Published in final edited form as:

Comput Geosci. 2019 June ; 127: 111–122. doi:10.1016/j.cageo.2019.02.009.

Design and implementation of a parallel geographically weighted k -nearest neighbor classifier

PU Yingxia^{*,a,b,c}, ZHAO Xinyi^a, CHI Guangqing^d, ZHAO Shuhe^{a,b,c}, WANG Jiechen^{a,b,c}, JIN Zhibin^e, YIN Junjun^f

^aSchool of Geography and Ocean Science, Nanjing University, Nanjing 210023, China

^bJiangsu Provincial Key Laboratory of Geographic Information Science and Technology, Nanjing 210023, China

^cJiangsu Center for Collaborative Innovation in Geographical Information Resource Development and Application, Nanjing 210023, China

^dDepartment of Agricultural Economics, Sociology, and Education, Population Research Institute, and Social Science Research Institute, Pennsylvania State University, University Park, PA 16802, USA

^eNanjing Haixing Power Technology Co. Ltd., Nanjing 211100, China

^fSocial Science Research Institute and Population Research Institute, Pennsylvania State University, University Park, PA 16802, USA

Abstract

The development of high-performance classifiers represents an important step in improving the timeliness of remote sensing classification in the era of high spatial resolution. Geographically weighted k -nearest neighbors (gw k -NN)—a classifier that incorporates spatial information into the traditional k -NN classifier—has demonstrated to be better at mitigating salt-and-pepper noise and misclassification. However, the integration of spatial dependence relationships into spectral information is computationally intensive. To improve computing performance, this paper discusses two commonly used parallel strategies—data and task parallelism—used to parallelize the gw k -NN classifier in the model training and classification stages, and implements the parallel algorithm by calling MPI and GDAL in the C++ development environment on a standalone eight-core computer. We further investigate the potential performance of dual parallelism (the simultaneous exploitation of data and task parallelism) in image classification. The experimental results demonstrate that the parallel gw k -NN classifier can improve the efficiency of high-resolution, remotely sensed images with multiple land cover types. Specifically, data parallelism is more effective than task parallelism in both model training and classification stages because of the minor role of parallel overhead in total execution time. In addition, dual parallelism can take advantage of data and task parallel strategies in the image classification stage, as evidenced by the two largest speedups attained under dual parallelism I (5.28 $\times$) and II (5.73 $\times$). Comparatively, dual parallelism II, in which priority is given to data decomposition, achieves the best performance by

*Corresponding author: Tel.: +86-150-51848622; Fax: +86-25-89682686; yingxiapu@nju.edu.cn.

overlapping computation and data transmission, which is compatible with the current trend toward multicore architectures.

Keywords

remotely sensed image classification; parallel computing; geostatistical models; parallel gwk -NN classifier

1. Introduction

Advancement in sensor technology has led to remote sensing applications with extremely high computational requirements for implementing, for example, time-critical, large-scale environmental monitoring and disaster assessment (Chen et al., 2015; Ma et al., 2015; Plaza and Chang, 2008; Qin et al., 2014; van Zyl, 2012). Significant efforts have recently been directed toward the incorporation of high-performance computing (HPC) technologies such as parallel and distributed computing into image data acquisition, classification, and visualization (Choi et al., 2013; Lee et al., 2011; Li, et al., 2017; Plaza, 2009; Plaza et al., 2011a, 2011b). Most widely used classification methods (e.g., maximum likelihood classifier [MLC], k -nearest neighbor [k -NN], and support vector machine [SVM]), have been parallelized in various HPC platforms, including a massively parallel multiprocessor, a specialized field programmable gate array (FPGA), and graphic processing units (GPUs) (Caccetta et al., 1996; Choi et al., 2013; Gualtieri, 2005; Jó wik et al., 1998; Kubota et al., 2000; Li et al., 2017; Plaza and Chang, 2008). Despite increases in the spatial resolution of sensors, however, the accuracy of spectra-based land cover classification can be compromised by the so-called salt-and-pepper effect, as higher-resolution images are likely to have higher within-class variability (Atkinson and Naser, 2010; Chen et al., 2016; Hay et al., 1996; Kettig and Landgrebe, 1976; Li et al., 2014; Myint et al., 2011).

A number of spatio-contextual analysis techniques including texture extraction, Markov random fields (MRFs) modeling, image segmentation, and object-based image analysis have been developed to improve classification accuracy and reduce the salt-and-pepper effect in many application areas (Andekah et al., 2017; Atkinson, 2004; Blaschke, 2010; Chen et al., 2016; De Jong and van der Meer, 2004; Johnson et al., 2012; Moser et al., 2013; Qiao et al., 2011; Stuckens et al., 2000; Tso and Mather, 1999; Wang et al., 2016). Although such applications originated as computer vision, pattern recognition, and image analysis techniques, these attempts have been directed to address the problem of spatial dependence (Li et al., 2014). As a kind of technique for estimating the local values of properties that vary across geographic space from sample data, geostatistics has been widely used in characterizing inherent spatial variation among pixels (Atkinson and Lewis, 2000; Curran, 1988; Curran and Atkinson, 1998; Comber et al., 2016; Dungan, 1998; Herzfeld and Higginson, 1996; Ramstein and Raffy, 1989; Woodcock et al., 1998a, 1998b). Particularly, best fit geostatistical models can outperform the direct use of sample variograms in measuring texture in remotely sensed classification. Atkinson and Naser (2010) accordingly developed a geographically weighted k -NN (gwk -NN) classifier that combined geostatistical models with a supervised k -NN classifier; this had a significant increase in accuracy and

eliminated the salt-and-pepper effect. Despite these advantages, techniques that use spatio-contextual information also add computational complexity, which creates new challenges to using HPC technologies in remote sensing. With the goal of accelerating the computational performance of remote sensing classification, in this study we examine the application of HPC technologies in designing and implementing a parallel version of the *gwk*-NN classifier.

