GPU Accelerated PK-means Algorithm for Gene Clustering

Wuchao Situ, Yau-King Lam, Yi Xiao, P.W.M. Tsang, and Chi-Sing Leung
Department of Electronic Engineering, City University of Hong Kong, Hong Kong, China

Abstract - *In this paper, a novel GPU accelerated scheme for the PK-means gene clustering algorithm is proposed. According to the native particle-pair structure of the PK-means algorithm, a fragment shader program is tailor-made to process a pair of particles in one pass for the computation-intensive portion. As the output channel of a fragment consisting of 4 floating-point values is fully utilized, overhead for each data points in searching for its nearest centroid throughout the particle-pair is reduced. Experimental evaluations on three popular gene expression datasets show that the proposed GPU accelerated scheme can attain an order of magnitude speedup as compared with the original PK-means algorithm.*

Keywords: Gene clustering, K-Means, PK-means, GPU

1 Introduction

Nowadays, gene clustering has attracted more and more attentions as the advancement of the technologies both in microarray [1] and computing. Microarray technology allows producing gene expression data in lower cost and monitoring large amount of data simultaneously for whole genome though a single chip only [2]. There is a huge of gene expression data produced in laboratory. To study the interactions among thousands of genes in the massive data sets, cluster analysis is the first and important step. An efficient cluster analysis is required to rapidly extract useful information from the raw data. Later advanced processing can be done further, such as protein structure prediction, biological network modeling, by using some natural computing [3].

There are numerous methods developed for clustering in the past [4-7], and among them K-means, with its simplicity and effectiveness, is perhaps the most popular one. However, due to its sensitivity to the initial condition, it is easy to get trapped in local optimal. To overcome this problem, recently a new clustering algorithm, known as PK-means, is proposed [8], which merges K-Means with the particle swarm optimization (PSO) [9, 10] algorithm. Experimental evaluation shows that it can reach better clustering results. However, its computation time is rather long, especially for large dimensional dataset. The bottleneck lies in the operation of K-means, which is a basic part of PK-means.

To overcome this problem, we propose to introduce the graphics process units (GPU), with its powerful stream processing units, to perform the tedious K-means operation. As the PK-means is working on particle-pairs, each particle-pair is packed together and fit to the programmable graphics pipeline [11] in GPU, where the two particles are together evaluated within a single fragment program. With the output channel of each fragment fully utilized, overhead for each data point in searching for its nearest centroid throughout the particle-pair is reduced. Organization of this paper is listed as follows. Section 2 gives a brief review of the PK-means algorithm. In section 3, we describe GPU accelerated scheme for PK-means. This is followed by the experimental evaluation on the proposed method in Section 4. Finally, a conclusion summarizing the essential findings is drawn in Section 5.

2 The PK-means clustering algorithm

The PK-means clustering method is the integration of the particle-pair optimizer (PPO) [12] and the well-known K-means. The former is a variation of the traditional particle swarm optimization (PSO) algorithm, while introducing a smaller swarm size based on particle pairs. For the clustering problem, a particle's position is a set of *K* cluster centroids, each of which is a *D*-dimensional vector, i.e.,

$$X_{i,n} = \left(x_{i,n,1}, x_{i,n,2}, \cdots, x_{i,n,K}\right), \quad (1)$$

where *n* is the iteration number. The velocity vector of this particle towards its next position is denoted by

$$V_{i,n+1} = wV_n + C_1 r_1 \left(p_{i,n} - X_{i,n}\right) + C_2 r_2 \left(Gbest - X_{i,n}\right), \quad (2)$$

where w is the inertia weight; $p_{i,n}$ is the best position for the particle '*i*' recorded so far; $Gbest$ represents the globally best position for the whole swarm throughout history; C_1 and C_2 are called acceleration factors; r_1 and r_2 are two random numbers within [0,1) . With the velocity vector available, the particle updates its position by

$$X_{i,n+1} = X_{i,n} + V_{i,n+1}. \quad (3)$$

To begin with, an initial swarm of four randomly generated particles is created and partitioned into two particle-pairs: {P1, P2} and {P3, P4}, as shown in Fig 1. Each particle pair evolves independently. Particles in each pair update their positions and velocity according to Eqs. (2) and (3), and perform K-means to update and evaluate its fitness. After a certain number of iterations, two particles (denoted as EP1 and EP2) with the better fitness values in their respective particle-pair are selected and combined together to form an elitist particle-pair {EP1, EP2}. The latter will continue to evolve and finally the particle EP3 with a better fitness value as the winner of {EP1, EP2} will represent the final solution.

