Journal of
Applied Remote Sensing

Fast supervised hyperspectral band selection using graphics processing unit

Wei Wei
Qian Du
Nicolas H. Younan

SPIE

Fast supervised hyperspectral band selection using graphics processing unit

Wei Wei, Qian Du, and Nicolas H. Younan
Mississippi State University, Department of Electrical and Computer Engineering
Mississippi State, Mississippi 39762
du@ece.msstate.edu

Abstract. Band selection is a common technique to reduce the dimensionality of hyperspectral imagery. When the desired object information is known, the reduction process can be achieved by selecting the bands that contain the most object information. It is expected that these selected bands can offer an overall satisfactory detection and classification performance. In this paper, we propose a new particle swarm optimization (PSO) based supervised band-selection algorithm that uses the known class signatures only without examining the original bands or the need of class training samples. Thus, this method requires much less computing time than other traditional methods. However, the PSO process itself may introduce additional computation cost. To tackle this problem, we propose parallel implementations via emerging general-purpose graphics processing units that can provide satisfactory results in speedup when compared to the cluster-based parallel implementation. © *2012 Society of Photo-Optical Instrumentation Engineers (SPIE).* [DOI: 10.1117/1.JRS.6.061504]

Keywords: band selection; particle swarm optimization; hyperspectral imagery; graphics computing units.

Paper 11164SS received Sep. 1, 2011; revised manuscript received Feb. 14, 2012; accepted for publication Mar. 15, 2012; published online May 4, 2012.

1 Introduction

The objective of hyperspectral band selection is to select a subset of original bands that contain the major data information. Band selection can be performed based on the availability of class information. When the interested class information has been obtained, we could apply supervised band selection to preserve the desired object information. When such information is unknown, unsupervised band selection has to be implemented to find the most informative and distinctive bands. In this paper, we will only discuss supervised band selection.

A large group of supervised band-selection algorithms calculates class separability when a subset of bands is selected. Class separability could be measured through various ways, i.e., divergence, transformed divergence (TD), Bhattacharyya distance, or Jeffries-Matusita (JM) distance, with the general criterion of the largest class separability being achieved by a smallest subset.[1–4] Under this circumstance, enough samples are usually required in order to examine class statistics. There are also other selection criteria that employ the spectral angle mapper (SAM) or orthogonal projection divergence (OPD),[5] where the average pairwise class distance is set as the band selection metric.

To avoid testing all the possible combinations, sequential forward selection (SFS) is often used for band searching. It is initiated with the first selected band, then the second band is selected such that it (together with the first one) can yield with the optimal value of a selection criterion, and so on. In this paper, we propose a more advanced searching method, i.e., particle swarm optimization (PSO). The PSO uses a simple mechanism that mimics swarm behavior in birds flocking and fish schooling to guide the particles to search for global optimal solutions. It is proved to be a very efficient optimization algorithm by searching the entire problem space.[6–8] Some researchers have explored the performance of PSO in band selection area,[9,10] where the number of particles is linked with the predefined number of bands to be selected, and the particles' positions in search

0091-3286/2012/$25.00 © 2012 SPIE

space indicate the indices of selected bands. In Ref. 9, a function related to greedy modular eigenspaces is used as the fitness function, which needs to examine the entire dataset; in Ref. 10, a neural network is embedded in the PSO training, which is very time-consuming.

The fitness function in a PSO system should be as simple as possible to avoid large computational burden. If we set the fitness function as the subset which provides the highest classification accuracy, this criterion function of band selection is not practical because classification needs to be conducted each time. It is computationally prohibitive if the selected classifier, for example, neural network or support vector machine (SVM), is very expensive with respect to training and testing. Here, we apply the minimum estimated abundance covariance (MEAC) method[11] which is simple but effective without performing classification during band selection, without calculating class statistical information using training samples, and without examining original bands (or band combinations). MEAC selects bands based on interested class spectral signatures only (one for each class).

Although PSO algorithms present attractive global optima search properties, they are plagued by high computational cost as measured by running time. It is natural to implement parallel computing for such an optimization system. Clusters are commonly used nowadays for high performance computing purpose. Recently, graphics computing units (GPUs) are attracting more and more attention from engineering and science area due to its portability and low-cost. GPUs were first invented for graphics acceleration, but its computational power for general purpose computing has also been explored.[12] GPU has already been successfully used in many computation fields, such as signal process problems,[13] computer vision problems,[14] Voronoi diagrams,[15] neural network computation,[16] and so on. It is also applied to hyperspectral image analysis, e.g., detection, classification, and unmixing.[17–19] Several GPU based PSO implementations have been developed. Veronese and Krohling[20] are the first to develop the PSO parallel implementation on GPU using compute unified device architecture (CUDA) platform. Zhou and Tan[21] have presented a model for PSO with local ring topology. Mussi et al.[22] implemented both local ring topology and global topology and applied their PSO model to detect road signs. Zhu and Curry[23] presented a hybrid PSO model including CPU and GPU codes on particle solution updates. In this paper, we will present GPU implementation with the latest GTX285 for the PSO-based band selection algorithm with the MEAC criterion and compare its performance with that of the cluster implementation.