A parallel algorithm is of crucial importance for a parallel *gwk*-NN classifier to fully exploit advanced computing environments and tools. In the domain of parallel computing, data and task parallelism—which apply data partition and task grouping, respectively—are the most commonly used parallel strategies for increasing computational speed (Nicolescu and Jonker, 2002; Ramaswamy et al., 1997; Subhlok and Vondran, 2000). At its core, data parallelism involves the simultaneous performance of duplicate operations on different partitioned datasets. Because all datasets are processed in a parallel and independent manner, data parallelism is an example of efficient peer-to-peer parallelism. Under task parallelism, which is also known as *control* or *process parallelism*, various types of processes are defined for carrying out different tasks on a given dataset in a process of “dividing and conquering.” In this way, the subresults produced by the respective computing processes must be transmitted to the master process for integrating into a final result in a master-slave mode (Chen et al., 2015; Nicolescu and Jonker, 2002). One of the most widely used ways to go parallel, Message Passing Interface (MPI) has been around since the early 1990s and encouraged the development of the parallel software industry as well as large-scale parallel applications. Not only suitable for the cluster environment, the computing capability of MPI is also useful and even staggering in the single node with multiple many-core GPUs (Kingham, 2015; Lusk and Gropp, 1995). In this study, we implemented the proposed parallel *gwk*-NN classifier in the C++ programming language using calls to MPI and the geospatial data abstraction library (GDAL). Data and task parallelism have their own characteristics in our parallel *gwk*-NN classifier; the appropriate design and implementation may effectively decrease the computation time during the entire classification.

In addition to the two conventional parallel strategies, we further employed a dual parallelism method to improve the parallel performance by combining data and task parallelism and synthesizing their respective characteristics during the image classification stage. Ramaswamy and Sapatnekar (1997) first proposed a framework for exploiting data and task parallelism simultaneously on distributed memory machines and obtained speedup improvements. However, the combination of data and task parallelism is complex, and choosing an optimal strategy to make better use of computing resources still needs to explore parallelizing the *gwk*-NN classifier. The purpose of this research is to consider the general parallelization of a *gwk*-NN classifier based on different parallel strategies in the model training and classification stages, and further evaluate their parallel performance.

Section 2 outlines in detail the design of the parallel *gwk*-NN classifier used in model training and image classification. Section 3 provides an example of land cover classification through application of the parallel *gwk*-NN classifier to a SPOT-5 image of Nanjing, China to highlight the excellent performance of the proposed technique. In Section 4, we discuss the effects of different parallel strategies and various influencing factors on the parallel

performance of the parallel gw k -NN classifier. Finally, Section 5 presents conclusions that can serve as guidelines to future improvement.

2 Methods

2.1 The gw k -NN classifier

Atkinson and Naser (2010) presented a gw k -NN classifier based on the first law of geography. In this classifier, spatial information is incorporated into model training and image classification by replacing the conditional probability in conventional k -NN classifiers with fitted geostatistical models that describe the spatial distribution features of ground objects. The specific algorithm is summarized in the following subsections.

2.1.1 Training geostatistical models

1) Sample acquisition.: Given K land cover classes of the study area are determined in advance, the K dataset of training samples is acquired, with each dataset containing multiple image pixels with spatial location and spectral information as well as their land cover class.

2) Calculation of class-conditional probability.: The probability $p_{k,k}(\mathbf{h})$ of any pixel i attaching to land cover class k , given any pixel j at a given distance lag $\mathbf{h}$ falling into the same class k , is calculated by

$$p_{k,k}(\mathbf{h}) = \frac{\sum_{i=1}^N I(z_{\mathbf{h}} = k | z_i = k)}{\sum_{i=1}^N I(z_i = k)}, \quad (1)$$

where $z_{\mathbf{h}}$ represents the class of pixel j , which is at lag $\mathbf{h}$ from pixel i . If pixels i and j both belong to class k , the indicator function $I(z_{\mathbf{h}} = k | z_i = k)$ takes the value of 1; otherwise, it takes the value of 0. N is the total number of training samples; correspondingly, $\sum_{i=1}^N I(z_i = k)$ is the number of samples in class k . By adjusting the step length, the lag $\mathbf{h}$ can be set to cover the entire image.

3) Model fitting.: For each class k , different geostatistical models are employed to fit the class-conditional probability $p_{k,k}(\mathbf{h})$, with the optimal model in terms of the minimized residual sum of squares used in the classifier. In this study, exponential and Gaussian models were selected as candidate geostatistical models. The exponential model used to represent $p_{k,k}(\mathbf{h})$ is as follows:

$$p_{k,k}(\theta, \mathbf{h}) = 1 - c\{1 - \exp(-\mathbf{h}/r)\}, \quad (2)$$

where $\mathbf{h}$ is the same as described above, c is the partial sill ($0 < c < 1$), r is the distance parameter of the exponential model, and $\theta = \{c, r\}$ is the set of model parameters. The effective range of the exponential model, within which 95% of the variation can be found, is $3r$.

The Gaussian model used to represent $p_{k,k}(\mathbf{h})$ is given by:

$$p_{k,k}(\theta, \mathbf{h}) = 1 - c\{1 - \exp(-3(\mathbf{h}/a)^2)\}, \quad (3)$$

where $\mathbf{h}$ and c are the same as described above, a is the range of the Gaussian model, and $\theta = \{c, a\}$ is the set of model parameters.

2.1.2 Classifying image pixels

1) Data preparation.: All pixels of the remotely sensed images and relevant information, including the number of bands, starting point coordinates, and spatial resolution are extracted for classification.

2) Pixel classification.: A gw k -NN classifier considering spatial distribution and dependence is produced using the optimal models selected for each land cover class in the model training stage. This results in a total of K gw k -NN classifiers corresponding to K land cover classes. The probability that any pixel i belongs to class k is as follows:

$$p(z_i = k) = \frac{\sum_{j=1}^J p_{k,k}(\mathbf{h}) \cdot \omega_{ij} \cdot I(z_j = k)}{\sum_{k'=1}^K \sum_{j=1}^J p_{k,k'}(\mathbf{h}) \cdot \omega_{ij} \cdot I(z_j = k')}, \quad \omega_{ij} = \frac{1}{d_{ij}^p}, \quad (4)$$

where $p_{k,k}(\mathbf{h})$ is the geographical weighting function of the selected geostatistical model for class k ; ω_{ij} is a weight corresponding to the distance d_{ij} between pixels i and j ($j = 1, 2, \dots, J$), in which p is a user-defined parameter; $I(z_j = k)$ denotes the membership degree of the neighboring pixel j belonging to class k , as discussed previously (Maselli et al., 2005); and k' is an additional class index that is similar to k in all K classes. The final land cover class allocated to the object pixel i is the maximum value of $p(z_i = k)$ from all classes produced by the K gw k -NN classifiers.