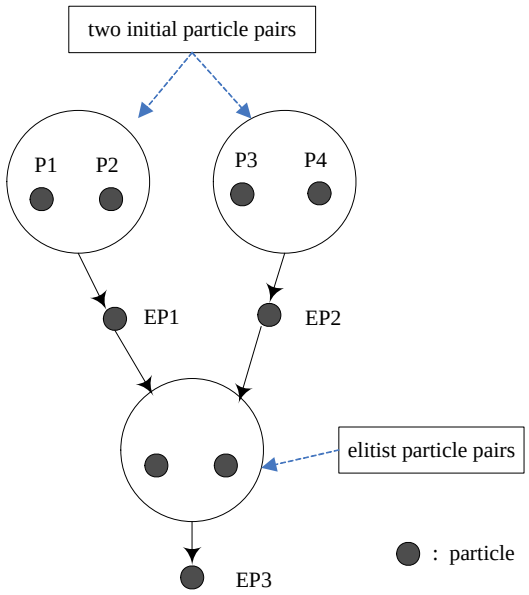

Fig 1. The evolution model of the Particle-Pair Optimizer.

3 The proposed GPU scheme for PK-means clustering

3.1 Overview of the GPU accelerated PK-means

The majority of computational cost in the PK-means algorithm lies in the operations of K-means for each particle-pair, and hence becomes the bottleneck of the algorithm. In view of this, we propose to convert this tedious step into a programmable graphics pipeline, where the Cg fragment shader program is tailor-made to perform the K-means operations for each particle-pair. An overview of the integration of GPU and PK-means algorithm is depicted in Fig. 2, where the building block "GPU accelerated K-means for particle-pair", performing K-means for two particles, will be explained in detail in next subsection.

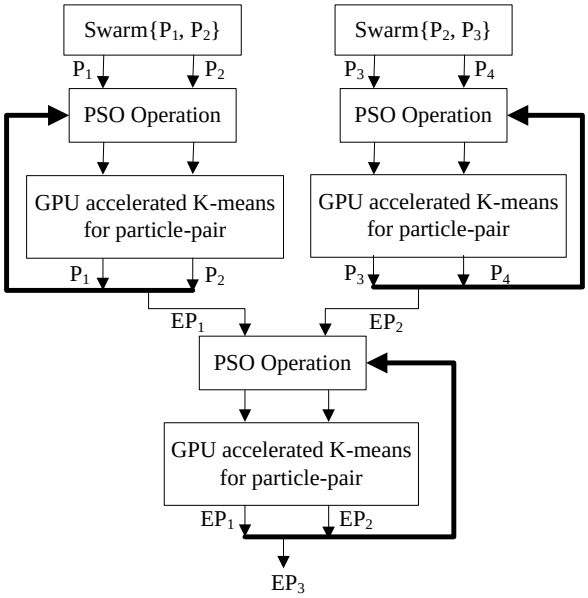

Fig 2. Overview of GPU accelerated PK-means

3.2 Fragment shader program based GPU Implementation of K-means for particle-pair

In the proposed GPU accelerated scheme, membership assignment for the particle-pair is conducted in the GPU while particle-pair updates (centroids updates) and the Mean Squared Error (MSE) calculation are carried out in the CPU. Fig. 3 gives the architecture of the GPU accelerated scheme.

Due to the large dimensionality of datasets we are dealing with (e.g. 77 for the Yeast cell-cycle data), both the data vectors and the particle-pair (each particle containing a set of K centriods) are stored in GPU textures, serving as look-up tables in the fragment shader program. Since a texture can store four single precision floating-point values (RGBA) per texel, a D-dimensional vector occupies $\lceil D/4 \rceil$ texels.

In the fragment shader program, each data vector is addressed by each fragment, and a pair of its nearest centroids is found in the two particles, respectively. Consequently, a pair of clustering memberships (each consisting of the ID of the nearest centroid and the nearest distance) is formed and rendered to the render texture. The latter is then downloaded to the CPU side where the particle-pair is updated and the MSEs are computed. Next round of iteration will be triggered from the CPU and the shader program repeats until a certain number of iterations has elapsed. Table 1 gives the Cg codes and Table 2 lists the notations for the fragment shader program.

