An SMP Soft Classification Algorithm for Remote Sensing

Rhonda D. Phillips*, Layne T. Watson, David R. Easterling, Randolph H. Wynne

Virginia Polytechnic Institute and State University,

Departments of Computer Science, Mathematics, and Forestry

***Abstract*– This work introduces a symmetric multiprocessing (SMP) version of the continuous iterative guided spectral class rejection (CIGSCR) algorithm, a semiautomated classification algorithm for remote sensing (multispectral) images. The algorithm uses soft data clusters to produce a soft classification containing inherently more information than a comparable hard classification at an increased computational cost. Previous work suggests that similar algorithms achieve good parallel scalability, motivating the parallel algorithm development work here. Experimental results of applying parallel CIGSCR to an image with approximately 10^8 pixels and six bands demonstrate superlinear speedup. A soft two class classification is generated in just over four minutes using 32 processors.**

Keywords: **remote sensing, semisupervised clustering, classification, IGSCR**

1. Introduction

The land cover classification of remote sensing images is fundamental to many scientific problems but is unfortunately laborious and computationally expensive. Land cover classifications are laborious because they either require hand labeling a set of training data prior to classification or manually labeling clusters after classification. The identification of good training data is difficult because each class should have within class similarity and dissimilarity to other classes. Furthermore, the set of training classes needs to span the range of data contained in the image to be classified. Techniques such as clustering can automatically find good training classes with these properties, but labeling these clusters with land cover class labels is nontrivial. Labeling can be further complicated if a particular cluster is composed of multiple classes. Supervised classifications that use training data can be computationally efficient as the same operation is applied to each pixel within an image. However, remote sensing images often contain billions of pixels, each described using tens or hundreds of bands, making classification computationally expensive. Partially supervised and unsupervised methods that make use of clustering algorithms are even more computationally expensive.

Partially supervised classification methods that use clustering and statistical methods to label clusters save human time by lessening training data requirements. These methods combine the automation of clustering with automatic labeling methods that leverage statistical tests and a limited amount of training data. This limited training data can be smaller, lessening the training burden, and even further, the training class requirements are substantially reduced. The classes are no longer required to have within class similarity and between class dissimilarity as sub-clusters address those issues. Similarly, the issue associated with having training data that is entirely representative of the image can be mitigated through clustering and labeling. This reduction in analyst burden is offset by an increase in computational requirements as clustering requires more processing and computer memory resources. The computation requirements increase even more when soft clustering and classification is required.

Soft clustering and classification methods are required to effectively model pixels or samples that can be assigned to multiple classes either because of class assignment uncertainty or multiple class memberships. Sensors with low spatial resolution result in pixels that can reasonably contain multiple land cover classes. Additionally, high spectral resolution (hyperspectral) image pixels are often modeled as mixtures of pure spectral elements or classes. Mixed membership and uncertainty are also incorporated in soft clustering methods, where each sample has partial membership in multiple clusters. These methods allow more flexibility in finding clusters at the

*Corresponding author. Tel. +1 781 981 1065.

E-mail addresses: rhonda.phillips@ll.mit.edu, ltw@ieee.org, dreast@vt.edu, wynne@vt.edu

boundary between land cover classes. Soft clusters and soft endmembers are frequently used in hyperspectral image classification.

Methods for endmember identification tend to be either manually intensive or require parameter tuning that can be just as manually intensive to achieve good classification results. However, the continuous iterative guided spectral class rejection (CIGSCR) algorithm leverages soft clustering to automatically find good training spectral classes/endmembers using a limited set of training data. CIGSCR is comparably insensitive to parameters. This is achieved through a series of clustering iterations that automatically evaluate and label soft clusters and then further refine them. This clustering iteration is computationally expensive, but can leverage parallel architectures to reduce execution times. In order to reduce the execution time required to run the highly automated CIGSCR algorithm, this paper introduces a parallel version. The next section outlines prior work that indicates CIGSCR is a good candidate for parallel speedup and useful to the remote sensing community. Section 3 describes CIGSCR, and Section 4 describes the parallel algorithm. Section 5 includes experimental results and discussion, and Section 6 concludes the paper.

2. Background

Partially supervised classification methods leverage the automation of clustering to overcome challenges in training data collection. For example, in hyperspectral image classification with a large number of spectral dimensions, clustering can mitigate the Hughes' phenomenon, or classification overfitting (Shahshahani and Landgrebe, 1994). Additionally, unsupervised spectral class formation can locate missing spectral classes (Mantero et al., 2005), and unsupervised approaches can be used to locate all classes when training data is only present for one class of interest (Gomez-Chova et al., 2004, Jeon and Landgrebe, 1999, Sanchez-Hernandez et al., 2007). These techniques proved to be suitable for the identification of one class of interest using supervised classification, with the advantage of allowing an analyst to focus training resources on only the class of interest (Sanchez-Hernandez et. al., 2007). In some cases, spectral class training data is unavailable for some images, but exists for similar images/areas of interest such as multiple images taken of the same scene at different times. Several partially supervised spectral class identification approaches are proposed to leverage an existing training data set (Rajan et al., 2006, Rajan et al., 2008, Cossu et al., 2005, Bruzzone and Prieto, 2001).