3) Result writing.: The classification results of the individual pixels are written to a hard disk and their respective classes are used to draw the final classification map.

2.2 Parallel design

Because the model training of the gw k -NN classifier is independent of the subsequent image classification, both stages can be parallelized by different parallel strategies to optimize the overall classification efficiency.

2.2.1 Data and task parallelism for model training—In fitting K classes of sample data using M models, it is necessary to carry out $M \times K$ training tasks to select the optimal model for each class; in the case of sequential execution, this can take a considerable amount of time. A flowchart of training geostatistical models for different classes is shown in Fig. 1.

Data parallelism is suitable for the model training stage, as each class is independently trained by different geostatistical models. A flowchart of this process is shown in Fig. 2(a). In this stage, n activated parallel processes access sample datasets corresponding to different land cover classes by simultaneously reading metadata information from the data source. Each process then independently uses candidate geostatistical models in sequence to train the

accessed dataset labelled a unique land cover class and selects the optimal model for this class. That is, different processes take charge of different datasets but the geostatistical models they employed are identical. Moreover, if n is less than K , the process that completes the computation of the assigned class first will continue to access and train the remaining classes until all classes are trained.

Task parallelism can also be applied to train models by associating different processes with different geostatistical models. As the models do not need to communicate with each other, it is not necessary in this stage to determine an order of precedence relation among the geostatistical models. Accordingly, each of the n processes simultaneously undertakes the training task of all the classes by applying one or more models, which depend on the size relationship between the number of activated parallel processes n and the number of geostatistical models M , similar to the judgement of the size relationship under the data parallelism introduced previously. The training results produced by the respective computing processes are then sent to the master process (process 0) to determine the optimal fitting model for each class, as shown in Fig. 2(b). When there is a sufficient number of processes, different models are trained by all the datasets in parallel. This contrasts with the process under data parallelism, in which the datasets of different classes for training all the models are performed in parallel while the models are trained by each process in a sequential manner.

2.2.2 Data and task parallelism for image classification—After training models for each land cover class, the selected optimal models are further combined with k -NN classifiers to form the multiple $gw k$ -NN classifiers that are used to calculate the probabilities of all pixels belonging to each class within an image.

During the classification stage, data parallelism partitions data into multiple datasets and assigns these datasets to different processes for classification in parallel. That is, all processes in charge of the $gw k$ -NN classifiers corresponding to all classes synchronously execute classification on different datasets. The classification results produced by each process are then integrated into a final result for the overall image. The execution of image classification under data parallelism is shown schematically in Fig. 3.

Task parallelism is also suitable for image classification, because $gw k$ -NN classifiers of different classes are independent. Similar to the execution of task parallelism in the model training, the number of classifiers managed by each computing process depends on the size relationship between the number of processes n and the number of $gw k$ -NN classifiers K . If the number of processes is sufficient to execute all of the classifiers, the respective processes will classify the entire image by different classifiers simultaneously; otherwise, some processes will execute multiple classifiers sequentially (Fig. 4). After receiving the subresults from each of the computing processes, the master process (process 0) produces the final classification result. In theory, task parallel strategy can exponentially reduce classification time.

3 Experimental study

3.1 Study area and data preparation

The study area was located in the eastern part of Bagua Zhou Island in Nanjing, China (Fig. 5). This island is a sandbar silted by the Yangtze River and situated four kilometers downstream of the Nanjing Yangtze River Bridge. It is geographically separated from the Nanjing urban area to the south and the Luhe District to the north by the Yangtze River. The ground objects in this area are typical and have a high spatial continuity, which makes it easy to collect sample data and verify the land cover classes.

Remotely sensed data to be used were extracted from the fused SPOT-5 image in 2002 (Fig. 6). In this study, a Brovey transform—a well-known, color-normalized fusion technique—was used to produce a higher spatial resolution image by merging low-resolution multispectral (MS) bands with a higher spatial resolution panchromatic (PAN) band (Vrabel 1996; Gharbia et al., 2014; Pohl and Genderen 2016). Here, each SPOT-5 MS band with a 10 m resolution was multiplied by a ratio of a PAN band with a 2.5 m resolution divided by the sum of the MS bands. The output PAN-sharpening image has the higher spatial resolution of the PAN image and the spectral characteristics of the MS image, which is more useful for image classification and other applications. The size of the fused SPOT-5 image is 321 rows $\times$ 361 columns. Based on visual interpretation of the SPOT image and an additional on-the-spot field survey, the land cover in the study area was divided into five classes: bare land, farmland, grassland, road, and water. Prior to image classification, 330 training samples were selected to fit geostatistical models for different land cover classes and 205 test samples were selected to assess the classification accuracy of the parallel gw k -NN classifier (Table 1).

Following a comparison of the spectral heterogeneities among different bands in the training sample data (Fig. 7), we selected the red and green bands for the gw k -NN classifier based on a discernible spectral variation within different classes and a relatively small spectral overlap.

3.2 Implementation of parallel gw k -NN classifiers

The parallel gw k -NN classifier was implemented in the C++ programming language using calls to MPI and GDAL on a standalone Intel Xeon E5–2620V4 eight-core computer. MPI as a portable message-passing standard is widely used in parallel computing architecture and the implementation of parallel programs (Lusk and Gropp, 1995; MPICH, 2012). GDAL is a translator library for raster and vector geospatial data formats that supports the acquisition of the raster data source and information on its driver, projection, and resolution, as well as other relevant information (GDAL, 2015). In our experiment, GDAL 1.11.2 and MPICH2 1.5, a high-performance and widely portable implementation of the MPI standard, were used to implement the parallelization of gw k -NN classifiers.

3.2.1 Data and task parallel implementation in the model training—The pseudocode for fitting geostatistical models in data and task parallel schemes is shown in Table 2. K is the number of land cover classes, M represents the number of geostatistical

models, and np stands for the number of processes executed in parallel. The functionality of the pseudocode corresponds to the appropriate descriptions given in Section 2.2.1.

Based on the empirical class-conditional probability plot for each land cover class, we selected exponential and Gaussian models as candidate geostatistical models to fit the spatial distribution features of the training samples. Other kinds of semivariogram models, such as spherical and power models, were excluded because of mismatch between empirical plots and theoretical models. Table 3 shows the selected optimal model for each land cover class with the corresponding parameter estimates, and Fig. 8 further depicts the graphs of the selected geostatistical models. Notably, the determination of the optimal model for each land cover class is not affected by the parallel strategy adopted.