2 Band Selection Method

2.1 *Theory Background*

In a supervised band selection situation where interested class signatures are known, band selection process can be greatly simplified. It saves a lot of computation time by utilizing only the class signatures rather than entire bands for band selection.

Assume that there are p classes present in an image scene with L bands. Based on the linear-mixture model, a pixel (r) can be considered as the mixture of the p endmembers. Let the end member matrix be $S = [s1, s2, \dots, \mathrm{sp}]$. So the pixel r can be expressed as

$$r = \mathbf{S}\alpha + n, \tag{1}$$

where $\boldsymbol{\alpha} = (\alpha_1 \alpha_2 \alpha_p)^T$ is the abundance vector and n is the uncorrelated whitened noise with statistical character of $E(\mathbf{n}) = 0$ and $\mathrm{Cov}(\mathbf{n}) = \sigma^2\mathbf{I}$ where I is identity matrix. The least square solution of α, denoted as $\hat{\boldsymbol{\alpha}}$, can be obtained as[24]

$$\hat{\boldsymbol{\alpha}} = (\mathbf{S}^T\mathbf{S})^{-1}\mathbf{S}^T r. \tag{2}$$

The stochastic features of $\hat{\boldsymbol{\alpha}}$ include

$$E(\hat{\boldsymbol{\alpha}}) = \boldsymbol{\alpha} \qquad \mathrm{Cov}(\hat{\boldsymbol{\alpha}}) = \sigma^2(\mathbf{S}^\mathbf{T}\mathbf{S})^{-1}. \tag{3}$$

In the scenario where q classes exist and $q > p$, which means that only p class signatures are known, then the noise n in Eq. (1) is not white any more. Instead, $\mathrm{Cov}(\mathbf{n}) = \sigma^2\Sigma$ where

Σ is the noise covariance matrix. In this case, the abundance of the p classes can be estimated using the weighted least square solution as

$$\hat{\boldsymbol{\alpha}} = (\mathbf{S}^{\mathbf{T}}\Sigma^{-\mathbf{1}}\mathbf{S})^{-\mathbf{1}}\mathbf{S}^{\mathbf{T}}\Sigma^{-\mathbf{1}}\mathbf{r}. \tag{4}$$

According to Ref. 24, the first- and second-order moments of $\hat{\boldsymbol{\alpha}}$ become

$$E(\hat{\boldsymbol{\alpha}}) = \boldsymbol{\alpha} \qquad \mathrm{Cov}(\hat{\boldsymbol{\alpha}}) = \sigma^2(\mathbf{S}^{\mathbf{T}}\Sigma^{-\mathbf{1}}\mathbf{S})^{-1}. \tag{5}$$

2.2 *Fitness Function and SFS Searching*

Intuitively, the subset of selected bands should reduce the deviation of $\hat{\boldsymbol{\alpha}}$ from the actual $\boldsymbol{\alpha}$. When all the classes are known, this is equivalent to minimizing the trace of the covariance, i.e.,

$$\mathrm{trace}[(\hat{\mathbf{S}}^{\mathbf{T}}\hat{\mathbf{S}})^{-1}]. \tag{6}$$

Let Φ^S denote the selected band subset, and then $\hat{\mathbf{S}}$ is the matrix containing class spectral signatures in Φ^S. If the number of existed classes is larger than that of known, it is equivalent to minimize

$$\mathrm{trace}[(\hat{\mathbf{S}}^{\mathbf{T}}\hat{\Sigma}^{-\mathbf{1}}\hat{\mathbf{S}})^{-1}], \tag{7}$$

where $\hat{\Sigma}$ is the data covariance matrix with the selected bands in Φ^S only. The resulting band-selection algorithm based on the equations above is referred to as the MEAC method. It is chosen as the fitness function for our PSO-based band selection due to its robust performance.

The basic steps of the traditional SFS searching using MEAC for band selection can be described as:

1. initializing the algorithm by choosing a pair of bands B_1 and B_2, then $\Phi_2^S = \{B_1, B_2\}$,
2. finding a third band B_3 such that Eq. (6) or Eq. (7) can be minimized, then the selected band subset is updated as $\Phi_3^S = \Phi_2^S \bigcup \{B_3\}$, and
3. continuing on step 2) until a desired number of bands has been added to Φ^S.

2.3 *PSO Searching for Band Selection*

PSO has been widely used in multi-dimensional optimization. It is one of the evolutionary algorithms targeted especially towards global optimization. PSO searches the solution space by starting from randomly distributed particles like swarm. It is very similar to other evolutionary computation algorithms but has relatively fast convergence. It shares some characteristics with evolutionary techniques: 1) It uses a large size of random particles as initials; 2) the optimal objective function (i.e., fitness function) value is determined by iteratively updating the generations; and 3) evolution adaptation uses the previous generations, and particles are flown through the problem space following the current best solution. For our problem, the predefined number of bands equals the number of particles. Each particle's position represents the index of that selected band.

