generation:

$$\left[\begin{array}{cccccccccccc} \alpha_1^t & |\alpha_2^t| & \dots & |\alpha_n^t| & |\alpha_{n+1}^t| & \dots & |\alpha_{2 \times n}^t| & \dots & |\alpha_{9n+1}^t| & \dots & |\alpha_{10 \times n}^t| \\ \beta_1^t & |\beta_2^t| & \dots & |\beta_n^t| & |\beta_{n+1}^t| & \dots & |\beta_{2 \times n}^t| & \dots & |\beta_{9n+1}^t| & \dots & |\beta_{10 \times n}^t| \end{array} \right] \quad (8)$$

where α is given as a random number between 0 and 1 initially and then β could be computed according to Eq. (5). The fitness function used in QEA is the MSSC error provided in Eq. (3).

5.3 Make and Repair Operation

To obtain the binary string, the step of “Make(x)” by observing the states of Q-bit can be implemented for each Q-bit individual as follows:

Procedure Make (x)

begin

$i \leftarrow 0$; $k \leftarrow 0$;

while $i < n \times 10$ do

begin

if $random(0, 1) < |\beta_i^2|$ and $k < n$

then $x(i) \leftarrow 1$; $k \leftarrow k + 1$;

else $x(i) \leftarrow 0$;

$i \leftarrow i + 1$;

end

end

where the variant k is used to guarantee that the number of ‘1’ in each binary string must be less than or equal to n . Afterwards, an additional operation “Repair (x)” should be done to be sure there should be and only be n ‘1’ in each binary string. That is if the number of ‘1’ is less than n in some strings, there must be stochastic search operation to increase the number of ‘1’ for each Q-bit individual as follows:

Procedure Repair(x)

```

begin
calculate the number of ‘1’ as  $k$ ;
while  $k < n$  do
begin
 $randpos \leftarrow random(1, n \times 10)$ ;
if  $x(randpos) = 0$ 
then  $x(randpos) \leftarrow 1$ ;
 $k \leftarrow k + 1$ ;
end
end

```

It can be seen that the repair (mutation) operation can intensify the local search ability of the algorithm and it has similar function of the mutation operation in GA. After obtaining the repaired binary string, we can choose n mean vectors from the set V by observing the binary string for each individual. If the i_{th} bit of the string is equal to ‘1’, then the i_{th} mean vector is chosen. So each Q-bit individual corresponds to different n mean vectors. It is represented as follows:

$$Individual_i \Rightarrow (v_{i1}, v_{i2}, \dots, v_{in}).v_{ik} \in V, k = 1, \dots, n. \quad (9)$$

5.4 Evaluate and Update Operation

Then the evaluate operation is executed to calculate the fitness value according to Eq. (3) for each Q-bit individual represented by Eq. (9) and so the best individual can be selected. The update operation is used to update Q-bit states of each individual by rotation gate in Eq. (7). It is like to the crossover operation in GA for the global search. In this experiment, the global migration period in generation is 10, the local migration period is 1, and the local group size is set to 5 according to others experience [13]. The angle parameter in Table 1 is set as follows according to our experiment observation:

If $\alpha \times \beta > 0$ then $\theta_3 = 0.01\pi$; $\theta_5 = -0.01\pi$;
 If $\alpha \times \beta < 0$ then $\theta_3 = -0.01\pi$; $\theta_5 = 0.01\pi$;
 If $\alpha \times \beta = 0$ then $\theta_3 = \pm 0.01\pi$; $\theta_5 = \pm 0.01\pi$.

6 Experiment Results

In all the experiments, QEA is run with a population size of 20. The k-means and QEA algorithms are all terminated when the maximum generation of 50

Table 2. Experiment results for the two data sets

Data set	Algorithm	Best obj	Avg obj	Excess
HL-60	QEA	1514.3	1598.5	5.56%
	MS+k-means	1514.6	1604.3	5.92%
	3×4-SOM	1523.2	1624.0	6.62%
Yeast	QEA	15741.0	15860.0	0.76%
	MS+k-means	15752.0	15905.0	0.97%
	3×4-SOM	15801.0	16002.0	1.27%

is arrived. The three algorithms are all run ten times, and the best and the average objective value are produced. In Table 2, the results are displayed for the described two gene expression data sets. The objective value is the minimum sum-of-squares referred in Eq. (3). Excess value is the percentage of the average objective value above the best objective value. The smaller excess value means better performance of the algorithm.

The proposed QEA algorithm performs better than other two algorithms even with a small population. The QEA can find best solutions in each data set in spite of its slowly convergence speed sometimes. For the HL-60 data set, the best solutions found by all algorithms have similar fitness, so we believe the best solutions found by the algorithms are optimal. But the excess value of QEA is smaller. For the Yeast data set, the k-means solution is close to the best solution found by QEA, and it seems that the data set is fit for k-means search.

In order to observe the convergence rate, the curve of fitness change is given for the HL-60 and Yeast data sets in Figure 1. The results show the superiority of QEA compared to multi-start k-means and self-organizing maps algorithms. It also shows that QEA can always adjust all individuals to the direction of better solution and find the (near) optimum or best-known solutions to the NP-hard clustering problem. In addition, the k-means algorithm performs better than self-organizing maps for all the two data sets.