Partially supervised methods are also used to divide the land cover classes identified in the training data into spectral classes that are suitable for classification. Perhaps one of the first such algorithms is described in (Fleming et al., 1975). Clustering is used to define spectral classes, and clusters are manually analyzed using a variety of statistical methods to determine the spectral classes to be used for classification. Guided clustering applies a clustering algorithm directly to the training data to identify spectral classes (Bauer et al., 1994). This method does not address the possibility that spectral classes that are present in the image may not be well represented by the training data. The iterative guided spectral class rejection (IGSCR) classification method (Wayman et al., 2001, Musy et al., 2006, Phillips et al., 2007) addresses this issue by clustering the entire image. The clusters are labeled using a statistical test that selects the land cover class that best describes the data in each cluster. If a cluster is confused (contains multiple classes), it is refined in subsequent iterations. Due to its high accuracy and automation, IGSCR is a frequently used partially supervised classification method in the remote sensing community (Jiang et al., 2004, Kelly et al., 2004, Sivanpillai et al., 2005, Wynne et al., 2007). Unlike the previously mentioned algorithms that leverage hard clustering algorithms and are therefore constrained to locating hard clusters, CIGSCR uses soft clustering. CIGSCR can locate more spectral classes (especially at land cover class boundaries) than IGSCR and often produces more accurate classifications (Phillips et al., 2012).

Partially supervised classification methods that leverage clustering are good candidates for parallelization as many parallel clustering algorithms exist. Unfortunately, few remote sensing classification algorithms have been implemented in parallel. Typical clustering algorithms that are used for remote sensing classification such as K-means, fuzzy K-means (a soft version of K-means), and ISODATA have parallel counterparts (Dhillon and Modha, 2002, Dhodhi et al., 1999, Kwok et al., 2002). A parallel version of IGSCR incorporates parallel K-means to achieve superlinear parallel speedup (Phillips et al., 2007). As CIGSCR is similar to IGSCR and built on similar parallel algorithms, a parallel version of CIGSCR should achieve similarly good parallel speedup. The next section describes the CIGSCR classification algorithm.

3. Continuous IGSCR

CIGSCR uses a soft clustering algorithm to locate soft clusters within an image, and then applies a statistical test to each cluster in order to assign a land cover class label. CIGSCR begins by applying a soft clustering

algorithm to all of the pixels in a particular image, forming soft clusters. Each pixel has a positive probability of belonging to each cluster, and the sum of one pixel's probabilities across all clusters equals one. In the hard clustering case, the clusters represent a group of mathematically similar pixels, and the cluster mean is representative of those pixels and of a spectral class within the image. The interpretation of soft clustering is that cluster means are still representative of spectral classes within the image, and that pixels are mixtures of those pure spectral classes. This may be a more accurate and realistic model of real image data. Once these clusters are identified, they have to be associated to the dominant class within the cluster, determined to be the class with the highest average probability of belonging to the cluster. This is determined using a set of labeled training data. All training data have class labels, and all training data are assigned probabilities to each cluster. In this manner, labeled training data from one class can be compared to training data from other classes for a specific cluster.

The test used to determine which clusters should be used for classification is a statistical hypothesis test. The hypothesis test measures whether a particular class is significantly stronger in a particular cluster than other classes. When hard cluster assignments are used, this test can use discrete probability distributions to model the data. CIGSCR uses soft cluster assignments, so a statistical distribution that takes the range of possible cluster assignment values (between 0 and 1) into account is necessary. The statistic used for the test for the cth class and the jth cluster is

$$\hat{z} = \frac{\sqrt{n_c}(\overline{w}_{c,j} - \overline{w}_j)}{S_{\overline{w}_j}}, \tag{1}$$

where $\hat{z}$ is assumed to be drawn from a normal distribution, n_c is the number of training samples in the cth class, $\overline{w}_{c,j}$ is the average probability of samples in the cth class of belonging to the jth cluster, $\overline{w}_j$ is the average probability of all samples belonging to the jth cluster, and $S_{\overline{w}_j}$ is the sample standard deviation of the probabilities associated to the jth cluster. If $P(Z > \hat{z}) < \alpha$, the dominant class (the cth class) within the jth cluster is significant, and the cluster does not require further refinement. α is the type-I error.