The exponential model is a much better fit to the spatial distribution features of bare land, grassland, and road; the Gaussian model is suitable for characterizing the other land cover classes of farmland and water. Additionally, farmland extends over geographic space with a range of 16.5 m (the spatial resolution is 2.5 m) and shows the strongest spatial dependence among the five land cover classes, with a remarkably high class-conditional probability of more than 50%. Comparatively, the class-conditional probability of water is rather low, generally less than 30%. Over the whole region, farmland covers the largest area and has the widest spatial distribution, which also enhances its spatial correlation to a certain degree.

3.2.2 Data and task parallel implementation in the image classification—The pseudocode for classifying an image in data and task parallel schemes is shown in Table 4. K , M , and np are the same as mentioned previously. The functionality of the pseudocode corresponds to the appropriate descriptions given in Section 2.2.2.

Based on different parallel schemes, the parallel gwk -NN classifiers were implemented to classify the whole image with the optimal model for each class. As a benchmark for comparison, we also ran the conventional k -NN classifier. Fig. 9 shows the classification results produced by these two classifiers. It is seen that the gwk -NN approach classifies the overall image more smoothly as a result of using geographical weights and spatial distribution features of the ground objects. In addition, the salt-and-pepper noise present in the k -NN classification results is greatly alleviated.

The overall and individual classification accuracies of the parallel gwk -NN classifier were summarized in a confusion matrix (Table 5). Bare land and water classes achieved very high classification accuracies (100%). Because the spatial dependent relationship is considered by the gwk -NN classifier, problems arising from the same object producing varying spectra and different objects producing the same spectrum were greatly mitigated, thereby increasing the overall accuracy to 98.54%. The results indicate the gwk -NN classifier can meet the requirements of land cover classification.

3.3 Performance evaluation of parallel gwk -NN classifiers

To evaluate the performance of parallel gwk -NN classifiers under a multicore computing environment, we took the running time, speedup, and efficiency into account (Czech, 2016). Running time (T), or the worst-case parallel running time, is recorded from the start of

executing an algorithm by the first process until the end of execution by all processes. Given the problem size, the less the time costs, the better the parallel performance. Speedup (S) measures the benefit achieved by parallel computation, which is defined as the ratio of the execution time of the best sequential algorithm with a single process to the execution time with a number of processes in a parallel computing environment. The speedup achieved is typically less than the number of processes because of unbalanced loads or computation overhead such as communication between different processors and synchronization of their operation. Efficiency (E), defined as the ratio of speedup (S) to the number of processors, measures processor utilization. The maximum value of efficiency is 1, meaning none of the processes are idle (Grama et al., 1993; Orii, 2010).

Theoretically, the performance of the parallel gw k -NN classifiers can be obtained by the overall speedup (the ratio of the total time of sequential execution involved in the model training and image classification to that of parallel execution time) because the model training and image classification are independent. In practice, however, the overall speedup might not reflect the great improvement in the respective stages of the parallel gw k -NN classifier because the execution times of the model training and classification are very different and there may be no comparability between them. In addition, different parallel strategies in the model training and classification stages will lead to different parallel performance. Therefore, we present the parallel performance in the model training and classification stages, respectively. Moreover, to ensure that multiple processes can simultaneously read different datasets allocated to them, each process first calculates the starting position in the corresponding dataset and then operates the reading task at the same time.

3.3.1 Model training—The specific parallel performance results under data and task parallelism in the model training stage are listed in Table 6.

Under data parallelism, the training time continuously decreases as the number of processes is increased to four, which is close but not equal to the total number of land cover classes (five). Ideally, optimal performance would occur when each of the five processes trains a respective sample of the corresponding class, i.e., when all datasets are trained in parallel and the total training time is determined by the maximum elapsed time among the five training processes. In this case, however, the optimized allocation and scheduling achieve the shortest training time in advance, with four processes. As the number of processes increases, the idle processes cost CPU resources and affect the overall performance. Based on this result, we hypothesize that the optimal performance under data parallelism occurs when the number of processes, up to the sample sizes of different classes, is equal to or slightly less than the number of classes.

By contrast, the parallel performance under task parallelism does not conform to our prior expectations. If there are only two processes, one will engage in model training for the overall sample dataset while the other will serve as the master process. In this case, extra time might be required to transfer messages, leading to a decrease in parallel efficiency. However, increasing the number of processes to three and then four results in a slight decrease in training time initially, followed by a significant increase (from four on), which in

some cases exceeds the time of sequential execution. It is possible that it is not necessary to involve additional processes to support task parallelism because only two geostatistical models are employed in fitting training samples in this stage. Thus, data parallelism is seen to be more suitable than task parallelism for model training in this study.

3.3.2 Image classification—Data and task parallelism were then applied to implement the parallelization of the *gw_k*-NN classifiers during the image classification stage. Table 7 shows classification times, speedups, and parallel efficiencies achieved using a limited number of processors. Both data and task parallelism reduce classification time significantly as the number of processes increases, indicating that both strategies play effective roles in this stage. Note, however, that the parallel performance under data parallelism is much better than that under task parallelism.

Under data parallelism, a remotely sensed image can be divided into multiple strip-sized datasets, each of which is then assigned to different processes to be classified in parallel. The size of each dataset decreases as the number of processes grows, thereby reducing the overall classification time. Although there is a continuous rise in the speedup, the parallel efficiency of the classifiers keeps decreasing. In contrast, the classification time under task parallelism tends to first decline and then rise as the number of processes increases, with parallel performance reaching a maximum at five processes. It is worth noting that the computing processes used in this case are sufficient to attain an optimal schedule for five *gw_k*-NN classifiers, as is the case for data parallel implementation in the model training. Using five classifiers, the classification time is the shortest and the speedup is the highest; adding processes beyond five leads to an increase in the time needed to pass messages between different processes, in turn decreasing the overall efficiency. For parallel efficiency, classification time fluctuates by first rising and then declining. The specific values for parallel efficiency are significantly smaller than those under data parallelism overall.