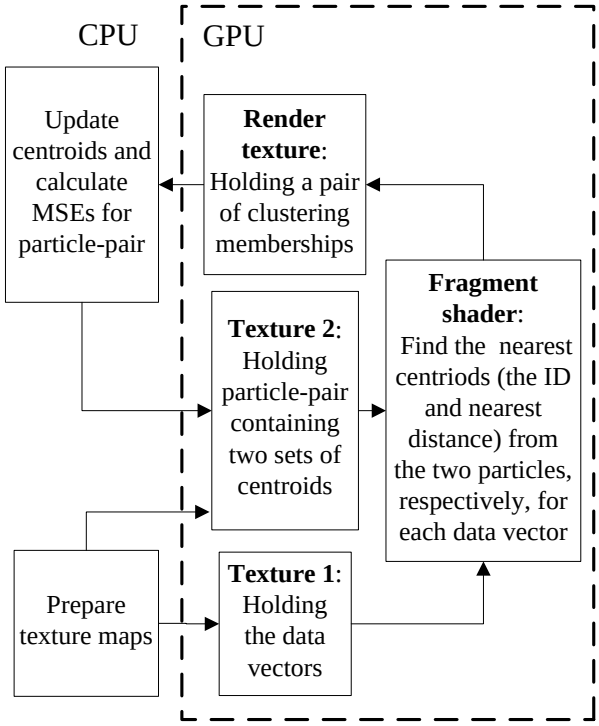

Fig 3. Overview of GPU accelerated K-means for particle-pair

Table 2 Notations for the fragment shader program in Table 1

Variable	Description
w	Equal to $\lceil D/4 \rceil$.
srctex	Texture holding the data vectors.
booktex	Texture holding the centroids.
index.y	The index of a vector in the data set.
codebook_size	Number of clusters/centroids.
minIndex1, minIndex2	The two IDs of the nearest centroid in the particle-pair, respectively.
mindist1, mindist2	Distances of a vector to its nearest centroids in the particle-pair, respectively.
memberships	A pair of clustering memberships, each consisting of the ID of the nearest centroid and the nearest distance.

Table 1 Fragment shader program - forming a pair of clustering memberships of the particle-pair for each data vector

```
#define num_of_centroids 256
#define FLT_MAX 3.402823466e+38F

void cgKMeans(
   float2 index: TEXCOORD0,
   uniform samplerRECT srctex,
   uniform samplerRECT booktex,
   out float4 memberships: COLOR0
){
  float mindist1, distance1, mindist2, distance2;
  float4 dn, dt;
  int i, k, minIndex1, minIndex2;
  minIndex1=minIndex2=-1;
  mindist2=mindist1=FLT_MAX;

  for ( k=0; k<num_of_centroids; k++)
  {
     distance2=distance1=0;
     for ( i=0; i<w; i++ ) // for all dimensions in a vector
     {
        // position of the data vector
        dn=texRECT(srctex,
             float2(i+(index.x-0.5)*w+0.5, index.y ) );

        // distance to the k-th centroid in particle 1
        dt= dn - texRECT( booktex, float2(i+0.5, k+0.5) );
        distance1 += dot(dt, dt);

        // distance to the k-th centroid in particle 2
        dt= dn - texRECT( booktex,
                  float2(i+0.5, k+num_of_centroids+0.5) );
        distance2 +=dot(dt, dt);
     }

     if ( distance1<mindist1 ) // for particle 1
     {
        mindist1=distance1;  //minimum distance
        minIndex1=k;  // ID of the nearest centroid
     }
     if ( distance2<mindist2 ) // for particle 2
     {
        mindist2=distance2;
        minIndex2=k;
     }
  }

  // output the pair of clustering memberships
  memberships=float4 (minIndex1, mindist1,
                      minIndex2, mindist2);
}
```

4 Experimental evaluation

The proposed scheme is evaluated with three popular gene expression datasets: Yeast cell-cycle [13] with 77 dimensions, Lymphoma [14] with 96 dimensions and Sporulation [15] with 7 dimensions. They have over 5 thousands, 4 thousands and 6 thousands of genes, respectively.