The clusters that do not pass the test are subjected to refinement in an iterative procedure. As long as there are clusters that fail the test, the cluster that had the lowest $P(Z > \hat{z})$ value is selected for splitting. The new cluster means are selected to be the mean value of the dominant class within the cluster and the former mean value of the cluster. All data are reassigned to the clusters, including the new cluster mean, and because the new cluster mean was seeded using the information from one class, it is likely to be representative of that class. This iterative process terminates when all clusters pass the test and all classes are represented by clusters, or a maximum number of iterations is reached.

Using the soft clusters, a soft classification is produced called the iterative stacked (IS) classification. The classification function for IS classification is

$$\text{IS}(x) = p(c_i|x) = \sum_{j=1}^{K} p(c_i|k_j, x)p(k_j|x), \tag{2}$$

where $p(k_j|x)$ is the probability that a pixel belongs to cluster k_j and $p(c_i|k_j, x) = 1$ if the k_jth cluster is assigned to the c_ith class and equals zero, otherwise. The statistics of these clusters can also be used to train a supervised classifier, such as the maximum likelihood decision rule assuming Gaussian data. This is called the decision rule (DR) classification. The classification function for the DR classification is the same as the IS classification in (2), but $p(k_j|x)$ is estimated using $p(x|k_j, U_j, \Sigma_j)$ (using Bayes' rule) and assuming k_j is a normal distribution with estimated mean U_j and estimated covariance Σ_j.

The full CIGSCR algorithm is listed below for completeness.

Algorithm CIGSCR
Input: X % 3-dimensional multispectral image
$\{(i, j, \phi(i, j))\}$ % training data: set of indices and class labels
$Kinit$ % number of initial clusters
$Kmax$ % maximum number of clusters
C % number of classes
ϵ % convergence threshold
α % Type-I error for one-sided hypothesis test
Output: DR % decision rule classification
IS % iterative stacked classification

begin

% Initialization

Initialize cluster means U along the the axis defined by the mean plus or minus the standard deviation of the image X;

$K := Kinit$;

% Determine number of training samples per class

for $c := 1$ **step** 1 **until** C **do**

$n_c :=\mid \phi^{-1}(c) \mid$;

% Begin Iteration

for $iteration := Kinit$ **step** 1 **until** $Kmax$ **do**

begin

$w := 0$; $convergence := 1$;

while $convergence > \epsilon$ **do**

begin

% Cluster Data

$num := 0; denom := 0$;

for $i := 1$ **step** 1 **until** $rows$ **do**

for $j := 1$ **step** 1 **until** $cols$ **do**

for $k := 1$ **step** 1 **until** K **do**

begin

$$\hat{w}_{ij,k} := \frac{1/||X^{(ij)} - U^{(k)}||_2^2}{\sum_{l=1}^{K} 1/||X^{(ij)} - U^{(l)}||_2^2};$$

% update sums for mean calculations.

$num^{(k)} := num^{(k)} + \hat{w}_{ij,k}^2 X^{(ij)}$;

$denom_k := denom_k + \hat{w}_{ij,k}^2$;

end

% update cluster means

for $k := 1$ **step** 1 **until** K **do**

$$U^{(k)} := \frac{num^{(k)}}{denom_k};$$

$convergence := \max_{i,j,k} |w_{ij,k} - \hat{w}_{ij,k}|$;

$w := \hat{w}$;

end

% Determine Good Clusters

for $k := 1$ **step** 1 **until** K **do**

begin

for $cl := 1$ **step** 1 **until** C **do**

$$\overline{w}_{cl,k} := \frac{\sum_{i,j \in \phi^{-1}(cl)} w_{ij,k}}{n_{cl}};$$

end

$c_k := \operatorname{argmax}_c \overline{w}_{c,k}$;

$$sw_k := \sum_{i=1}^{rows} \sum_{j=1}^{cols} w_{ij,k};$$

$$ssw_k := \sum_{i=1}^{rows} \sum_{j=1}^{cols} w_{ij,k}^2;$$

$$\overline{w}_k := \frac{sw_k}{rows * cols};$$

$s_{\overline{w}_k} := \sqrt{\dfrac{ssw_k - \dfrac{sw_k^2}{rows * cols}}{rows * cols - 1}}$;

$\hat{Z}_k := \dfrac{\sqrt{n_c}(\overline{w}_{c,k} - \overline{w}_k)}{s_{\overline{w}_k}}$;

end

if any class is not associated with a cluster **then**

begin

$c :=$ first unassociated class;

$k := \underset{k}{\text{argmax}} \dfrac{\overline{w}_{c,k}}{w_{c_k,k}}$;

$K := K + 1$;