To take full advantage of both parallel strategies, we implemented dual parallelism in the classification stage. Dual parallelism is more complicated than either pure data or task parallelism as it can entail various process allocations and implementation methods. Based on the preceding results, in which optimal parallel performance under task parallelism was obtained using four computing processes excluding the master process, the dual parallelism processes were divided into groups of four. In this scheme, if there are no more than four processes, the parallel mode will involve pure task parallelism, as there will be an insufficient number of processes to implement data parallelism. As the number of processes increases, they are grouped in advance to classify different divided datasets of the overall image in parallel. Notably, the process receiving the subresults in each group also executes the *gw_k*-NN classifiers to avoid the idleness of processes. For instance, six operating processes were separated into two groups comprising four and two processes, respectively. These two groups respectively classify the two halves of an image as independent datasets. From a general perspective, different groups classifying separate datasets is an implementation of data parallelism, while from a local perspective the classification of one dataset by a group of processes is an implementation of task parallelism. Table 8 shows the performance results produced by the parallel *gw_k*-NN classifiers under dual parallelism.

From the table, we can see that dual parallel performance achieved further improvements with respect to pure task parallelism, in which the optimal number of computing processes is four, excluding the master process. In general, the total classification time under dual parallelism declined significantly as the number of processes increased. At the maximum number (eight) of processes applied in this experiment, the classification time was reduced to its lowest value (5.38 s). In addition, the parallel efficiency began to rise after five processes rather than keep declining, reaching 0.66 in the case of eight processes. Overall, dual parallelism can make better use of processes in terms of sharing datasets and multiple *gw**k*-NN classifiers and is seen to be feasible and effective in image classification.

4. Discussion

The number of processes, the computing environment, and the objects to be classified all markedly impact parallel performance. For a given set of objects and computing environment, the most suitable number of processes and optimal parallel performance can be experimentally determined using an enumeration method. The speedup under different parallel strategies in the image classification stage is illustrated as a function of the number of parallel processes in Fig. 10. Among such strategies, dual parallelism is quite complex and will make different changes to the parallel performance under different process schedules. Dual parallelism I is based on the premise of task parallelism discussed in the previous section, while dual parallelism II, which takes data decomposition as a prerequisite and then further implements task parallelism on different datasets, is also considered as an alternative strategy to parallelize the *gw**k*-NN classifiers. Furthermore, we calculated the Karp-Flatt metric, an experimentally determined sequential fraction of the total operation that cannot be parallelized regardless of method (e.g., reading data, setting up calculations, control logic, and storing results), to analyze the impacts arising from variables on the parallel performance. The smaller a serial fraction is, the better an algorithm could be recognized (Karp and Flatt, 1990).

Under data parallelism, the increase in speedup flattens as the number of processes grows. With a smaller number of processes, the data partitioning significantly reduces the size of the dataset allocated to each process, resulting in a rapid drop in time consumption. As the number of processes increases, however, the benefit through parallel computation is further limited by insufficient degree of concurrency, computation overhead, or synchronization of operations. Finally, the parallel performance (speedup) stabilizes at around five processes. To minimize the cost of parallel computation, we must optimize the parallel algorithm by reducing redundant computations, avoiding idle processes, and/or inhibit overlapping communication and computation. Alternatively, we can increase the size of the image to keep speedup and efficiency growing with a fixed number of processes, which is indicated by the Amdahl effect.

Under task parallelism, there is an initial rise in speedup followed by a decline. Peak performance occurs at five parallel processes that relate to the number of land cover classes. The overall performance improvement under task parallelism is not as significant as that under data parallelism, with a peak speedup of only half that achieved in the latter case. As the number of processes surpasses five, the gap between data and task parallelism tends to

expand. The reason for the unsatisfactory parallel performance may lie in relatively higher serial fractions measured by the Karp-Flatt metric and frequent communication between slave and master process under task parallelism.

Dual parallelism increases parallel performance to a maximally practical extent by comprising data and task parallelism. Under dual parallelism I, priority is given to assign processes to different classifiers owing to the fixed number of land cover classes. As such, the key to dual parallelism I is adding processes to execute data parallelism based on the premise that task parallelism is fulfilled, which indicates that dual parallelism is indeed in effect following the task parallelism of four processes. The speedup pattern shown in Fig. 10 suggests that this type of parallel strategy further improves the performance of the task parallelism, except in the case of five processes, in which half of the images are classified in a sequential manner by one process. In addition, the improvement with the increasing number of processes is more significant than that under pure data parallelism, according to the increasing trend of speedup after five processes. This suggests that dual parallelism is an effective strategy for improving the classification efficiency through the mix of data and task parallelism, characterized by much smaller serial fractions than pure task parallelism and even pure data parallelism.

We also considered the priority of data partition in dual parallelism II. Specifically, if the number of processes is four or below, this method first assigns different processes to different partitioned datasets, giving priority to data parallelism. Subsequent processes are then employed to achieve the task parallelism of different datasets. Fig. 10 shows that, in general, the speedup achieved under dual parallelism II significantly improves as the number of processes increases, reaching a maximum at eight processes. Additionally, this method is particularly effective in situations in which the task parallelism for each dataset is synchronously achieved by two processes (np equal to eight). One reason for the good parallel performance under dual parallelism II is the lower serial fraction ratio than that of dual parallelism I.

Finally, the hardware of the computing environment is of equally critical importance to parallel performance. Generally speaking, multicore processors support more processes executed in parallel. If a processor is not able to accommodate the parallel execution of very large number of processes, it will result in the nonstop switching of current processes to satisfy the parallel requirement, which in turn affects overall parallel efficiency. In addition, increasing the number of land cover classes requires the addition of gwk -NN classifiers, which demands more processes to be executed in parallel. The optimal number of processes for achieving peak performance will also be affected by the characteristics of the image. Accordingly, it is important to consider both the computing environment and the classification object in optimizing the performance of the parallel gwk -NN classifier.

5. Conclusion

With the advancement of sensor and computer technologies in data collection and manipulation, the rapid growth in high-resolution remotely sensed imaging has led to computational demands in many time-critical remote sensing applications. To take full

advantage of HPC technologies, we designed the parallelization of the *gw**k*-NN classifier in terms of model training and image classification stages and further implemented it by using calls to MPI and GDAL in the C++ development environment on an eight-core CPU. Moreover, the parallel performance of the parallel strategies (data and task parallelism as well as their combination—dual parallelism) in our improved classifiers are evaluated on the basis of the implementation results.