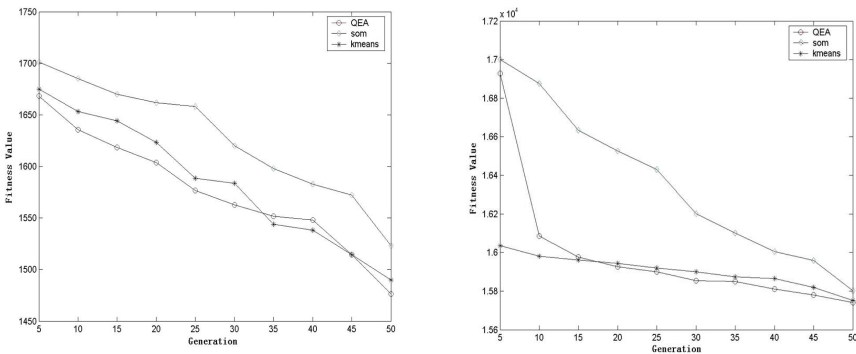


Fig. 1. Fitness changes for the HL-60 (left) and Yeast (right) data sets

Table 3. The distances to the best-known solutions found by QEA

Data set	Algorithm	Distance value
HL-60	MS+k-means	41.97
	3×4-SOM	56.08
Yeast	MS+k-means	487.31
	3×4-SOM	677.70

From the biologists' point of view, it is important how the cluster memberships of the genes change if QEA is used instead of other clustering algorithms. But it is often the case that although the solutions have similar fitness values, the clusters produced by different algorithms may differ drastically. Since the cluster memberships are more important, the distance to the best-known solutions found by QEA for the k-means and SOM algorithms are provided in Table 3. The distance of two solutions is defined as follows:

$$D(p, q) = \sum_{i=1}^n d^2(\hat{x}_{p[i]}, \hat{x}_{q[i]}). \quad (10)$$

For the Yeast and HL-60 data set, however, solutions obtained using the two algorithms appear to be far away from the best-known solutions in terms of cluster memberships. Hence, the previously proposed heuristic method is not sufficient to arrive at optimum clustering solution. The QEA algorithm has been shown to perform well and be more robust in finding better solutions.

7 Conclusions

In this paper, we first introduce a novel evolutionary algorithm so called quantum-inspired evolutionary algorithm, and then quantum-inspired evolutionary algorithm (QEA) with a new representation form and an additional mutation operation is proposed for clustering gene expression data from two data sets. We present experimental evidence that our method is high effective and produces better solutions than the conventional k-means and self-organizing maps clustering algorithms even with a small population. The results also show that the QEA can converge to the optimum solution and demonstrate clearly the effectiveness and feasibility of QEA for minimum sum-of-squares clustering problem. Although the solutions have similar fitness values, the clusters produced by different algorithms may differ drastically. So the distance between the best solutions found by two conventional algorithms and the QEA algorithm is given. It indicates that the proposed algorithm have the ability of finding the global optimum. More gene expression data sets will be tested for QEA and we will compare it with some other clustering techniques in future work.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant No. 60433020 and the Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education.

References

1. Eisen, M., Spellman, P., Botstein, D.: Cluster Analysis and Display of Genome-wide Expression Patterns. *Proceedings of the National Academy of Sciences* (1998) 14863-14867
2. Kohonen, T.: *Self-Organizing Maps*. Berlin Springer (1997)
3. Tavazoie S., et al.: Systematic Determination of Genetic Network Architecture. *Nat. Genet.*, 22 (1999) 281-285
4. Shamir, R., Sharan, R.: Algorithmic Approaches to Clustering Gene Expression Data. In: T. Jiang et al. (eds): *Current Topics in Computational Molecular Biology*. MIT Press (2002) 269-299
5. Xu, Y., Olman, V., Xu, D.: Clustering Gene Expression Data Using a Graph-theoretic Approach: An Application of Minimum Spanning Trees. *Bioinformatics*, 18 (2002) 536-545
6. Merz, P.: Analysis of Gene Expression Profiles: An Application of Memetic Algorithms to the Minimum Sum-of-squares Clustering Problem. *Biosystem*, 72 (2003) 99-109
7. Brucker, P.: On the Complexity of Clustering Problems. *Lecture Notes in Economics and Mathematical Systems*, 157 (1978) 45-54
8. Han, K., Kim, J.: Quantum-inspired Evolutionary Algorithm for a Class of Combinatorial Problem. *IEEE transactions on evolutionary computation*, 6 (2002) 580-593
9. Han, K., Kim, J.: Genetic Quantum Algorithm and Its Application to Combinatorial Optimization Problem. *Proceedings of the Congress on Evolutionary Computation* (2000) 1354-1360
10. Hey, T.: Quantum Computing: An Introduction. *Computing & Control Engineering Journal*, 10 (1999) 105-112
11. Tamayo, P., Slonim, D., et al.: Interpreting Patterns of Gene Expression with Self-organizing Maps: Methods and Application to Hematopoietic Differentiation. *Proceedings of the National Academy of Sciences* (1999) 2907-2912
12. Cho, R.J., Winzeler, E.A., Davis, R.W.: A Genome-wide Transcriptional Analysis of the Mitotic Cell Cycle. *Mol. Cell*, 2 (1998) 65-73
13. Han, K., Kim, J.: On Setting the Parameters of Quantum-inspired Evolutionary Algorithm for Practical Applications. *Proceedings of the Congress on Evolutionary Computation* (2003) 178-184