$U^{(K)} = \dfrac{\sum\limits_{ij \in \phi^{-1}(c)} w_{ij,k} X^{(ij)}}{\sum\limits_{ij \in \phi^{-1}(c)} w_{ij,k}}$;

end

elseif (any$(\hat{Z}_k < Z(\alpha), k = 1, \ldots, K)$ **then**

begin

$k := \text{argmin}_k \hat{Z}_k$;

$K := K + 1$;

$U^{(K)} = \dfrac{\sum\limits_{ij \in \phi^{-1}(c_k)} w_{ij,k} X^{(ij)}}{\sum\limits_{ij \in \phi^{-1}(c_k)} w_{ij,k}}$;

end

else

exit for loop;

end

end

for $k := 1$ **step** 1 **until** K **do**

begin

% initialize for covariance calcs.

$\Sigma_k := 0$;

$denom_k := 0$;

end

% IS classification

for $i := 1$ **step** 1 **until** $rows$ **do**

for $j := 1$ **step** 1 **until** $cols$ **do**

begin

$csum := 0$;

for $k := 1$ **step** 1 **until** K **do**

if $(\hat{Z}_k > Z(\alpha))$ **then**

$csum_{c_k} := csum_{c_k} + w_{ij,k}$;

for $cl := 1$ **step** 1 **until** C **do**

$IS_{ij,cl} := \dfrac{csum_{cl}}{\sum\limits_{k=1}^{C} csum_k}$;

% calculate covariance matrices

for $k := 1$ **step** 1 **until** K **do**

begin

$\Sigma_k := \Sigma_k + w_{ij,k}$

$\cdot(X^{(ij)} - U^{(k)})(X^{(ij)} - U^{(k)})^T$;
$denom_k := denom_k + w_{ij,k}$;
end
end
for $k := 1$ **step** 1 **until** K **do**
$\Sigma_k := 1/denom_k \cdot \Sigma_k$;
% DR classification
for $i := 1$ **step** 1 **until** $rows$ **do**
for $j := 1$ **step** 1 **until** $cols$ **do**
begin
$csum := 0$;
for $k := 1$ **step** 1 **until** K **do**
if $(\hat{Z}_k > Z(\alpha))$ **then**
begin
$p := \frac{2e^{-\frac{1}{2}(X^{(ij)}-U^{(k)})^T\Sigma_k^{-1}(X^{(ij)}-U^{(k)})}}{\pi^{B/2}|\Sigma_k|^{\frac{1}{2}}}$;
$csum_{c_k} := csum_{c_k} + p$;
else
$csum_{c_k} := 0$;
end
for $cl := 1$ **step** 1 **until** C **do**
$DR_{ij,cl} := \frac{csum_{cl}}{\sum_{k=1}^{C} csum_k}$;
end
end

4. Parallel CIGSCR

The shared memory parallel CIGSCR algorithm is similar to parallel IGSCR. Both algorithms rely on a clustering algorithm based on K-means, and various parallel versions of K-means run well in parallel. The other expensive portions of both algorithms are the final classifications that assign each pixel to a class independently of other pixels' assignments. Because the assignment of each pixel is independent, this operation can be executed in parallel.

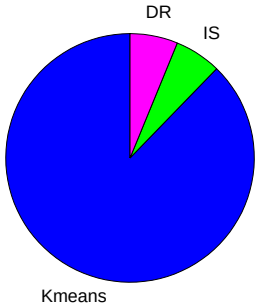

Fig. 1. Proportion of time devoted to each major operation in CIGSCR using one processor.

The most expensive part of CIGSCR (as shown in Figure 1) is the soft clustering implemented as fuzzy K-means. The execution time for fuzzy K-means, represented by a wedge in the pie chart in Figure 1, is the execution time required for multiple iterations of the already iterative clustering algorithm. Each clustering iteration involves the iterative (soft) assignment of each pixel to the clusters and the recalculation of cluster centers once the assignment is complete. The parallel version of fuzzy K-means performs the assignment of pixels to clusters in parallel. The mean values of the clusters are updated based on the pixel assignments, requiring the sum of pixel assignments to a cluster over all processors. This is implemented as a parallel reduction operation (a sum). Additionally, the cluster significance test requires cluster statistics that can be computed using a parallel sum reduction operation. A custom version of fuzzy K-means is implemented specifically for CIGSCR to tally the sums necessary to calculate $\hat{z}$ for each cluster, making the calculation of $\hat{z}$ efficient. Each computationally expensive iteration of fuzzy K-means is implemented in parallel in this manner, and each iteration is followed by the much less expensive cluster evaluation, which can also be performed in parallel by assigning different processors to evaluate different clusters.