Our preliminary experiment demonstrates that the proposed parallel *gw**k*-NN classifier can improve the efficiency of high-resolution remotely sensed images with multiple land cover types. In general, data parallelism is found to be effective in both the model training and classification stages, while dual parallelism can exploit the potentialities of data and task parallelism in the image classification stage. Specifically, the two largest speedups were achieved under dual parallelism I and II with eight processes, which are superior to those under pure data or task parallel strategies.

The parallel performance of multiple *gw**k*-NN classifiers is limited by inherently sequential computation and communication overhead. Given the fixed size of an image, parallel overhead under data parallelism tends to grow with the increasing number of processes, but does not constitute a major part in the total time of parallel computation. However, frequent communication between slave and master process under task parallelism leads to unsatisfactory parallel performance. Dual parallelism can reduce the execution time and improve parallel efficiency by overlapping computation and data transmission. In particular, dual parallelism II achieves the best performance with the priority to data decomposition.

Although the proposed parallel *gw**k*-NN classifier can be significantly improved in performance under different parallel strategies, we only tested the parallel algorithm on a standalone eight-core computer. Further studies may involve: (1) the use of a parallel cluster environment to enhance the performance of the parallel *gw**k*-NN classifier, (2) the introduction of additional geostatistical models for multiple land cover classes to achieve more accurate classification, (3) the use of remotely sensed data from larger areas to test the scalability of the parallel *gw**k*-NN classifier, and (4) the consideration of dynamic load balancing to generate more balanced distribution of loads across processes.

Acknowledgments

This research was supported in part by the National Natural Science Foundation of China (Grant #41271388), Priority Academic Program Development of Jiangsu Higher Education Institutions, Jiangsu Center for Collaborative Innovation in Geographical Information Resource Development and Application, the U.S. National Science Foundation (Awards #1541136 and #1823633), the U.S. Department of Transportation (Awards #DTRT12GUTC14-201307 and #DTRT12GUTC14-201308), and the Eunice Kennedy Shriver National Institute of Child Health and Human Development (Award #P2C HD041025).

References

- Andekah ZA, Naderan M, Akbarizadeh G, 2017 Semi-supervised hyperspectral image classification using spatial-spectral features and superpixel-based sparse codes. 2017 Iranian Conference on Electrical Engineering (ICEE), Tehran, pp. 2229–2234.
- Atkinson PM, 2004 Spatially weighted supervised classification for remote sensing. *Int. J. Appl. Earth Obs. Geoinf* 5(4), 277–291.

- Atkinson PM, Lewis P, 2000 Geostatistical classification for remote sensing: an introduction. *Comput. Geosci* 26(4), 361–371.
- Atkinson PM, Naser DK, 2010 A geostatistically weighted k-NN classifier for remotely sensed imagery. *Geogr. Anal* 42(2), 204–225.
- Blaschke T. 2010 Object based image analysis for remote sensing. *ISPRS J. Photogramm. Remote Sens* 65(1), 2–16.
- Caccetta PA, Campbell NA, West GA, 1996 A massively parallel implementation of an image classifier. In: *Proceedings of the 16th Asian Conference on Remote Sensing*, pp. 203–307.
- Chen L, Ma Y, Liu P, Wei J, Jie W, He J, 2015 A review of parallel computing for large-scale remote sensing image mosaicking. *Cluster Comput.* 18, 517–529.
- Chen Y, Ge Y, Jia Y, 2016 Integrating object boundary in super-resolution land-cover mapping. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens* 10(1), 219–230.
- Choi SW, Lee CH, 2013 A FPGA-based parallel semi-naïve Bayes classifier implementation. *IEICE Electron. Expr* 10(19), 1–7.
- Comber AJ, Harris P, Tsutsumida N, 2016 Improving land cover classification using input variables derived from a geographically weighted principal components analysis. *ISPRS J. Photogramm. Remote Sens* 119, 347–360.
- Curran PJ, 1988 The semivariogram in remote sensing: an introduction. *Remote Sens. Environ* 24(3), 493–507.
- Curran PJ, Atkinson PM, 1998 Geostatistics and remote sensing. *Prog. Phys. Geogr* 22(1), 61–78.
- Czech ZJ, 2016 *Introduction to Parallel Computing*. Cambridge University Press, Cambridge, UK p. 66.
- De Jong SM, van der Meer FD 2004 *Remote Sensing Image Analysis: Including the Spatial Domain*. Springer, Dordrecht, The Netherlands p. 359.
- Dungan J, 1998 Spatial prediction of vegetation quantities using ground and image data. *Int. J. Remote Sens* 19(2), 267–285.
- GDAL, 2015 Geospatial Data Abstraction Library, version 1.11.2. Open Source Geospatial Foundation. <http://www.gdal.org>, accessed date: June 2015.
- Gharbia R, Baz AHE, Hassanien AE, Tolba MF, 2014 Remote sensing image fusion approach based on Brovey and Wavelets transforms. In: Kömer P, Abraham A, Snášel V. (eds.), *Proceedings of the Fifth Intern. Conf. on Innov. in Bio-Inspired Comput. and Appl IBICA 2014, Advances in Intelligent Systems and Computing* 303, Springer International Publishing, Switzerland pp. 311–321.
- Grama AY, Gupta A, and Kumar V. 1993 Isoefficiency: Measuring the scalability of parallel algorithms and architectures. *IEEE Parallel Distrib. Technol* 1(3): 12–21.
- Gualtieri JA, 2005 A parallel processing algorithm for remote sensing classification In: *Proc. Airborne Earth Science Workshop*, Pasadena, CA, 1–14.
- Hay GJ, Niemann KO, Mclean G, 1996 An object-specific image-texture analysis of H-resolution forest imagery. *Remote Sens. Environ* 55, 108–122.
- Herzfeld UC, and Higinson CA, 1996 Automated geostatistical seafloor classification-principles, parameters, feature vectors, and discrimination criteria. *Comput. Geosci* 22(1), 35–52.
- Johnson B, Tateishi R, Xie Z, 2012 Using geographically weighted variables for image classification. *Remote Sens. Lett.* 3(6), 491–499.
- Jó wik A, Serpico S, Roli F, 1998 A parallel network of modified 1-NN and k-NN classifiers—application to remote-sensing image classification. *Pattern Recognit. Lett* 19(1), 57–62.
- Karp AH, Flatt HP, 1990 Measuring parallel processor performance. *Commun. ACM* 33(5), 539–543.
- Kettig RL, Landgrebe DA, 1976 Classification of multispectral image data by extraction and classification of homogeneous objects. *IEEE Trans. Geosci. Electron* 14(1), 19–26.
- Kinghorn D, 2015 5 ways of parallel programming. <http://www.pugetsystems.com/labs/hpc/5-Ways-of-Parallel-Programming-652>.
- Kubota K, Nakase A, Sakai H, Oyanagi S, 2000 Parallelization of decision tree algorithm and its performance evaluation. In: *Proceeding of the Fourth International Conference/exhibition on High Performance Computing in the Asia-Pacific Region*, Beijing, China, 2, 574–579.