Here, possible solutions (i.e., selected bands) are called particles, and recursive solution update is called velocity. The update of particles is stated by Eq. (8). It calculates the new velocity V_{id} for each particle based on the previous velocity V_{id}, particle's current location x_{id}, the location (p_{id}) that it has reached so far for the objective function Eq. (6) or Eq. (7), and the particle's location (p_{gd}) that has been reached so far globally for the objective function. These particles are all potential solutions, and their locations are updated by Eq. (9). Two learning rates are c_1 and c_2. The inertia weight $w = \{0.5 + [\mathrm{rand}(\bullet)/2.0]\}$ is used as the scaling factor of previous velocity V_{id}, which provides improved performance in various applications.[7,8] Here, rand($\bullet$) represents a function generating a random variable uniformly distributed within [0, 1].

$$V_{\mathrm{id}} = w \times V_{\mathrm{id}} + c_1 \times \mathrm{rand}(\bullet) \times (p_{\mathrm{id}} - x_{\mathrm{id}}) + c_2 \times \mathrm{rand}(\bullet) \times (p_{\mathrm{gd}} - x_{\mathrm{id}}). \tag{8}$$

$$x_{\mathrm{id}} = x_{\mathrm{id}} + V_{\mathrm{id}}. \tag{9}$$

2.4 *Performance Evaluation*

When pixel-level ground truth is unavailable, the classification maps from using all the original bands can be considered as ground truth, and those from the selected bands are compared with them with spatial correlation coefficient ρ. An average ρ closer to one generally means better performance. This is under the assumption that using all the original spectral bands (after bad-band removal), the best or, at least, satisfying classification performance can be provided. This method measures the image similarity and in turn provides us the assessment of quantitative performance in an unsupervised situation.

3 GPU Implementations

The GPU comes with the shared memory architecture. As all processors of the GPU can share data within a global address space, it fits the data parallelism very well. To achieve satisfied parallel performance, the data throughput is very critical in GPU parallel algorithm design, which means enough data and computation should be designed ahead to feed into the GPU to take advantage of its computing power. Previous work shows that it can achieve excellent speedup performance only when the data size is increased to thousands. As it uses the shared memory model, the major bottleneck is memory communication between the host and device; unnecessary data transfer between host and device should be avoided. After all, two key rules should be followed in the parallel algorithm design stage: (1) reduce the communication between host and device, and (2) parallel data size as large as possible.

In this paper, our purpose is to accelerate the running speed of MEAC-based PSO searching method on GPU; meanwhile, performances of the GPU-implemented PSO should not be deteriorated. By exploring the full power of the parallel computing ability of GPU, we expect the implementation can solve the global optimization problem for high-dimensional data with large swarm population.

3.1 *Data Organization*

In PSO, the information of position and velocity for all the particles is stored on the global memory of GPU chips. One-dimensional arrays are used for storing parameters, including position, velocity, pbest and gbest fitness values for all the particles. Here, we assume the dimension of the problem is D (equal to the number of bands to be selected), and the swarm population is N. So an array of length DN is used to represent each swarm by storing all the position and velocity values. The pbest fitness and fitness value is stored on an array of length D.

3.2 *Random Number Generation*

In the process of optimization, PSO requires random numbers for velocity updating. Three random numbers are needed during each iteration. One is for the inertia weight, and two are for the learning rates. As the absence of high precision integer arithmetic, generating random numbers in GPUs is not easy, we generate random numbers on CPU first and then transfer these numbers to the global memory of GPU. However, the data transportation between GPU and CPU is quite time consuming. If we generate random numbers on CPU for each iteration and then transfer them to GPU, the speedup performance will be degraded. In order to reduce the communication time between CPU and GPU as much as possible, we would transfer the random numbers in advance. First, we generate T random numbers on CPU before running PSO where T is large enough as necessary in the predefined iterations. Then, they are transported to GPU global memory and stored in an array R. When it comes to the update of the velocity, we pass three random numbers from R instead of transporting three times of max iterations random numbers from CPU to GPU. The running speed can be obviously improved by using this technique. In the experiment, we make a comparison with the cellular automata (CA) random number generator (RNG) approach[25] that generates random numbers in GPU itself.

3.3 *Overall Algorithm of GPU-Implemented PSO*

The main steps illustrated in Fig. 1 for GPU-implemented PSO can be described as below. Here, we set the maximum number of iterations as the stopping criterion for the optimization process based on our experience:

```
1. Initialize the positions and velocities of all particles.
2. Transfer these data and hyperspectral data from CPU to GPU's global memory.
3. For $i = 1$ to Max Iteration do
    Compute fitness values of all particles using Eq. (6) or Eq. (7)
    Update pbest and pbest position of each particle
    Update gbest and gbest position for all the particles
    Update velocity and position of each particle using Eqs. (8) and (9).
  end for
  Transfer results data back to CPU and output.
```