Once the clusters have been identified, two classifications are applied using those clusters. Both classifications, like their counterparts in IGSCR, are applied to each pixel independently, making the classifications straightforward to implement in parallel. The pixels are divided evenly among the processors for both classifications. For the DR classification, the calculation of Σ_j should be performed prior to classification. Pixels are then divided among processors to apply the DR classification. The parallel CIGSCR algorithm is given below.

Algorithm Parallel CIGSCR
Input: X % 3-dimensional multispectral image
$\{(i, j, \phi(i, j))\}$ % training data: set of indices and class labels
$Kinit$ % number of initial clusters
$Kmax$ % maximum number of clusters
C % number of classes
ϵ % convergence threshold
α % Type-I error for one-sided hypothesis test
Output: DR % decision rule classification
IS % iterative stacked classification
begin
% Initialization
Initialize cluster means U in parallel along the the axis
defined by the mean plus or minus the standard deviation
of the image X;
$K := Kinit$;
% Determine number of training samples per class
for c := 1 **step** 1 **until** C **do**
$n_c :=\mid \phi^{-1}(c) \mid$;
% Begin Iteration
for $iteration := Kinit$ **step** 1 **until** $Kmax$ **do**
begin
$w := 0$; $convergence := 1$;
while $convergence > \epsilon$ **do**
begin
% Cluster Data
$num := 0; denom := 0$;
private: *i, j, k*
shared: *rows, cols, K,* $\hat{w}, w, X, U$
reduction: (+, num, $denom$)
for $i := 1$ **step** 1 **until** $rows$ **fork** CLUSTER
CLUSTER:
for $j := 1$ **step** 1 **until** $cols$ **do**
for $k := 1$ **step** 1 **until** K **do**
begin

$$\hat{w}_{ij,k} := \frac{1/||X^{(ij)} - U^{(k)}||_2^2}{\sum_{l=1}^{K} 1/||X^{(ij)} - U^{(l)}||_2^2};$$

% update sums for mean calculations.

$num^{(k)} := num^{(k)} + \hat{w}_{ij,k}^2 X^{(ij)}$;

$denom_k := denom_k + \hat{w}_{ij,k}^2$;

end

join *processes*;

% update cluster means

private: k

shared: $denom$, num, U, K

for $k := 1$ **step** 1 **until** K **fork** UPDATE

UPDATE:

$$U^{(k)} := \frac{num^{(k)}}{denom_k};$$

join *processes*;

$$convergence := \max_{i,j,k} |w_{ij,k} - \hat{w}_{ij,k}|;$$

$w := \hat{w}$;

end

% Determine Good Clusters

private: *i, j, k, cl*

shared: *rows, cols, C, K, w,* $\overline{w}_{(\cdot)}$, $\overline{w}_{(\cdot,\cdot)}$, $s_{\overline{w}_{(\cdot)}}$, $\hat{Z}$, sw, ssw, n, c

for $k := 1$ **step** 1 **until** K **fork** STAT

STAT:

begin

for $cl := 1$ **step** 1 **until** C **do**

$$\overline{w}_{cl,k} := \frac{\sum_{i,j \in \phi^{-1}(cl)} w_{ij,k}}{n_{cl}};$$

$$c_k := \underset{c}{\text{argmax}}\, \overline{w}_{c,k};$$

$$sw_k := \sum_{i=1}^{rows} \sum_{j=1}^{cols} w_{ij,k};$$

$$ssw_k := \sum_{i=1}^{rows} \sum_{j=1}^{cols} w_{ij,k}^2;$$

$$\overline{w}_k := \frac{sw_k}{rows * cols};$$

$$s_{\overline{w}_k} := \sqrt{\frac{ssw_k - \frac{sw_k^2}{rows * cols}}{rows * cols - 1}};$$

$$\hat{Z}_k := \frac{\sqrt{n_c}(\overline{w}_{c,k} - \overline{w}_k)}{s_{\overline{w}_k}};$$

end

join *processes*;

if any class is not associated with a cluster **then**

begin

$c :=$ first unassociated class;

$$k := \underset{k}{\text{argmax}}\, \frac{\overline{w}_{c,k}}{w_{c_k,k}};$$

$K := K + 1$;

$$U^{(K)} = \frac{\sum_{ij \in \phi^{-1}(c)} w_{ij,k} X^{(ij)}}{\sum_{ij \in \phi^{-1}(c)} w_{ij,k}};$$

end

elseif (any($\hat{Z}_k < Z(\alpha), k = 1, \ldots, K$) **then**

begin

$k :=$ argmin$_k\, \hat{Z}_k$;

$K := K + 1$;

$$U^{(K)} = \frac{\sum_{ij \in \phi^{-1}(c_k)} w_{ij,k} X^{(ij)}}{\sum_{ij \in \phi^{-1}(c_k)} w_{ij,k}};$$

end

else

exit for loop;

end

end

for $k := 1$ **step** 1 **until** K **do**

begin

% initialize for covariance calcs.

