Journal of

Software Engineering

ISSN 1819-4311

Academic
Journals Inc.

www.academicjournals.com

OPEN ACCESS Journal of Software Engineering

ISSN 1819-4311
DOI: 10.3923/jse.2017.183.193

Research Article
Hybrid Algorithm of Gene Clustering Based on GPU

Zhangrong Qin, Zhang Li, Xuan Zhou, Binghai Wen and Chaoying Zhang

Guangxi Key Lab of Multi-source Information Mining and Security, Guangxi Normal University, 541004 Guilin, China

Abstract

Background: In order to improve the defects of falling into local optimum prematurely and the low global search capability in K-means algorithm for gene clustering analysis, an efficient hybrid algorithm by combining PK-means, Cellular Automata (CA) and Firefly Algorithm (FA), called PK-CA-FA is presented. **Materials and Methods:** In the algorithm, CA is introduced for relieving the problem of easy to fall into a local optimum at the first iterative stage of the PK-means and then FA is introduced to enhance the global search ability at the second iterative stage. Furthermore, in order to improve the computational efficiency, this algorithm is implemented on Graphics Processing Unit (GPU) with a Compute Unified Device Architecture (CUDA) parallelly. **Results:** For verifying its performance, the algorithm and its parallel version are utilized to cluster gene expression data on several benchmark datasets. The experimental results show that the proposed algorithm can effectively avoid being trapped in a bad local optimum and is generally more accurate and stable than PK-means algorithm. At the same time, the parallel implementation of the algorithm on GPU is significant, by which a considerable acceleration ratio with respect to CPU is obtained. **Conclusion:** It is concluded that the PK-CA-FA is an efficient algorithm for gene clustering with strong accuracy, stability and high speedup and the algorithm can be expected to find its further applications for practical gene clustering analysis.

Key words: Gene clustering, PK-means, CA, FA, GPU

Received: July 12, 2016 Accepted: August 29, 2016 Published: March 15, 2017

Citation: Zhangrong Qin, Zhang Li, Xuan Zhou, Binghai Wen and Chaoying Zhang, 2017. Hybrid algorithm of gene clustering based on GPU. J. Software Eng., 11: 183-193.

Corresponding Authors: Binghai Wen and Chaoying Zhang, Guangxi Normal University, No. 15, Yucai Road, Guilin, Postal Code 541004 Guangxi, China Tel:+86-0773-5811693

Competing Interest: The authors have declared that no competing interest exists.

Data Availability: All relevant data are within the paper and its supporting information files.

INTRODUCTION

With the development of life science study, the study category of bioinformatics is increasingly expanded. As one of the most promising technologies in bioinformatics, by which expression levels of thousands of genes can be simultaneously detected and a large number of gene expression data contained the genetic information can be generated, microarray technology has received a lot of attention and been popular from scholars. How to extract the meaningful information for human beings from analyzing efficiently the vast amounts of gene expression data is a hot issue to be solved urgently. Clustering analysis for gene expression data is one of the effective methods. It can group the genes containing similar expression levels into a co-expression category, which would be helpful to the comprehensive studies of gene function, gene regulation and cell subtype etc. The widely used traditional algorithms for gene clustering mainly include K-means[1], hierarchical clustering[2], self-organizing maps (SOMs)[3] and Fuzzy K-means (FKM)[4] etc. The K-means is one of the well-known popular clustering algorithms because of its simplicity and fast convergence. However, it is sensitive to the selection of an initial clustering and easily becoming trapped in a local minimum untimely. Many efforts have been contributed to overcome the problems. In recent years, an integration of intelligent bionic algorithm based on natural biological characteristic with K-means is considered as an alternate solution. Rahman and Islam[5] proposed a hybrid clustering algorithm combining a novel genetic algorithm with K-means in which it is capable of automatically finding the right number of clusters and identifying the right genes through a novel initial population selection approach. The numerical results showed that a statistically significant superiority of their algorithm over five recent algorithms on 20 natural datasets. Prabha and Visalakshi[6] introduced an improved gene clustering algorithm based on Particle Swarm Optimization (PSO) and K-means. The results reflected that the improved algorithm is more efficient than K-means algorithm. Du *et al.*[7] presented a hybrid gene clustering algorithm which combines particle-pair optimizer (PPO)[8] and K-means algorithm. It applied PPO for gene clustering instead of PSO, in which two particles work cooperatively and formed a particle-pair as a swarm with small population size. Because population size was relatively small, the position between particles was easy to coordinate and the particles could move towards the optimal solutions. Their results indicated that PK-means had a higher accuracy and robustness than K-means. However, the algorithm would result in the lack of adequate information exchange and sharing between particles because of its smaller population size in PK-means. The evolutionary formula of standard PSO algorithm[9] is still adopted, thus, it is easy to fall into local optimum and degrade its global search ability.

Cellular Automaton (CA)[10] based on evolutionary dynamics and Firefly Algorithm (FA)[11] based on bionic principle have been gradually applied in the field of gene clustering in the last few years. Shuai *et al.*[12] proposed a generalized cellular automata algorithm for clustering which showed a good clustering effect. Shi *et al.*[13] presented a cellular particle swarm optimization algorithm and claimed that it was better than other variant PSO algorithms. Senthilnath *et al.*[14] applied FA to the clustering analysis of the datasets according to the attribute values of data objects and obtained better clustering results than those from PSO. Generally, CA has an excellent local search and information exchange ability, meanwhile FA can efficiently search solution space and obtain the local and global optimal solutions at the same time. It could be an effective way to develop an accurate and efficient gene clustering algorithm by combining CA with FA to create a hybrid algorithm.