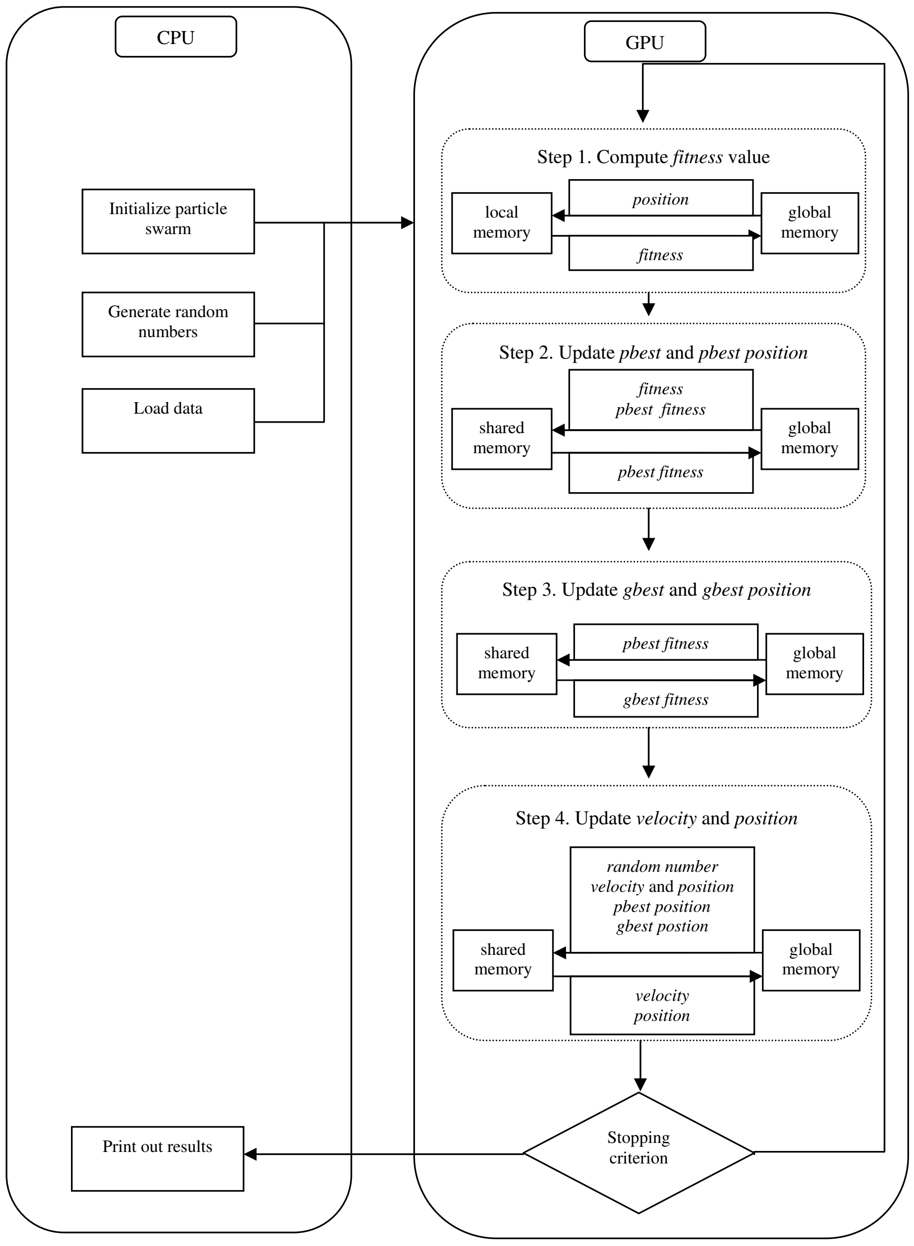

Fig. 1 The Diagram for the GPU-implemented PSO band selection.

3.4 *Parallelization Design On GPU*

The difference between the implementation on CPU and a GPU kernel is that the kernel function of GPU is designed for single-instruction, multiple-data (SIMD) parallelized computing. So we design the parallelization methods for all the sub-processes in PSO.

1. Compute fitness values: Fitness values calculation is the most important task in the entire process, where the computation intensity is determined by the number of particles and the size of each particle. It should be carefully designed for parallelization so as to improve the overall efficiency of the algorithm. The steps for fitness value calculation are shown as follows:
   a. Set the block size and grid size with the number of threads equal to the number of particles N.
   b. Load the position data of each particle from global memory to local memory of each thread.
   c. Apply arithmetical operations to each thread for fitness function in parallel.
   d. Store the final fitness values of all particles to an array fitness which stores the fitness value for each particle.
2. Update pbest and gbest: After the fitness values are computed, each particle may result in a better value than ever before in its history and new global best particles may be found. So pbest and gbest must be updated according to the current information of the particles. The updating of pbest can be done as follows:
   a. Transfer pbest position, pbest fitness, and fitness data from global to shared memory of each block.
   b. Map each thread to each particle.
   c. If fitness value of any thread is better than its pbest fitness, then the new fitness value replaces the old one for each dimension d do.
   The update of gbest is different from that of pbest. Its parallel implementation is shown as below:
   a. Transfer pbest fitness data from global to shared memory.
   b. Apply the reduction on each block for minimum element; store the minimum elements of each block to one array.
   c. Apply the reduction again for the array we got in step b.
   d. Update gbest fitness and gbest position by one thread.
3. Update velocity and position: After the pbest and gbest position of all the particles have been updated, the velocities and positions should also be updated according to Eqs. (8) and (9), respectively. The update is critical in the whole algorithm which makes use of the new information provided by pbest and gbest.
   a. Map each thread to each particle.
   b. For each dimension d do

Transfer the gbest position, pbest position, position, and random number to share memory of each block.

Update each particle on dimension d.
end for.