$\Sigma_k := 0$;

$denom_k := 0$;

end

% IS classification

private: *i, j, k, cl, csum*

shared: *rows, cols, C, K,* $\hat{Z}$, $Z(\alpha)$, IS, w, X, U, c

reduction: (+, Σ, $denom$)

for $i := 1$ **step** 1 **until** $rows$ **fork** ISCLASS

ISCLASS:

for $j := 1$ **step** 1 **until** $cols$ **do**

begin

$csum := 0$;

for $k := 1$ **step** 1 **until** K **do**

if $(\hat{Z}_k > Z(\alpha))$ **then**

$csum_{c_k} := csum_{c_k} + w_{ij,k}$;

for $cl := 1$ **step** 1 **until** C **do**

$$IS_{ij,cl} := \frac{csum_{cl}}{\sum_{k=1}^{C} csum_k};$$

% calculate covariance matrices

for $k := 1$ **step** 1 **until** K **do**

begin

$\Sigma_k := \Sigma_k + w_{ij,k}$

$\cdot (X^{(ij)} - U^{(k)})(X^{(ij)} - U^{(k)})^T$;

$denom_k := denom_k + w_{ij,k}$;

end

end

join *processes*;

for $k := 1$ **step** 1 **until** K **do**

$\Sigma_k := 1/denom_k \cdot \Sigma_k$;

% DR classification

private: *i, j, k, cl, csum, p*
shared: *rows, cols, C, K,* $\hat{Z}$, $Z(\alpha)$, DR, Σ, X, U, c
for $i := 1$ **step** 1 **until** $rows$ **fork** DRCLASS
DRCLASS:
for $j := 1$ **step** 1 **until** $cols$ **do**
begin
$csum := 0$;
for $k := 1$ **step** 1 **until** K **do**
if $(\hat{Z}_k > Z(\alpha))$ **then**
begin

$$p := \frac{2e^{-\frac{1}{2}(X^{(ij)}-U^{(k)})^T \Sigma_k^{-1}(X^{(ij)}-U^{(k)})}}{\pi^{B/2}|\Sigma_k|^{\frac{1}{2}}};$$

$csum_{c_k} := csum_{c_k} + p$;
else
$csum_{c_k} := 0$;
end
for $cl := 1$ **step** 1 **until** C **do**

$$DR_{ij,cl} := \frac{csum_{cl}}{\sum_{k=1}^{C} csum_k};$$

end
join *processes*;
end

5. Experimental Results and Discussion

The classification algorithm is applied to two representative classification problems in order to demonstrate the scalability of parallel CIGSCR. First, a two-class classification is applied to a multispectral image, where the classes of interest are forest and nonforest. This image has many rows and columns (millions of pixels), but only six bands. The second classification problem uses a hyperspectral image to determine land cover classes for three species of pines and a nonpine class. This dataset has fewer rows and columns, but many more bands. Additionally, the discrimination between spectrally similar pine classes requires many clusters. The first dataset is less computationally expensive than the second dataset. Performance on a range of problem sizes can be inferred by testing parallel CIGSCR on this set of problems. The algorithms described above were implemented using Fortran 95, LAPACK (Linear Algebra PACKage), and OpenMP, and executed on a SunFire X4600 M2 with a total of 64 GB of RAM (2 GB per core) and eight quad-core 2.0 GHz AMD Opteron Model 8356 processors.

5.1. Multispectral Classification Results

The first image classified using Parallel CIGSCR is a mosaicked Landsat Enhanced Thematic Mapper Plus (ETM+) satellite image taken from Landsat Worldwide Reference System (WRS) path 17, row 34, located in Virginia, USA, shown in Figure 2. This image contains approximately 10^8 pixels and six bands (six brightness values per pixel). This image, hereafter referred to as Virginia17, was obtained on November 2, 2003 and consists largely of forested, mountainous regions, and a few developed regions that are predominantly light blue and light pink in Figure 2. Figure 2 is a three color representation of Virginia17 where the red color band in Figure 2 corresponds to the near infrared wavelength in Virginia17, the green color band in Figure 2 corresponds to the red wavelength in Virginia17, and the blue color band in Figure 2 corresponds to the green wavelength in Virginia17.