- Lee CA, Gasster SD, Plaza A, Chang C, and Huang B, 2011 Recent developments in high performance computing for remote sensing: a review. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens* 4(3), 508–527.
- Li M, Zang S, Zhang B, Li S, Wu C, 2014 A review of remote sensing image classification techniques: the role of spatio-contextual information. *Eur. J. Remote Sens* 47, 389–411.
- Li W, Fu H, You Y, Yu L, Fang J, 2017 Parallel multiclass support vector machine for remote sensing data classification on multicore and many-core architectures. *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens* 10(10), 4387–4398.
- Lusk E, Gropp W, 1995 The MPI message-passing interface standard: Overview and status. *Adv. Parallel Comput* 10(6), 265–269.
- Ma Y, Wu H, Wang L, Huang B, Ranjan R, Zomaya A, Jie W, 2015 Remote sensing big data computing: challenges and opportunities. *Future Gener. Comp. Sy* 51, 47–60.
- Maselli F, Chirici G, Bottai L, Corona P, Marchetti M, 2005 Estimation of Mediterranean forest attributes by the application of k-NN procedures to multitemporal Landsat ETM+ images. *Int. J. Remote Sens* 26(17), 3781–3796.
- MPICH, 2012 Message-passing interface, version 1.5. Message passing interface forum. <http://www.mpich.org>, accessed date: 1 June 2013.
- Moser G, Serpico SB, Benediktsson JA, 2013 Land-cover mapping by Markov modeling of spatio-contextual information in very-high-resolution remote sensing images. In: *Proceedings of the IEEE*, 101(3), 631–651.
- Myint SW, Gober P, Brazel A, Grossman-Clarke S, Weng Q, 2011 Per-pixel vs. object-based classification of urban land cover extraction using high spatial resolution imagery. *Remote Sens. Environ* 115(5), 1145–1161.
- Niculescu C, Jonker PP, 2002 A data and task parallel image processing environment. *Parallel Comput.* 28(7): 945–965.
- Orii S. 2010 Metrics for evaluation of parallel efficiency toward highly parallel processing. *Parallel Comput.* 36(1): 16–25.
- Plaza A. 2009 Special issue on architectures and techniques for real-time processing of remotely sensed images, *J. Real-time Image Pr* 4(3): 191–193
- Plaza A, Chang C, 2008 High performance computing in remote sensing. Chapman & Hall/CRC, Boca Raton, London, New York, 451pp.
- Plaza A, Du Q, Chang Y, King RL, 2011a High performance computing for hyperspectral remote sensing. *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens* 4(3), 528–544.
- Plaza A, Plaza J, Paz A, Sanchez S, 2011b Parallel hyperspectral image and signal processing. *IEEE Signal Process. Mag* 28, 119–126.
- Pohl C, Genderen JV, 2016 Remote Sensing Image Fusion: A Practical Guide. CRC Press, Taylor & Francis Group. p. 115.
- Qiao C, Shen Z, Wu N, Hu X, Luo J, 2011 Remote sensing image classification method supported by spatial adjacency. *J. Remote Sens* 12, 88–99.
- Qiao C, Wang J, Shang J, & Daneshfar B, 2015 Spatial relationship-assisted classification from high-resolution remote sensing imagery. *Int. J. Digit. Earth* 8(9), 710–726.
- Qin C, Zhan L, Zhu A, Zhou C, 2014 A strategy for raster-based geocomputation under different parallel computing platforms. *Int. J. Geogr. Inf. Sci* 28(11), 2127–2144.
- Ramstein G, Raffy M, 1989 Analysis of the structure of radiometric remotely sensed images. *Int. J. Remote Sens* 10(6), 1049–1073.
- Ramaswamy S, Sapatnekar S, Banerjee P, 1997 A framework for exploiting task and data parallelism on distributed memory multicomputers. *IEEE Trans. Parallel Distrib. Syst* 8(11), 1098–1116.
- Stuckens J, Coppin PR, Bauer ME, 2000 Integrating contextual information with per-pixel classification for improved land cover classification. *Remote Sens. Environ* 71(3), 282–296.
- Subhlok J, Vondran G, 2000 Optimal use of mixed task and data parallelism for pipelined computations. *J. Parallel Distr. Com* 60(3), 297–319.
- Tso BCK, Mather PM, 1999 Classification of multisource remote sensing imagery using a genetic algorithm and Markov random fields. *IEEE Trans. Geosci. Remote Sens* 37(3), 1255–1260.

- van Zyl J, 2012 Application of satellite remote sensing data to the monitoring of global resources. In: 2012 IEEE Technology Time Machine Symposium (TTM), Dresden, Germany, 1–1.
- Vrabel J, 1996 Multispectral imagery band sharpening study. *Photogramm. Remote Sens* 62(9), 1075–1083.
- Wang L, Shi C, Diao C, Ji W, Yin D, 2016 A survey of methods incorporating spatial information in image classification and spectral unmixing. *Int. J. Remote Sens* 37(6), 3870–3910.
- Woodcock CE, Strahler AH, and Jupp DLB, 1998a The use of variograms in remote sensing I: scene models and simulated images. *Remote Sens. Environ* 25(3), 323–348.
- Woodcock CE, Strahler AH, and Jupp DLB, 1998b The use of variograms in remote sensing II: real digital images. *Remote Sens. Environ* 25(3), 349–379.

Fig. 1.
Schematic of model training stage

Fig. 2. Flowchart of different parallel strategies in the model training stage: (a) data parallelism and (b) task parallelism

Fig. 3.
Flowchart of data parallelism in the image classification stage

Fig. 4.
Flowchart of task parallelism in the image classification stage

Fig. 5.
The location map of the study area

Fig. 6.
The SPOT-5 image of the study area

Fig. 7.
Scatter plots of spectral heterogeneity of training samples (red, green, and blue bands)

Fig. 8.
The graph of the optimal models for different land cover classes

Fig. 9.