4 Experiments

4.1 *Computing Facilities and Dataset*

The CPU machine used in the experiments is an Intel Pentium4 3.40 GHz with Hyper thread and 2 GB of memory. The GPU is NVidia's GeForce GTX285 that has 240 cores with 1 GB memory. The Linux-based cluster used in the experiments has 2048 cores which is composed of 512 Sun Microsystems SunFire X2200 M2 servers, each with two dual-core AMD Opteron

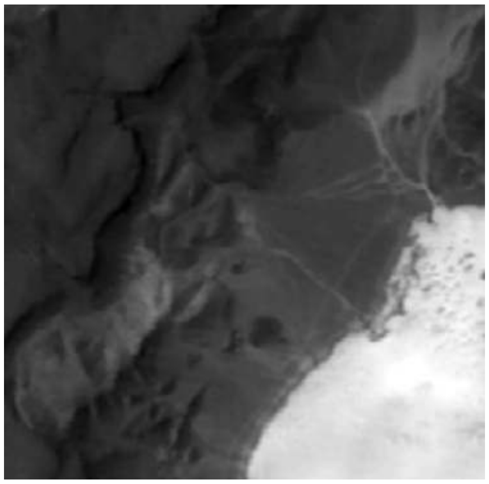

Fig. 2 AVIRIS lunar lake scene.

2218 processors (2.6 GHz) and 8 GB of memory. All of the computing nodes are diskless. The system uses gigabit ethernet to connect the 32 nodes in each rack together and 10 GbE to connect the 16 racks to one another. The parallel algorithms on the cluster are implemented in the C++ with the message passing interface (MPI) and EIGN library. The GPU versions are implemented in the CUDA.

The airborne visible/infrared imaging spectrometer (AVIRIS) data used in this experiment, as shown in Fig. 2, was taken from the Lunar Crater Volcanic Field in Northern Nye County, Nevada, with 200×200 pixels and 158 bands after low-SNR and water-absorption bands were removed. The spatial resolution is about 20 m, which means that a significant number of pixels are mixed pixels. According to prior information, there are five classes: cinders, playa, rhyolite, shade, vegetation.

Since all five spectral signatures are available, we could use them for band selection. The OSP classifier was chosen for soft classification,[26] and spatial correlation coefficient was calculated between the corresponding classifications maps using all the original bands and the selected bands. The averaged correlation coefficient versus the number of selected bands was plotted in Fig. 3. It also shows that the performance of PSO is better than the widely used SFS.

In this experiment, we set the number of particle swarm size of 10,000 to fully explore the computational power of GPU. The maximum iteration was set as 500 which was large enough in this case. The number of selected bands was changed from 5 to 25. The average running time in

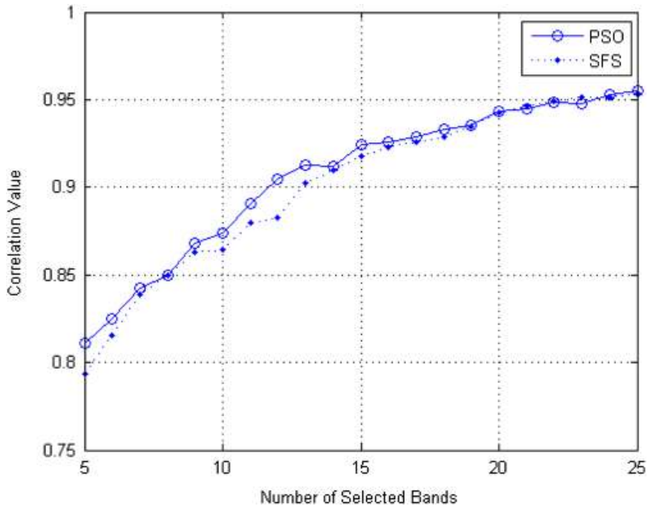

Fig. 3 Supervised band selection with MEAC metric for AVIRIS lunar lake experiment.

Table 1 Average parallel band selection running time (in second).

Number of Processors	1	4	8	16	32
Cluster	31.95	10.87	4.82	2.45	1.20
GPU	1.14				
GPU with CA PRNG	1.29				

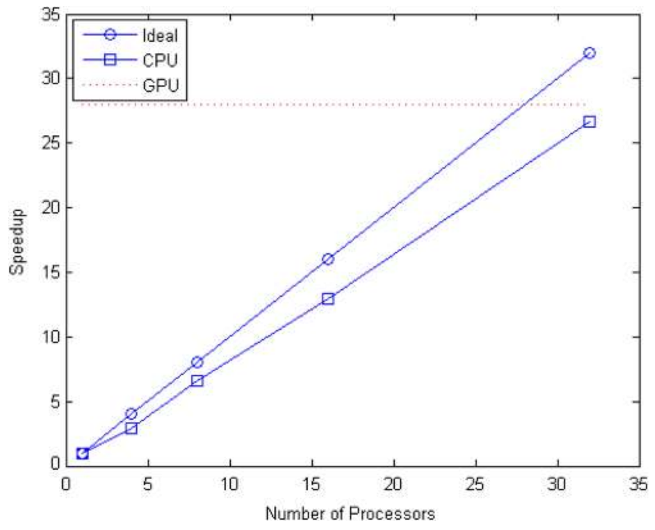

Fig. 4 Speedup of the parallel band selection algorithm.

the cluster and GPU is shown in Table 1, and the speedup comparison of both parallel computing is shown in Fig. 4. From these results, we can see that the GPU achieved slightly better performance than the cluster with 32 cores, and generating random numbers in CPU is slightly better than in GPU.