With the rapid development of microarray technology, the amounts of gene expression data become more and more large. The computational intensity and complexity of gene clustering analysis have been far beyond the scope of a personal computer. In recent years, Graphics Processing Unit (GPU) has been treated as a parallel computing device on which a large number of threads can run simultaneously. In 2007, Compute Unified Device Architecture (CUDA) suitable for GPU general computing was launched by NVIDIA. This technology decreases the complexity of GPU development and promotes its applications in scientific computations. Many heavy computing tasks which were only achieved on a large computer previously and can be easily accomplished by parallel computing on a station with a few GPU cards. Nowadays, CUDA technology has been gradually applied in clustering analysis. Yan *et al.*[15] and Hooda and Nandal[16] used GPU to speed up the K-means algorithm. Serapiao *et al.*[17] accelerated the hybrid algorithm combining K-means and K-harmonic with fish school search algorithm by GPU. The accelerations are obvious in these applications and GPU computation is a promising technology to analyze gene expression data.

In this study, a new hybrid algorithm for gene clustering is proposed by combining PK-means, CA and FA on the basis of PK-means algorithm called PK-CA-FA. In this way, the advantages of FA and CA mentioned above are introduced into the PK-means algorithm and the final hybrid algorithm may effectively avoid falling into local optimum and enhance the global search capability. The algorithm is designed and

implemented on GPU by using CUDA technology and the numerical results show that the computational efficiency is improved notably.

MATERIALS AND METHODS

PK-means algorithm: The PK-means[7] is a hybrid algorithm for gene clustering which is the integration of PPO and K-means algorithm. The PPO inherits the basic features of PSO, two particles work cooperatively and form a particle-pair with small population size to replace the traditional particle swarm. In PK-means, the position of each particle is composed of K clustering centroids, each clustering centroid is an L-dimensional vector and the dimension of a particle N is K×L. The structure of a particle is shown in Eq. 1 as:

$$z_i = (y_{i1}, y_{i2},\ldots, y_{ij},\ldots, y_{iK}) \quad (1)$$

where, y_{ij} represents the jth clustering centroid vector of the ith particle.

In the process of iteration, the velocity and position of a particle are updated according to the standard PSO evolution in Eq. 2 and 3 as:

$$v_{id}(t+1) = w\times v_{id}(t)+c_1\times r_1\times(p_{id}(t)-x_{id}(t))+c_2\times r_2\times(p_{gd}(t)-x_{id}(t)) \quad (2)$$

$$x_{id}(t+1) = x_{id}(t)+v_{id}(t+1) \quad (3)$$

where, v and x represent the velocity and position of a particle, respectively, i is subscript of the ith particle, i = 1, 2, d is the dth dimension of the particle, d = 1, 2,..., N, t is times of iterations, w is the inertia weight, it is used to control the change of a particle flight speed, c_1 and c_2 are called acceleration factors, r_1 and r_2 are two random numbers within (0, 1), p_i represents a previous optimal position of the ith particle and p_g represents a global optimal position of the whole particle swarm.

A description of PK-means algorithm in detail is given in references[7].

Firefly algorithm: Firefly algorithm proposed by Yang[10] is a stochastic optimization algorithm based on swarm intelligence. It is widely used to solve the optimization problems and finds the best optimal solutions through imitating the fireflies cooperative behaviors in which the fireflies are attracted to each other by the lights generated from their life habits such as foraging, mate choice and so on. The algorithm description is as follows.

Assuming that the absolute brightness of the firefly i is bigger than that of the firefly j, the firefly j is usually attracted by the firefly i and moves to the ith one. The definition of the relative brightness of the firefly i-j is in Eq. 4 as:

$$I_{ij}\left(r_{ij}\right) = I_i \times \exp\left(-\gamma r_{ij}^2\right) \quad (4)$$

where, I_i is the absolute brightness of the firefly i, its value is equal to the objective function value located at the firefly i. γ is the light absorption coefficient, r_{ij} is the cartesian distance from the firefly i-j.

The attraction of the firefly i relative to j is defined in Eq. 5 as:

$$\beta_{ij}(r_{ij}) = \beta_0\times\exp(-\gamma r_{ij}^2) \quad (5)$$

where, β_0 is the maximum attractiveness that is the attraction from the fireflies in the light sources.

Attracted by a firefly i, firefly j moves toward to i and its position $\vec{x}_j$ will be updated according to Eq. 6:

$$\vec{x}_j(t+1) = \vec{x}_j(t)+\beta_{ij}(r_{ij})(\vec{x}_i(t)-\vec{x}_j(t))+\alpha\vec{\varepsilon}_j \quad (6)$$

In Eq. 6, $\vec{x}_I$ and $\vec{x}_j$ are the position vector of the firefly i and j, respectively, α is a perturbation factor within (0,1) and $\vec{\varepsilon}_j$ is a random number vector obtained by the Gauss distribution or other distributions. On the right of the Eq. 6, the first term represents the current position of the firefly j, the second term indicates the position variation due to being attracted by other fireflies in the population and reflects the global optimization ability of FA, the third term indicates the movement process of the local search and reflects the local optimization ability of FA.

A detailed description of the FA is given in references[10].

Cellular automaton: Cellular Automaton (CA) is first proposed by Von Neumann, it is a dynamic model capable of simulating concurrently a complex structure and process by using a large number of cells[11]. The CA model is mainly composed of cells, space, neighbors and rules. Figure 1 is a commonly used Moore neighborhood model of a two-dimensional cellular automaton. In this model, each grid represents a cell, all of the cells constitute cellular space, the eight cells around a cell are called neighbors of this cell. The states of a cell are defined as anyone of the solutions of the problem to be optimized as well as the corresponding objective function values. The state of a

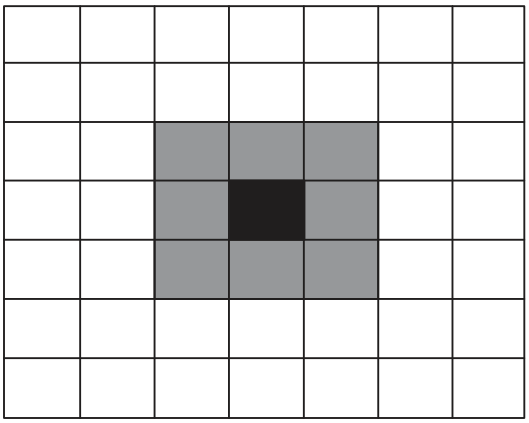

Fig. 1: Moore neighborhood model

cell at the next moment is determined by both of the cell and its neighbors. The evolution rules are the dynamic functions with which the state of a cell at the next time is fully determined by the current states of this cell and its neighbors. Each cell is updated concurrently according to the given local evolution rules and a large number of cells create the dynamic system evolution through simple interactions with each other.