Performance of the GPU accelerated PK-means method is compared against the same PK-means algorithm implemented without GPU (referred to as the parent scheme). Both methods are applied to cluster each of the dataset into 256 clusters. To obtain reliable statistics, a total of 10 repeated trials for the three datasets are conducted. All the evaluations are based on the CPU (Intel Core2 Duo E6550 2.33GHz) and GPU (NVIDIA GTX260). The results of average computation time (in second) taken to reach convergence for both methods, are listed in Table 3. It can be seen that the GPU accelerated PK-means is at least 11 times faster than the parent scheme, and for the lower-dimensional dataset (i.e. Sporulation), over 20 times' speedup can be noted.

Table 3. Average computation time for the parent scheme and proposed scheme (Speed-up ratio: time of parent scheme / time of proposed GPU scheme)

Gene dataset	Scheme	Time (Sec)	Speed-up ratio
Yeast cell-cycle	Parent scheme	78.6	11.2
	GPU scheme	7.0	
Lymphoma	Parent scheme	68.4	11.4
	GPU scheme	6.0	
Sporulation	Parent scheme	16.7	20.5
	GPU scheme	0.8	

5 Conclusions

Gene cluster analysis plays an important role in discovering the function of gene. K-means is one of the well-known clustering methods for its simplicity and effectiveness. However, due to its sensitivity to the initial clustering, it is prone to be trapped in a local minimum. Recently, an enhanced clustering method, known as PK-means, which incorporates K-means with the particle swam optimization is developed. Despites its success in finding better clustering results, the process is usually too time-consuming. The bottleneck lies in the K-means operation which is a basic portion of PK-means. To address the shortcoming, this paper proposes a novel GPU accelerated scheme for the PK-means algorithm. Based on the particle-pair structure of PK-means, each particle-pair is packed together and fit to a tailor-made fragment shader program, where a pair of clustering membership is formed for the particle-pair and then sent to the entire output channel of each fragment. As the latter is fully utilized, overhead is reduced. Experimental evaluation on three gene expression datasets reveals that the proposed GPU accelerated scheme can attain an order of magnitude speedup as compared with the parent scheme.

6 References

[1] P. Brown, D. Botstein, "Exploring the New World of the Genome with DNA Microarrays"; Nature Genetics, Vol. 21, 33-37, 1999

[2] A. Brazma, A. Robinson, G. Cameron, M. Ashburner. "One-stop Shop for Microarray Data"; Nature, Vol. 403, 699-700, 2000

[3] F. Masulli, S. Mitra. "Natural computing methods in bioinformatics: A survey"; Information Fusion, Vol. 10, issue 3, 211–216, 2009.

[4] R. Shamir, R. Sharan. "Algorithmic approaches to clustering gene expression data". Current Topics in Computational Biology, MIT Press, Cambridge, MA, pp. 269–299, 2002

[5] M.B. Eisen, P.T. Spellman, P.O. Brown, D. Botstein. "Cluster analysis and display of genome-wide expression patterns"; PNAS, Vol. 95, 14863–14868, 1998.

[6] T. Kohonen; "The self-organizing map"; Proc. IEEE 78, 1464–1480, 1990.

[7] J.C. Bezdek, R. Ehrlich, W. Full. "FCM: the Fuzzy c-means clustering algorithm"; Comput. Geosci. Vol. 10 issue 2–3, 191–203, 1984.

[8] Z. Du, Y. Wang, Z. Ji. "PK-Means: A new algorithm for gene clustering"; Comput. Biol. Chem., Vol. 32, issue 4, 243-247, 2008.

[9] R. Eberhart, J. Kennedy. "A new optimizer using particle swarm theory"; In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science. IEEE Service Center, Nagoya, Japan, pp. 39–43. 1995.

[10] J. Kennedy, R. Eberhart. "Particle swarm optimization"; In: Proceedings of IEEE International Conference on Neural Networks. IEEE Service Center, Piscataway, NJ, pp. 1942–1948, 1995.

[11] R. Fernando, M. J. Kilgard. “The Cg tutorial: the definitive guide to programmable real-time graphics”. Addison-Wesley, 2003.

[12] Z. Ji, H., Liao, W. Xu, L. Jiang. “A strategy of particle-pair for vector quantization in image coding”; Acta Electron. Sin., Vol. 35, issue 7, 86-89, 2007.

[13] P.T. Spellman, et al. “Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization”; Mol. Biol. Cell, Vol. 9, 3273–3297, 1998.

[14] A.A. Alizadeh, et al. “Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling”; Nature, Vol. 403, 503–511, 2000.

[15] S. Chu, et al. “The transcriptional program of sporulation in budding yeast”; Science, Vol. 282, 699–705, 1998.