4.2 *Running Time and Speedup Versus Number of Selected Bands*

Now we fixed the swarm population to a constant number and varied the number of bands to be selected. Analysis about the relationship between running time (as well as speedup) and the number of selected bands was conducted. The parallel PSO was run five times, and the average results are shown in Table 2 ($N = 10000$, Iter $= 500$).

As seen from Fig. 5, the speedup of the cluster system has not been affected a lot by the number of bands selected, or the particle dimension. Figure 6 shows the speedup of GPU, and the accelerations were decreased as the number of selected bands increased. This is because when the number of selected bands is increased, it directly leads to the increase of particle dimension, which greatly degraded the speedup performance of GPU.

Table 2 further shows that the computation time all increased but that of GPU increased greatly in proportion compared with the serial algorithm. In other words, we could say the cluster has fast computation speed and speedy data transfer mechanism, thus the increase of particle dimension does not induce more computational burden.

4.3 *Running Time and Speedup Versus Swarm Size*

Now we fixed the number of selected bands to a constant number and explored the effect of swarm size on the parallel performance. Corresponding analysis of the results of running

Table 2 Results of parallel band selection running time (in second).

Number of selected band	5	10	15	20	25
1 processor	25.289	28.853	34.302	34.929	36.790
4 processor	8.710	9.664	11.169	11.2909	13.324
8 processor	3.833	4.312	4.857	5.444	5.854
16 processor	1.840	2.065	2.347	2.537	2.808
32 processor	0.981	1.078	1.214	1.329	1.427
GPU	0.609	0.805	1.139	1.504	1.649
GPU with CA PRNG	0.710	0.828	1.242	1.696	1.871

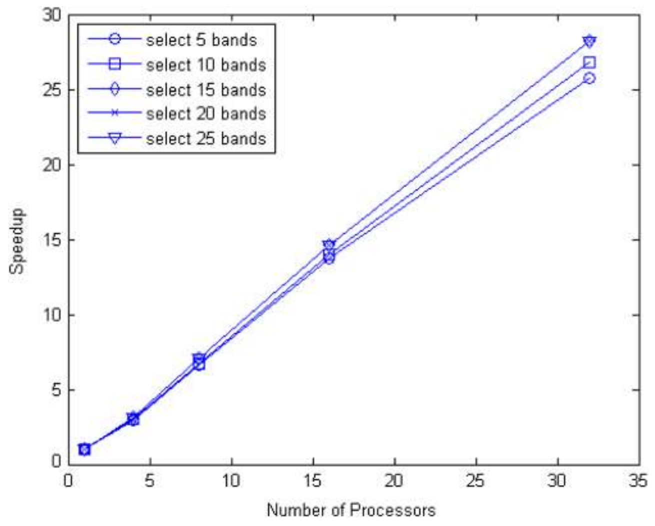

Fig. 5 Speedup performance with different number of band selected for cluster.

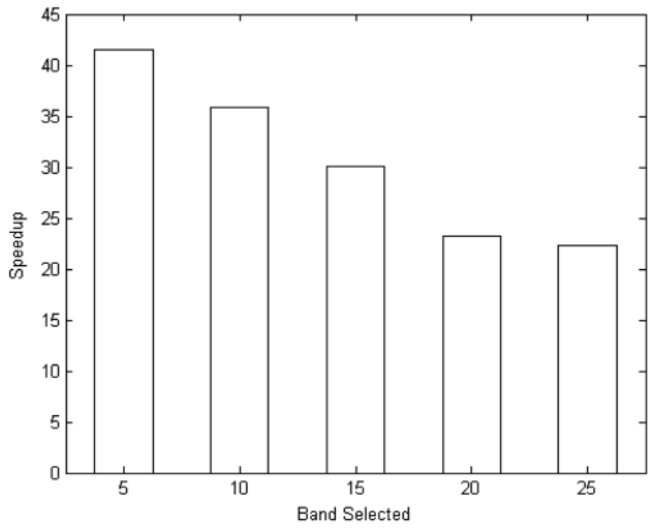

Fig. 6 Speedup performance with different number of band selected for GPU.

Table 3 Results of parallel band selection running time (in second).