New algorithm PK-CA-FA: A new hybrid algorithm is proposed by combining PK-means, CA and FA called PK-CA-FA in order to overcome the defects in PK-means. The CA is introduced in the first iterative stage of the PK-means. The defect of the PK-means easy to fall into local optimum can be improved by using the advantages of the strong message passing abilities between cells in CA. Each particle is considered as a cell and then put into the CA for evolving. The cellular evolution rules to control the evolution of cellular state are as follows: for each cell, comparing its previous optimal the Mean Square Error (MSE) with the one of its neighbors, if the MSE of the neighbors is better, its previous optimal position and MSE are updated by those of the neighbors, respectively. When introducing the CA, there needs to increase appropriately the number of the particles in population in order to obtain the elite particle-pairs in a wider range. In this way, not only the propagation velocity of the global optimal value of particles can be improved but also a better accuracy of the new hybrid algorithm can be obtained because it can search the neighborhood of a particle more adequately. In the second iterative stage of the PK-means, FA is introduced into PK-means if the MSE fluctuation of a particle is continuously within a specified range for INV times where the value of the parameter INV will be given experimentally. At this time, a particle is seen as a firefly, its fitness value is identified as the brightness of the firefly and its position is determined by Eq. 6. By using the advantages of obtaining the better local and the better global optimal solutions at the same time of FA which can efficiently search solution space, the new algorithm can further avoid falling into local optimal solution prematurely and improve its global search ability.

In this study, the Euclidean distance D (a, b) between the gene a and b is used to determine the clustering partition, the MSE function is used as the criterion function of a clustering objective, the within-class compactness D_1 is used to measure the homogeneity and the between-class separability D_2 is used to indicate the difference in following equations as:

$$D(a,b)=\sqrt{\sum_{j=1}^{n}(x_{aj}-x_{bj})^2} \tag{7}$$

$$MSE=\frac{1}{M}\sum_{i=1}^{m}\left[\min_{y_j\in Y}D\left(x_i,y_j\right)\right]^2 \tag{8}$$

$$D_1=\frac{1}{M}\sum_{j=1}^{K}\sum_{i=1}^{K_j}\left[\underset{y_j\in Y}{D}\left(x_i,c_j\right)\right]^2 \tag{9}$$

$$D_2=\frac{1}{\sum_{i\neq j}K_iK_j}\sum_{i\neq j}\left[K_iK_jD\left(c_i,c_j\right)\right] \tag{10}$$

where, x_{ij} represents the expression level of the gene i under the experimental sample condition j, M is the number of genes, K_j is the number of genes in the jth class and c_j represents the jth clustering centroid.

The pseudo code of the PK-CA-FA algorithm shown in algorithm 1.

Algorithm 1: PK-CA-FA algorithm

```
FOR i = 1:2 // Obtain two elite particles (EP) from two particle swarms
  Randomly set the initial positions and velocities of the particles p1(0), p2(0),⋯,
  p(0)popsize
    Calculate the MSEs of the particles p1(0), p2(0),⋯, p(0)popsize
    Select the particle having the best MSE from{p1(0), p2(0),⋯, p(0)popsize} as pg(0)
    FOR t =1:tmaxloop1
      FOR s = 1: popsize
        Update the position and velocity of the particle ps(t) by Eq. 2 and 3
        Perform three iterations of K-means for the particle ps(t) and update
        the cluster centroid
        Calculate the MSE of the particle ps(t),put all of the particles with the
        size of CELLROWS* CELLCOLS into CA
          FOR idx=0: CELLROWS-1
            FOR idy=0: CELLCOLS-1
            Search out the particle having the best (optimal) position from
            the neighborhood of the cell[idx][idy],and then take it as the
            p(t)pbest(idx,idy) of the corresponding cell
            END
          END
```

Algorithm 1: Continue

```
            Save the best position, found so far, of the particle into p(t)pbests
            Take the best position from p(t)pbest1, p(t)pbest2,⋯, p(t)pbests as pg(t)
        END // The end of the s loop
      END //The end of the t loop
        EPi=pg(t)
      END // The end of the i loop
    p1(t)=EP1, p2(t)=EP2, // Obtain an elite particle-pair
    Set randomly the initial velocity of the particle p1(0) and p2(0)
    FOR t =1:tmaxloop2
        FOR s=1: 2
          Update the position and velocity of the particle ps(t) by Eq. 2 and 3
            Perform three iterations of K-means for the particle ps(t) and update the
            cluster centroid
            Calculate the MSE of the particle ps(t),and then take its best position,
            found so far, as the p(t)pbests
            IF the MSE fluctuation of the particle ps(t) is continuously within the
specified range for INV times
                doFA() // Perform firefly algorithm
            END
            Take the best position from the p(t)pbest1, p(t)pbest2,⋯, p(t)pbests as pg(t)
        END
        EP3= pg(t)
      END
// According to the EP3, the final clustering results can be obtained.
```

RESULTS AND DISCUSSION

Experimental datasets and parameters: The experiments were done on Yeast[18], Lym[18] and Yeast_GOE[19] three real datasets. The simulation platform is shown as follows: The operating system is 64-bit Windows 7, CPU is Intel(R) Core(TM) i5-3470 3.20GHz and GPU is NVIDIA GeForce GTX650.