The training data for this image was created by the interpretation of point locations from a systematic, hexagonal grid over Virginia Base Mapping Program (VBMP) true color digital orthophotographs. A two-class classification was performed (forest/nonforest) using 25 initial clusters and a maximum of 30 clusters (30 clusters were used in the final classifications). The α value for the cluster test was 0.0001. Figures 3 and 4 contain the IS and DR classification results, respectively. The serial version of CIGSCR was executed and compared to the parallel version using up to 32 processors as shown in Table 1. The total amount of time required to run Parallel CIGSCR on the test image using 32 processors is about four minutes (for an image with approximately 10^8 pixels and six

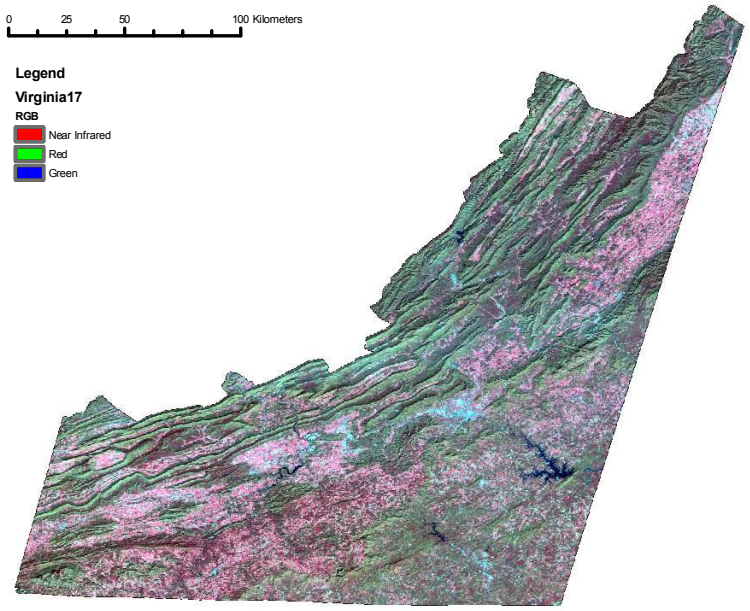

Fig. 2. Landsat ETM+ satellite image Virginia17.

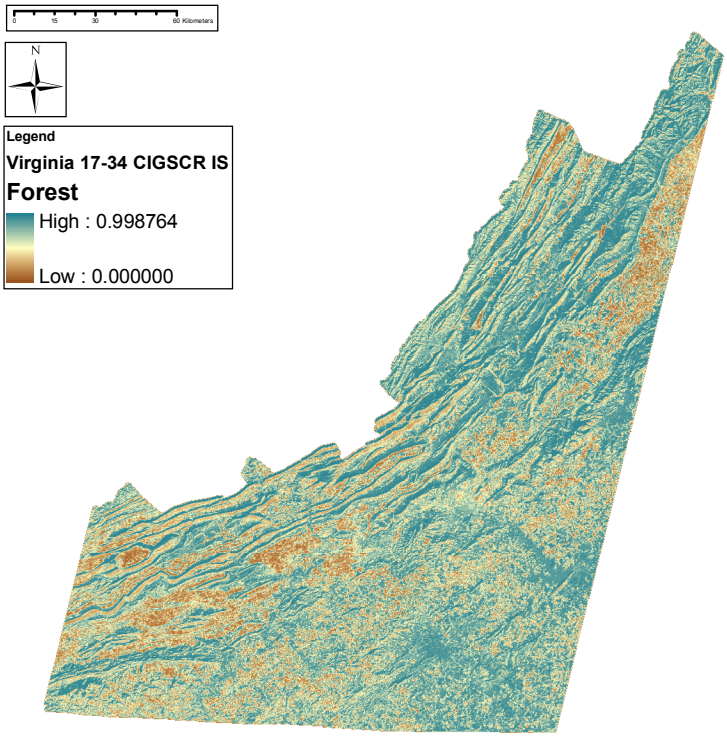

Fig. 3. CIGSCR IS (forest is green, nonforest is brown) classification of Virginia17.

bands). Also note that classification requires 891 seconds (or about 15 minutes) using eight processors. Many modern desktop computers contain eight cores.

In order to measure parallel speedup, the serial execution time of CIGSCR was compared to the execution times of Parallel CIGSCR using two through 32 processors. Parallel speedup for p processors is defined as the serial execution time (using one processor) divided by the observed parallel execution time using p processors.