Swarm size	10000	5000	1000
1 processor	36.790	18.845	3.976
4 processor	13.324	6.639	1.367
8 processor	5.854	2.962	0.646
16 processor	2.808	1.469	0.362
32 processor	1.427	1.017	0.262
GPU	1.649	1.084	0.519
GPU with CA PRNG	1.881	1.173	0.615

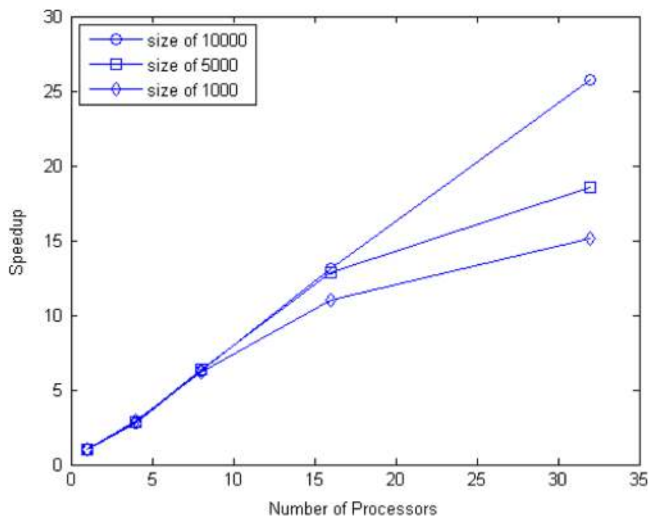

Fig. 7 Speedup performance with different swarm size for cluster.

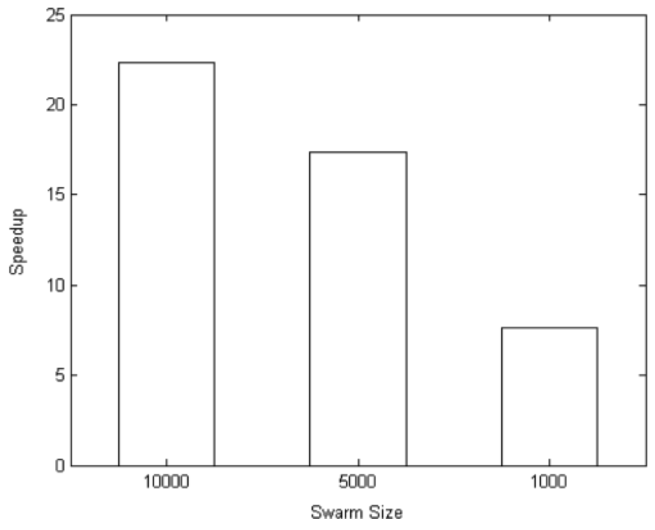

Fig. 8 Speedup performance with different swarm size for GPU.

time on different swarm size was performed. The parallel PSO was run five times, and the average results are shown in Table 3 (Dimension = 25, Iter = 500).

As we can see from Fig. 7, the speedup of the cluster system was largely affected by the parameter of swarm size in PSO, especially when the number of processors was increased. This illustrates that as the swarm size allocated on each processor was decreased, the overhead between the processors became dominant and thus reduced the related speedup. The speedup of GPU, as shown in Fig. 8, shows the accelerations also decreased as the swarm size was reduced which is, again, due to the communication overhead between the host and device.

The detailed running time of GPU is shown in Table 3. Although the decrease of swarm size directly brought down the computational intensity, the performance was also affected due to the proportion of the communication overhead which was enlarged in whole computation process.

5 Conclusions

In this paper, we propose GPU parallel implementation for MEAC-based supervised hyperspectral band selection. By employing the idea of optimization search method of PSO, the optimal combination of bands can be searched given that the number of selected bands is known. However, to reduce the workload of PSO, its implementation on GPU is presented, and its performance is compared with the cluster-based parallel implementation. The experimental results show that GPU implementation has high scalability and is comparable to cluster implementation. In addition, we notice that the running time and swarm population size take an approximately linear relationship, which is also true for running time and dimension. However, the swarm size has a major impact on the speedup performance; to fully explore the power of GPU, a large size of swarm should be adopted.

As for future work, we will further improve the performance of parallel PSO with several strategies including the parallel simulated annealing approach in Ref. 27.

References

1. R. Pouncey, K. Swanson, and K. Hart, *ERDAS Field Guide*, 5th ed., ERDAS Inc., Atlanta, GA (1999).
2. J. A. Richards and X. Jia, *Remote Sensing Digital Image Analysis: An Introduction*, 4th ed., Springer-Verlag, Berlin, Germany (2006).
3. A. Ifarraguerri and M. W. Prairie, "Visual method for spectral band selection," *IEEE Geosci. Remote Sens. Lett.* **1**(2), 101–106 (2004), http://dx.doi.org/10.1109/LGRS.2003.822879.
4. R. Huang and M. He, "Band selection based on feature weighting for classification of hyperspectral imagery," *IEEE Geosci. Remote Sens. Lett.* **2**(2), 156–159 (2005), http://dx.doi.org/10.1109/LGRS.2005.844658.
5. C. -I Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*, Kluwer, New York, NY (2003).
6. R. C. Eberhart and J. Kennedy, "A new optimizer using particle swam theory," in *Proc. of the Sixth International Symposium on Micromachine and Human Science*, Nagoya, Japan, pp. 39–43 (4–6 October 1995), http://dx.doi.org/10.1109/MHS.1995.494215
7. R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications, resources," in *Proc. of the Congress on Evolutionary Computation*, Indianapolis, IN, Vol. **1**, pp. 81–86 (2001), http://dx.doi.org/10.1109/CEC.2001.934374
8. X. Hu and R. C. Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization," in *Proc. of 6th World Multiconference on Systemics*, Cybernetics and Informatics (2002).
9. Y.-L. Chang et al., "Band selection for hyperspectral images based on parallel particle swarm optimization schemes," in *Proc. of IEEE Geosci. and Remote Sens. Symposium*, Cape Town, Vol. **5**, pp. V84–V87 (12–17 July 2009), http://dx.doi.org/10.1109/IGARSS.2009.5417728