Two populations are used in the experiments, each population consists of 16 particles and the clustering number K is 256. The size of the cellular spaces is 4×4 and the Moore neighborhood model is used in CA. The parameter settings in the previous study[7] are used: The inertia weight w is 0.1, the acceleration constants $c_1$and c_2 are 0.3 and 0.5, respectively the iteration number in the first iterative stage MAXLOOP1 is 19 and the iteration number of in the second iterative stage MAXLOOP2 is 14. For FA, the maximum attractiveness β_0 takes 1.0, the perturbation factor α takes 0.01 and the light absorption coefficient γ takes 0.01. In the second stage of the iterative process, FA is introduced if the fluctuation of a particle's MSE value is continuously less than the threshold ROU for INV times. Experiments show that the efficiency of the algorithm is the highest when ROU takes 0.0001 and INV takes 3.

In order to verify the effect of CA being introduced into PK-means, this study combines PK-means and CA in the first iterative stage of the PK-means and obtains a so called PK-CA algorithm in which the population size is expanded. For comparison, another based PK-means algorithm called PK-noCA is also presented in which the PK-means doesn't combine with CA and the population size is expanded. In order to verify the effect of FA being introduced into PK-CA, PK-CA and FA is combined in the second iterative stage and obtained a hybrid algorithm called PK-CA-FA. The PK-CA-FA, PK-CA, PK-noCA, PK-means and K-means algorithms are run on the three datasets above mentioned, respectively. Each algorithm is executed 10 times in each dataset. The average, minimum and maximum of MSE in 10 times are listed in Table 1.

From Table 1, it can be seen that the clustering results of K-means is the worst in all of the three datasets (The smaller MSE is the better the clustering result is). The PK-means and PK-CA are followed by K-means. The optimal clustering results are those obtained by PK-CA-FA with which the minimum average MSE and the minimum MSE in 10 runs can be achieved on each dataset. These results show that the clustering accuracy is improved to a certain extent by the PK-CA-FA.

From Table 1, it also can be seen that the within-class compactness D_1 of PK-CA-FA is better than four other

Table 1: MSE, D1 and D2 of five algorithms in 10 runs

Dataset	Algorithm	MSE (Average)	MSE (Minimum)	MSE (Maximum)	D1	D2
Yeast	K-means	8525.1	8399.1	8621.2	87.5	561.2
	PK-means	7933.3	7864.0	7996.3	85.8	564.2
	PK-noCA	8040.5	7902.9	8140.7	86.2	556.7
	PK-CA	7850.8	7823.0	7893.4	85.4	561.7
	PK-CA- FA	7850.0	7822.6	7879.7	85.3	584.1
Lym	K-means	292954.0	281182.0	305165.0	440.8	1254.6
	PK-means	239049.0	234671.0	241366.0	426.3	1306.7
	PK-noCA	236857.0	233674.0	240497.0	426.8	1486.4
	PK-CA	232135.0	230325.0	233717.0	425.1	1416.4
	PK-CA-FA	231987.0	229000.0	233954.0	424.2	2185.6
Yeast_GOE	K-means	33.1	32.7	33.3	5.4	11.1
	PK-means	30.5	30.4	30.7	5.4	11.4
	PK-noCA	30.6	30.3	30.9	5.4	11.7
	PK-CA	30.2	30.0	30.2	5.3	12.0
	PK-CA- FA	30.1	29.1	30.3	5.2	16.2

algorithms. The smaller D_1 is the higher similarity of gene expression belonging to the same class is and the gene function may be more similar. On the other hand, the between-class separability D_2 of PK-CA-FA is the biggest in the five algorithms. It means the greater difference of gene expression and the smaller association between different classes. These results further indicate that the proposed algorithm is improved in avoiding being fall into local optimum prematurely and enhancing the global search capability because of the introduction of CA and FA.

In order to analyze the effect of CA being incorporated into PK-means in detail, the researchers compare the MSEs obtained, respectively by PK-means, PK-noCA, PK-CA and PK-CA-FA after 19 iterations in the first iterative stage with those obtained by using K-means as shown in Fig. 2. It can be seen that the MSE values obtained by PK-CA and PK-CA-FA is obviously smaller than those obtained by the algorithms with noCA. Because of CA being introduced, the particle number of the K-means is increased suitably which can improve the propagation speed of the global optimal value of a particle in a population and fully search neighborhood of a particle by using the CA's powerful neighbor communication ability. Overall, the incorporation of CA into PK-means is effective and it can obtain a better results.

Figure 3 is a comparison of the MSEs obtained by the mentioned algorithms by executing 10 runs on the three datasets. It can be seen that the MSE curves of PK-CA and PK-CA-FA are lower than those of the PK-noCA, the PK-means and the K-means, the clustering results of the former is superior to those of the latter. This also shows that with the CA being introduced, the clustering algorithms can avoid being trapped prematurely in local optimum and thus, the results are significantly improved. By comparison of the MSEs of PK-CA-FA and PK-CA, it can be seen that the clustering results of the former are better than those of the latter and this may indicate that the global searching ability of the new algorithm is enhanced and the clustering accuracy is improved by the introduction of the FA being introduced. In addition, the MSE curve of PK-CA-FA has less fluctuation than those of the other algorithms and it also hints that the PK-CA-FA has a better stability.