Classification results: (a) k -NN classifier and (b) gw - k -NN classifier

Fig. 10. Speedups and Karp-Flatt ratios under different parallel strategies in the image classification stage

Table 1.

Training samples and test data

Class code	Class name	Number of training samples	Number of test data
C1	Bare land	60	40
C2	Farmland	110	60
C3	Grassland	100	60
C4	Road	20	15
C5	Water	40	30

Table 2.

Data and task parallel implementation in the model training stage

Data parallelism	Task parallelism
<pre> if $np = K$ for training samples of class i ($i = 0, 1, \dots, K - 1$) do //The classes are fitted in parallel $O_i \leftarrow$ Optimal model for class i from M models end for else $i = 0, 1, \dots, np - 1$ while $i = K - 1$ do for training samples of class i do $O_i \leftarrow$ Optimal model for class i from M models end for $i \leftarrow i + np$ //The class to be fitted in the next loop end while end if return optimal model set O </pre>	<pre> if $np = M + 1$ for geostatistical model i ($i = 0, 1, \dots, M - 1$) do //The models are called in parallel $R_i \leftarrow$ Model i fitting to K classes by process $i + 1$ $SendMessage(R_i, 0)$ //Send results to process 0 end for else $i = 0, 1, \dots, np - 2$ $cp \leftarrow i + 1$ //cp is the current process with model i while $i = M - 1$ do for geostatistical model i do $R_i \leftarrow$ Model i fitting to K classes by process cp $SendMessage(R_i, 0)$ end for $i \leftarrow i + np - 1$ //The models to be used in the next loop end while end if $R \leftarrow ReceiveMessage(R_i, any)$ //Process 0 receives results $O \leftarrow$ Gather optimal models for all classes from results R return optimal model set O </pre>

Table 3.

The fitted parameters of the optimal models for different land cover classes

Class code	Class name	Model	Sill (c)	Range
C1	Bare land	Exponential	0.6161	1.8000
C2	Farmland	Gaussian	0.4871	6.6000
C3	Grassland	Exponential	0.5891	3.6000
C4	Road	Exponential	0.6711	0.6000
C5	Water	Gaussian	0.7241	1.6000

Table 4.

Data and task parallel implementation in the image classification stage

Data parallelism	Task parallelism
<pre> //Divide the image into N datasets for dataset i ($i = 0, 1, \dots, np - 1$) do //The datasets are classified in parallel if $i \neq np - 1$ //Calculate the number of pixels in divided direction $p_ysize \leftarrow RasterYsize / N$ $S_i \leftarrow$ Classify dataset i by K gwk-NN classifiers else $ysize \leftarrow RasterYsize / N * (N - 1)$ $p_ysize \leftarrow RasterYsize - ysize$ $S_i \leftarrow$ Classify dataset i by K gwk-NN classifiers end if end for return final classification result S </pre>	<pre> if $np = K + 1$ for gwk-NN classifier i ($i = 0, 1, \dots, K - 1$) do //The classifiers are employed in parallel $S_i \leftarrow$ Classify the image by classifier i (process $i + 1$) $SendMessage(S_i, 0)$ //Send the sub result to process 0 end for else $i = 0, 1, \dots, np - 2$ $cp \leftarrow i + 1$ //cp is the current process for classifier i while $i = K - 1$ do for gwk-NN classifier i do $S_i \leftarrow$ Classify the image by classifier i (process cp) $SendMessage(S_i, 0)$ //Send the result to process 0 end for $i \leftarrow i + np - 1$ //The classifiers in the next loop end while end if $S \leftarrow ReceiveMessage(S_i, any)$ //Process 0 receive the results return final classification result S </pre>

Table 5.Classification accuracies of parallel gw_k -NN classifiers

Land cover classes	Classified results					Total
	Bare land	Farmland	Grassland	Road	Water	
Bare land	40	0	0	0	0	40
Farmland	0	59	1	0	0	60
Grassland	1	0	59	0	0	60
Road	0	0	1	14	0	15
Water	0	0	0	0	30	30
Total	41	59	61	14	30	205
User's accuracy (%)	97.56	100	96.72	100	100	
Producer's accuracy (%)	100	98.33	98.33	93.33	100	
Overall accuracy (%)		98.54		Kappa coefficient	0.9808	

Table 6.

Parallel performance under data and task parallelism in the model training stage

No. of processes	Data parallelism			Task parallelism		
	Training time (s)	Speedup	Efficiency	Training time (s)	Speedup	Efficiency
1	97.40	1.00	1.00	93.00	1.00	1.00
2	77.10	1.26	0.63	94.23	0.99	0.50
3	56.14	1.73	0.58	86.36	1.08	0.36
4	41.54	2.34	0.59	87.46	1.06	0.27
5	45.65	2.13	0.43	94.81	0.98	0.20
6	49.37	1.97	0.33	102.78	0.90	0.15
7	51.23	1.90	0.27	109.90	0.85	0.12
8	51.44	1.89	0.24	112.15	0.83	0.10

Table 7.

Parallel performance under data and task parallelism in the image classification stage

No. of processes	Data parallelism			Task parallelism		
	Classification time (s)	Speedup	Efficiency	Classification time (s)	Speedup	Efficiency
1	28.73	1.00	1.00	28.68	1.00	1.00
2	14.36	2.00	1.00	29.44	0.97	0.49
3	10.18	2.82	0.94	17.78	1.61	0.54
4	8.26	3.47	0.87	13.07	2.19	0.55
5	7.35	3.90	0.78	11.38	2.52	0.50
6	6.52	4.41	0.74	11.54	2.48	0.41
7	5.85	4.91	0.70	13.18	2.17	0.31
8	5.54	5.19	0.65	14.05	2.04	0.26

Table 8.

Parallel performance under dual parallelism in the image classification stage

Number of processes	Classification time (s)	Speedup	Efficiency
1	28.38	1	1.00
2	18.42	1.54	0.77
3	14.13	2.01	0.67
4	11.08	2.56	0.64
5	13.62	2.08	0.42
6	10.17	2.79	0.47
7	7.34	3.87	0.55
8	5.38	5.28	0.66