10. S. T. Monteiro and Y. Kosugi, "A particle swarm optimization-based approach for hyperspectral band selection," in *Proc. of the Congress on Evolutionary Computation*, Singapore, pp. 3335–3340 (25–28 Sept. 2007), http://dx.doi.org/10.1109/CEC.2007.4424902
11. H. Yang et al., "An efficient method for supervised hyperspectral band selection," *IEEE Geosci. Remote Sens. Lett.* **8**(1), 138–142 (2011), http://dx.doi.org/10.1109/LGRS.2010.2053516.
12. M. D. McCool, "Signal processing and general-purpose computing on GPUs," *IEEE Signal Process. Mag.* **24**(3), 109–114 (2007), http://dx.doi.org/10.1109/MSP.2007.361608;
13. C. Tenllado et al., "Parallel implementation of the 2D discrete wavelet transform on graphics processing units: filter bank versus lifting," *IEEE. Trans. Parallel. Distr. Syst.* **9**(3), 299–310 (2008), http://dx.doi.org/10.1109/TPDS.2007.70716.
14. R. Yang and G. Welch, "Fast image segmentation and smoothing using commodity graphics hardware," *J. Graph. Tool.* **7**(4), 91–100 (2002).
15. K. E. Hoff et al., "Fast computation of generalized Voronoi diagrams using graphics hardware," in *Proc. of SIGGRAPH*, 277–286, ACM Press/Addison-Wesley Publishing Co, New York, NY (1999), http://dx.doi.org/10.1145/311535.311567
16. Z. W. Luo, H. Z. Liu, and X. C. Wu, "Artificial neural network computation on graphic process unit," in *Proc. of International Joint Conference on Neural Networks*, Wuhan, China, Vol. **1**, pp. 622–631 (31 July–4 Aug. 2005).http://dx.doi.org/10.1109/IJCNN.2005.1555903
17. J. Setoain et al., "Real-time onboard hyperspectral image processing using programmable graphics hardware," in *High Performance Computing in Remote Sensing*, A.Plaza and C.-I. Chang, Eds., Chapman & Hall/CRC, Boca Raton, FL (2008).
18. J. Setoain et al., "Parallel morphological endmember extraction using commodity graphics hardware," *IEEE Geosci. Remote Sens. Lett.* **4**(3), 441–445 (2007), http://dx.doi.org/10.1109/LGRS.2007.897398.
19. A. Paz and A. Plaza, "Clusters versus GPUs for parallel target and anomaly detection in hyperspectral images," *EURASIP J. Adv. Signal Process.* **2010** (2010), http://dx.doi.org/10.1155/2010/915639
20. L. de P. Veronese and R. A. Krohling, "Swarm's flight: accelerating the particles using C-CUDA," in *Proc. of IEEE Congress on Evolutionary Computation*, Trondheim, pp. 3264–3270 (18–21 May 2009).http://dx.doi.org/10.1109/CEC.2009.4983358
21. Y. Zhou and Y. Tan, "GPU-based parallel particle swarm optimization," in *Proc. of IEEE Congress on Evolutionary Computation*, Trondheim, pp. 1493–1500 (18–21 May 2009), http://dx.doi.org/10.1109/CEC.2009.4983119
22. L. Mussi, S. Cagnoni, and F. Daolio, "GPU-based road sign detection using particle swarm optimization," in *Proc. of Ninth International Conference on Intelligent Systems Design and Applications*, Pisa, pp. 152–157 (30 Nov.–2 Dec. 2009), http://dx.doi.org/10.1109/ISDA.2009.88
23. W. Zhu and J. Curry, "Particle swarm with graphics hardware acceleration and local pattern search on bound constrained problems," in *Proc. of IEEE Swarm Intelligence Symposium*, Nashville, TN, pp. 1–8 (30 March–2 April 2009), http://dx.doi.org/10.1109/SIS.2009.4937837
24. R. A. Johnson and D. W. Wiechern, *Applied Multivariate Statistical Analysis*, 6th ed., Prentice-Hall, Upper Saddle River, NJ (2007).
25. W. -M. Pang, T. -T. Wong, and P. -A. Heng, "Generating massive high-quality random numbers using GPU," in *Proc. of IEEE Congress on Evolutionary Computation*, Hong Kong, pp. 841–847 (1–6 June 2008), http://dx.doi.org/10.1109/CEC.2008.4630894
26. J. Harsanyi and C. -I Chang, "Hyperspectral image classification and dimensionality reduction: an orthogonal subspace projection approach," *IEEE Trans. Geosci. Remote Sens.* **32**(4), 779–785 (1994), http://dx.doi.org/10.1109/36.298007.
27. Y. -L. Chang et al., "A parallel simulated annealing approach to band selection for high-dimensional remote sensing images," *IEEE J. Select. Topic. Appl. Earth Observat. Rem. Sens.* **4**(3), 579–590 (2011), http://dx.doi.org/10.1109/JSTARS.2011.2160048

Biographies and photographs of the authors are not available