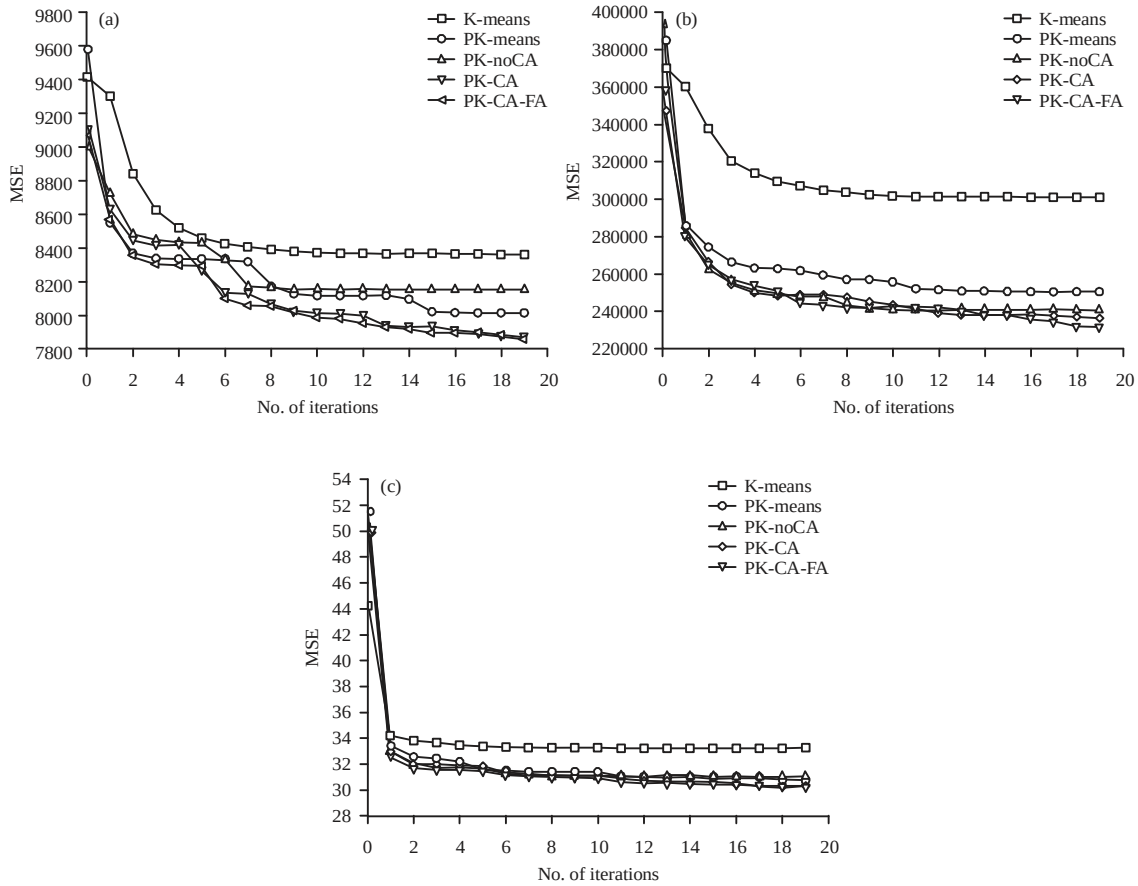

Fig. 2(a-c): MSE plots by 19 iterations on three datasets, (a) Yeast, (b) Lym and (c) Yeast_GOE

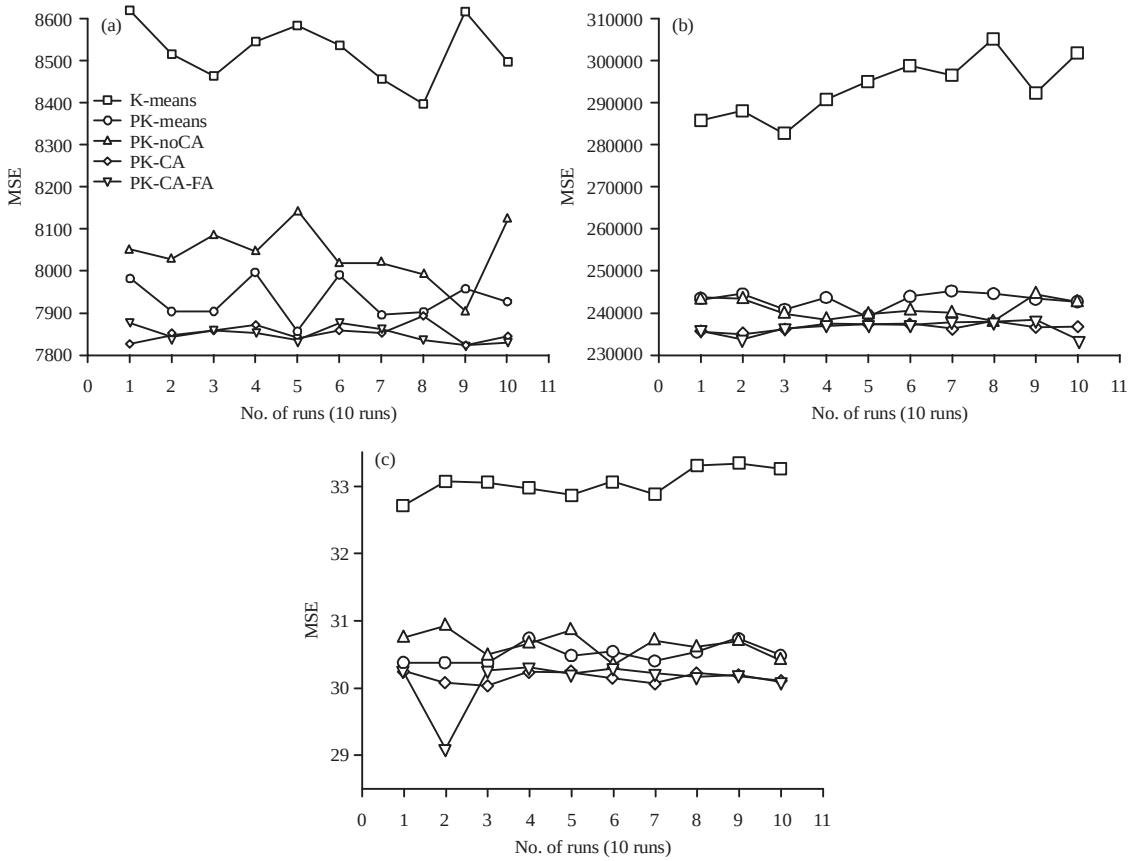

Fig. 3(a-c): MSE plots by 10 runs on three datasets, (a) Yeast, (b) Lym and (c) Yeast_GOE

CUDA programming model: The GPU is a single instruction multiple data (SIMD) multi-core processor in the graphics card of a personal computer. It has dozens or even hundreds of processing cores, its core number is far more than CPU and has a powerful floating point operation capability. The CUDA technology based on general purpose computation on GPU provides a similar C language development environment, designer don't need to know complex graphics API knowledge and can develop a CUDA program by using C language and the CUDA extension library. In the CUDA programming model[20] CPU is regarded as a host for controlling the whole serial logic and the task scheduler of a program while, GPU a coprocessor or device which performs the parallel computing parts of the program. Figure 4 shows that a CUDA program is split into the host code that is serially executed on CPU and the device code that is executed on GPU parallelly. The device code is organized into a Kernel in the CUDA program which is the function executed in concurrent threads on GPU. A thread is the basic unit of the concurrent execution, a certain number of threads are grouped into thread blocks which execute the same instructions on different data, thread blocks are grouped into a grid and a grid is a kernel in a CUDA program. First, a CUDA program starts its execution in host and then the host initializes the device and copies data to the device memory,

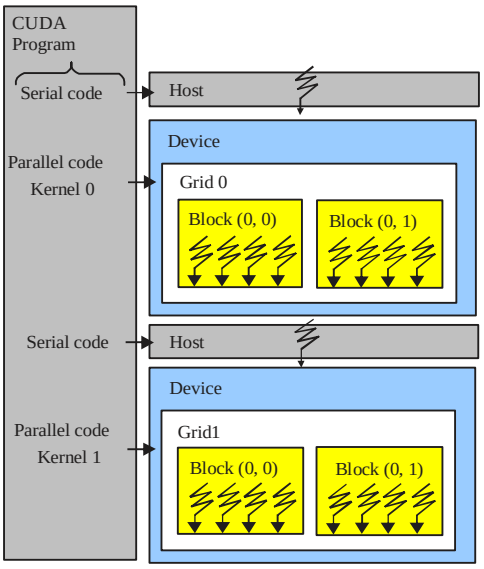

Fig. 4: CUDA programming model

after that the host calls the kernel function and the parallel computing is executed in device finally, the results will be copied back to the host memory from the device.

There provide several different characteristic memories used for programming in GPU. Global memory and shared memory are the major kinds of them. The global memory has large storage capacity and slow access speed, the shared memory has limited storage capacity and but its access speed is fast. The selection of the appropriate GPU memories will help to improve the computational efficiency.

Parallel analysis of the new algorithm: Through analyzing the new algorithm, it's found that there are three cases which can be calculated in parallel; (1) Two particles in a particle-pair are independent of each other, (2) The Euclidean distances between each data object and the clustering centroids and classifying the data objects in the K-means algorithm. Although, the update of the new clustering centroids can be also executed in parallel, it is put on CPU to process according to the recommendations of the previous study[21] and (3) The fitness values calculation of the particles during the two iterative stages. The computational efficiency will be improved significantly if the new algorithm is performed on GPU because the (2) and (3) are the most time-consuming operations in the whole process.

Parallel design and implementation of the new algorithm: The CUDA program consists of both of the CPU codes and the GPU codes in which these two codes work cooperatively, the CPU codes are responsible for serious computing and task scheduling and the parallel computing is executed by the GPU codes. The flow chart of the CUDA program for the first iterative stage is shown in Fig. 5 and that for the second iterative stage is omitted as is the same to the previous one. The program execution is as follows: (1) The clustering centroids are initialized on CPU and then the related data are transmitted to GPU, (2) On GPU, the Euclidean distances between each data object and clustering centroids are calculated then all of the data objects are classified according to the Euclidean distances and finally, the clustering results are returned to CPU, (3) The clustering centroids are recalculated based on the returned clustering results on CPU and then the new clustering centroids are again transferred to GPU, (4) The fitness values of the particles are calculated in parallel on GPU and returned to CPU and (5) The other parts of the algorithm are performed sequentially on CPU.

Clustering centroid data is a K×L, 2-dimensional matrix and gene expression data is a M×L, 2-dimensional matrix. One GPU thread is used to correspond to a row of the gene expression data matrix (i.e., a gene) and thus, the thread number is M as shown in Fig. 6. Taking into account the characteristics of GPU memory, the gene expression data and clustering centroid data are stored in a 1-dimensional array of the GPU global memory, respectively as the number of them is large. Because two particles are processed in parallel, there need to create four 1-dimensional arrays, d_genes1,

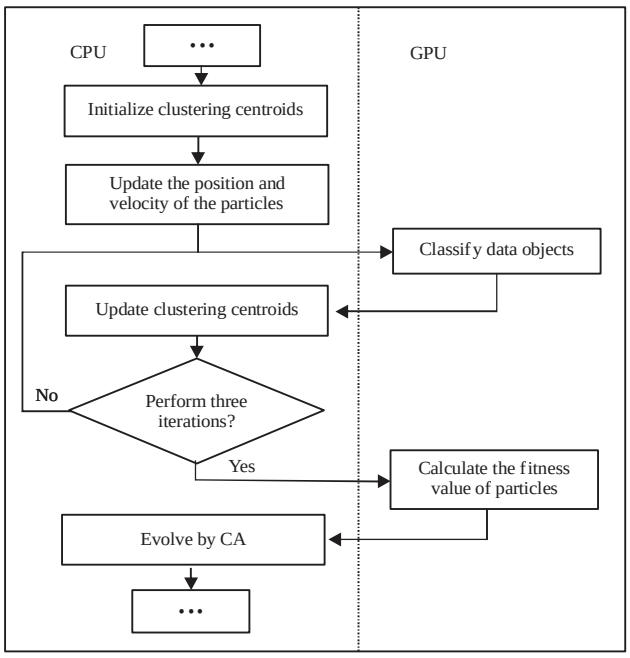

Fig. 5: Flow chart of the CUDA program

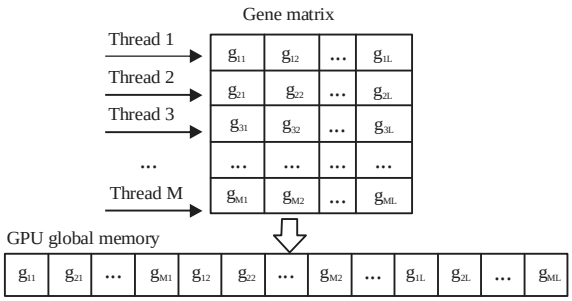

Fig. 6: Data layout on GPU global memory

d_genes2, d_cents1 and d_cents2 for storing the gene expression data and clustering centroid data of the two particles, respectively. It is well known that the speed of the global memory is slow according to the recommendation of the previous study[20], one of the most effective way to improve the access efficiency of the global memory is to make a reasonable arrangement of the storage layout so that the access to the global memory is as much as possible to be the coalescent access. This study layout is shown in Fig. 6, the gene expression data is stored in a 1-dimensional GPU array by column. When the threads within a warp (32 threads) of a thread block access a contiguous segment of 128 bytes (single precision calculation is used and each data occupies 4 bytes) in the global memory then the 32 times accesses to the global memory will be coalesced into one time and this make the access efficiency have a significant improvement. In addition, in order to further improve the computational efficiency and some of the temporary results will be used share memory.

According to the parallel analysis about the new algorithms above mentioned two kernel functions are created and run on GPU. The first one is DataObjectsClassify, its main function is to calculate the Euclidean distances between each data object and the clustering centroids and then classify data objects and its main code is shown in algorithm 2. The second one is CalculateFitnessValues, its main function is to calculate the fitness value of the two particles and its main code is shown in algorithm 3.

Experimental datasets, parameters and environment are the same as those described above mentioned. The thread block size is 8×8 threads, the total number of threads is M and the number of the thread block is (M+63)/64 where, M is also the number of the genes in the datasets. The data sizes (M×L) of the dataset Yeast, Lym and Yeast_GOE are 2884×17, 4026×43 and 2944×173, respectively.

Algorithm 2: DataObjectsClassify kernel

```
__global__ void DataObjectsClassify (float* d_genes1,float* d_genes2, float*
d_cents1,float* d_cents2,float *rs_clu1,float *rs_clu2)
{ int idx=threadIdx.x+blockIdx.x*blockDim.x;
    int idy=threadIdx.y+blockIdx.y*blockDim.y;
    int offset=idx+idy*blockDim.x*gridDim.x;
    float tmp_a,eu_dist1,eu_dist2,min_index1,min_index2;
    float mindist1=3.4028235E+38,mindist2=3.4028235E+38;
    if (offset<M)
    { for (int j=0; j<K; j++)
     { eu_dist1=0;eu_dist2=0;
```

Algorithm 2: Continue

```
    for (int m=0; m<L; m++)
                { tmp_a=d_genes1[offset*L+m]-d_cents1[j*L+m];
                eu_dist1+=tmp_a*tmp_a; //for Particle1
                tmp_a=d_genes2[offset*L+m]-d_cents2[j*L+m];
            eu_dist2+=tmp_a*tmp_a; // for Particle2
} // calculate the Euclidean distance
                if (mindist1>eu_dist1)
                { mindist1=eu_dist1;min_index1=j;}
                if (mindist2>eu_dist2)
                { mindist2=eu_dist2;min_index2=j;}
    }
    if (rs_clu1[offset]!=min_index1) //Particle1 Update clustering results
     {rs_clu2[offset]=min_index1}
    if (rs_clu2[offset]!=min_index2) //Particle2 Update clustering results
     {rs_clu2[offset]=min_index2}
 }
}
```

Algorithm 3: The CalculateFitnessValues kernel

```
__global__void CalculateFitnessValues(float* d_genes1,float* d_genes2, float*
d_cents1,float* d_cents2,float *d_block1,float *d_block2)
{ int idx=threadIdx.x+blockIdx.x*blockDim.x;
    int idy=threadIdx.y+blockIdx.y*blockDim.y;
    int offset=idx+idy*blockDim.x*gridDim.x;
    int ca_index=threadIdx.x+threadIdx.y*BLOCK_SIZE;
    float mindist1=3.4028235E+38,mindist2=3.4028235E+38;
float tmp_a,eu_dist1,eu_dist2;
    __shared__ float cache1[BLOCK_SIZE*BLOCK_SIZE];
__shared__ float cache2[BLOCK_SIZE*BLOCK_SIZE];
    if (offset<M)
 { for (int j=0; j<K; j++)
     { …..// the code of calculating the Euclidean distance between two
particles is the same as DataObjectsClassify kernel
                if (mindist1>eu_dist1){ mindist1=eu_dist1;} //for Particle1
                if (mindist2>eu_dist2){ mindist2=eu_dist2;} //for Particle2
     }
     cache1[ca_index]=mindist1;
     cache2[ca_index]=mindist2;
    }
    __syncthreads();
    // calculate the summation by parallel reduction
    int i=BLOCK_SIZE*BLOCK_SIZE/2;
    while (i!=0)
 { if (ca_index<i)
     { cache1[ca_index]=cache1[ca_index]+cache1[ca_index +i];
     cache2[ca_index]=cache2[ca_index]+cache2[ca_index +i];
     }
     __syncthreads();
     i/=2;
    }
 if (ca_index==0)
    { d_block1[blockIdx.x]=cache1[0]; //for Particle1
     d_block2[blockIdx.x]=cache2[0]; //for Particle2
    }
}
```

Table 2: Average runtimes and the speedup

Dataset	CPU (s)	GPU (s)	Speedup
Yeast	176.5	3.6	49.0
Lym	555.8	10.9	50.9
Yeast_GOE	1371.6	37.4	36.6

The CPU program and the program based on GPU are executed 10 runs, respectively the average runtimes of 10 runs and the speedup are shown in Table 2.

It can be seen from Table 2 that on the one hand, in general, the parallel algorithm can obtained a considerable speedup on all of the three datasets because the number of threads is large and the efficiencies of these threads executed in parallel on GPU are much higher than those of the threads executed serially on CPU, on the other hand, the acceleration effects are different in three datasets. For the Lym dataset, the parallel algorithm has achieved 50.9 times maximum speedup as the gene number M of this dataset is maximum, the number of threads can be used for parallel computing is the largest and another reason may be that its sample number L is relatively small among the three datasets (The bigger L is the more data needed to be processed serially in one GPU thread are). Although, the gene number M of the dataset Yeast and Yeast_GOE is similar, the sample number L of the dataset Yeast_GOE is maximum and the data needed to be processed serially is also maximum so, the speedup obtained on the Yeast_ GOE is slower than that obtained on the Yeast. The experimental results show that it is feasible for the new algorithm being carried out in parallel on GPU and it can achieve a good acceleration performance.

CONCLUSION

This study presents a new hybrid algorithm PK-CA-FA which combines CA and FA basing on the PK-means algorithm and the related experiments are carried out on three real datasets. The results show that the proposed algorithm can obtain better accuracy and stability than PK-means. At the same time, in order to improve the computational efficiency of the algorithm, the parallel algorithm is designed and implemented on GPU by the CUDA technology. The experiment results indicate that it is very feasible for the new algorithm to be carried out in parallel on GPU and it can achieve a good acceleration performance. This provides a very effective new way for people to use a low-cost personal computer to deal with large scale data and complex optimization problems efficiently. In the future study, the parallel computing of the new algorithm on GPU will be further optimized to improve its computational efficiency.

ACKNOWLEDGMENTS

This study was supported by the National Natural Science Foundation of China (Grant No. 11162002, 11462003, 11362003), Guangxi "Bagui Scholar" Teams for Innovation and Research Project and the Project of Promoting Young and Middle-aged Teachers' Basic Ability of Guangxi (Grant No. KY2016YB063).

REFERENCES

1. Hartigan, J.A. and M.A. Wong, 1979. Algorithm AS136: A K-means clustering algorithm. J. R. Stat. Soc. Series C: Applied Stat., 28: 100-108.
2. Eisen, M.B., P.T. Spellman, P.O. Brown and D. Botstein, 1998. Cluster analysis and display of genome-wide expression patterns. Proc. Natl. Acad. Sci. USA., 95: 14863-14868.
3. Tamayo, P., D. Slonim, J. Mesirov, Q. Zhu and S. Kitareewan *et al*., 1999. Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. Proc. Natl. Acad. Sci., 96: 2907-2912.
4. Bezdek, J.C., R. Ehrlich and W. Full, 1984. FCM: The fuzzy c-means clustering algorithm. Comput. Geosci., 10: 191-203.
5. Rahman, M.A. and M.Z. Islam, 2014. A hybrid clustering technique combining a novel genetic algorithm with K-Means. Knowl.-Based Syst., 71: 345-365.
6. Prabha, K.A. and N.K. Visalakshi, 2014. Improved particle swarm optimization based k-means clustering. Proceedings of the International Conference on Intelligent Computing Applications, March 6-7, 2014, Coimbatore, pp: 59-63.
7. Du, Z., Y. Wang and Z. Ji, 2008. PK-means: A new algorithm for gene clustering. Comput. Biol. Chem., 32: 243-247.
8. Ji, Z., H.L. Liao, W.H. Xu and L. Jiang, 2007. A strategy of particle-pair for vector quantization in image coding. Acta Electronica Sinica, 35: 1916-1916.
9. Kennedy, J., 2010. Particle Swarm Optimization. In: Encyclopedia of Machine Learning, Sammut, C. and G.I. Webb (Eds.). Springer, US., pp: 760-766.
10. Yang, X.S., 2010. Nature-Inspired Metaheuristic Algorithms. 2nd Edn., Luniver Press, USA., ISBN: 9781905986286, Pages: 160.
11. Von Neumann, J., 1951. The General and Logical Theory of Automata. In: Cerebral Mechanisms in Behavior: The Hixon Symposium, Jeffress, L.A. (Ed.). John Wiley, New York, pp: 1-31.
12. Shuai, D., Y. Dong and Q. Shuai, 2007. A new data clustering approach: Generalized cellular automata. Inform. Syst., 32: 968-977.
13. Shi, Y., H. Liu, L. Gao and G. Zhang, 2011. Cellular particle swarm optimization. Inform. Sci., 181: 4460-4493.

14. Senthilnath, J., S.N. Omkar and V. Mani, 2011. Clustering using firefly algorithm: Performance study. Swarm Evol. Comput., 1: 164-171.
15. Yan, B., Y. Zhang, Z. Yang, H. Su and H. Zheng, 2014. DVT-PKM: An improved GPU based parallel k-means algorithm. Proceedings of the 10th International Conference on Intelligent Computing, August 3-6, 2014, Taiyuan, China, pp: 591-601.
16. Hooda, H. and R. Nandal, 2014. Implementation of K-means clustering algorithm in CUDA. Int. J. Enhanced Res. Manage. Comput. Applic., 3: 15-24.
17. Serapiao, A.B.S., G.S. Correa, F.B. Goncalves and V.O. Carvalho, 2016. Combining k-means and k-harmonic with fish school search algorithm for data clustering task on graphics processing units. Applied Soft Comput., 41: 290-304.
18. Cheng, Y. and G.M. Church, 2000. Biclustering of expression data. Pcoc. Int. Conf. Intell. Syst. Mol. Biol., 8: 93-103.
19. Gasch, A.P., P.T. Spellman, C.M. Kao, O. Carmel-Harel and M.B. Eisen *et al*., 2000. Genomic expression programs in the response of yeast cells to environmental changes. Mol. Biol. Cell, 11: 4241-4257.
20. NVIDIA Corporation, 2010. NVIDIA CUDA compute unified device architecture programming guide, version 3.2. NVIDIA Corporation, California, USA.
21. Farivar, R., D. Rebolledo, E. Chan and R.H. Campbell, 2008. A parallel implementation of K-means clustering on GPUs. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, July 14-17, 2008, Las Vegas, Nevada, USA., pp: 340-345.