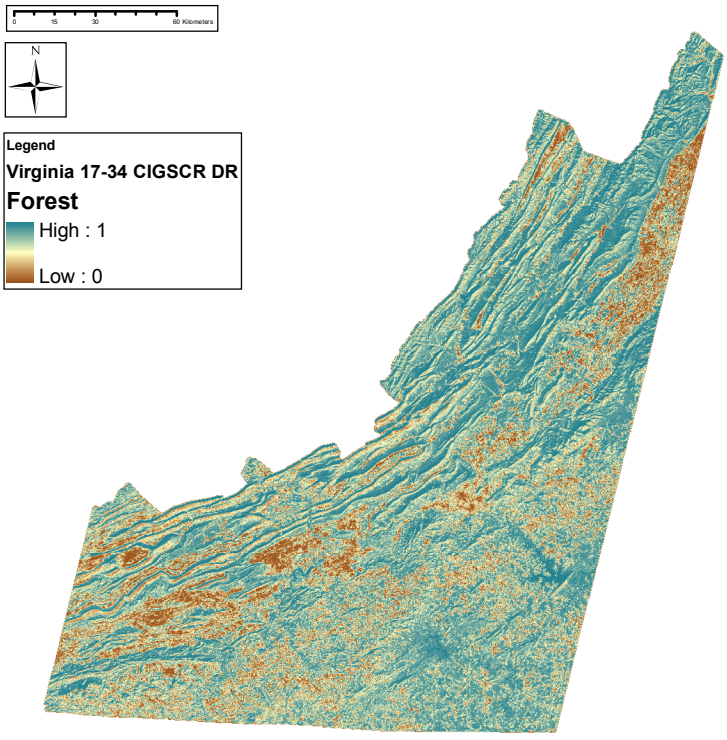

Fig. 4. CIGSCR DR (forest is green, nonforest is brown) classification of Virginia17.

Table 1
Parallel CIGSCR execution times in seconds—Virginia17.

no. processors	K-means	IS	DR	Total
1	7467	516	528	8524
2	3013	244	243	3611
4	1510	138	117	1776
6	1007	89	77	1184
8	756	67	58	891
10	605	53	46	714
12	504	44	38	596
14	432	37	32	512
16	378	37	28	454
18	337	35	26	408
20	303	33	24	370
22	276	33	24	343
24	253	31	22	316
26	233	34	24	302
28	217	36	24	287
30	202	35	23	271
32	190	38	24	**262**

Ideal speedup for p processors is p, indicating that p processors can evenly divide the work of one processor. Observed speedup is rarely ideal as most parallel algorithms contain some serial sections, and even algorithms that parallelize perfectly incur parallel overhead when implemented. Computed speedup results are shown in Figure 5.

Table 2
Parallel CIGSCR execution times in seconds—ABpines.

no. processors	K-means	IS	DR	Total
1	18630	20	316	18971
2	8677	10	159	8850
4	4046	5	79	4133
6	2624	3	53	2685
8	1912	2	41	1960
10	1508	2	33	1548
12	1242	2	28	1276
14	1059	1	25	1089
16	905	1	22	932
18	798	1	20	823
20	718	1	18	741
22	642	1	16	663
24	585	1	15	605
26	538	1	14	557
28	502	1	13	520
30	464	1	12	482
32	434	1	12	**450**

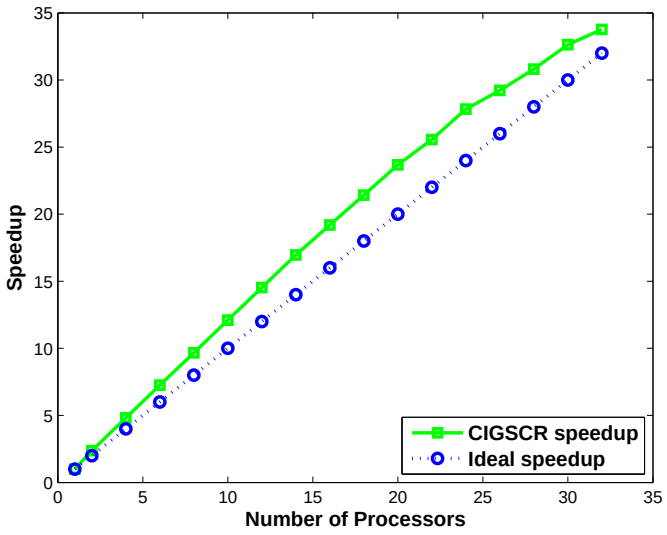

Fig. 5. CIGSCR parallel speedup compared to execution time of serial IGSCR, Virginia17.

5.2. Hyperspectral Classification Results

The second image was collected using the airborne AVIRIS sensor over the Appomattox-Buckingham state forest, hereafter referred to as ABpines and shown in Figure 6. In situ surveys were conducted to acquire ground truth for three different species of pines, loblolly pines, shortleaf pines, and Virginia pines. In addition to these ground truth, known tree stand maps and a region growing algorithm were used to generate additional training data. An "other" class was created using all of these data sources to encompass regions in the scene that did not fall into the pine categories, namely deciduous forest and nonforest. The final training set has 5257 points in loblolly, 439 points in shortleaf, 1920 in Virginia pines, and 7357 in other. The final CIGSCR classification is shown in Figure 7. The CIGSCR parameters used to generate these results are 100 clusters and $\alpha = .0001$. Speedup is shown in Figure 8.

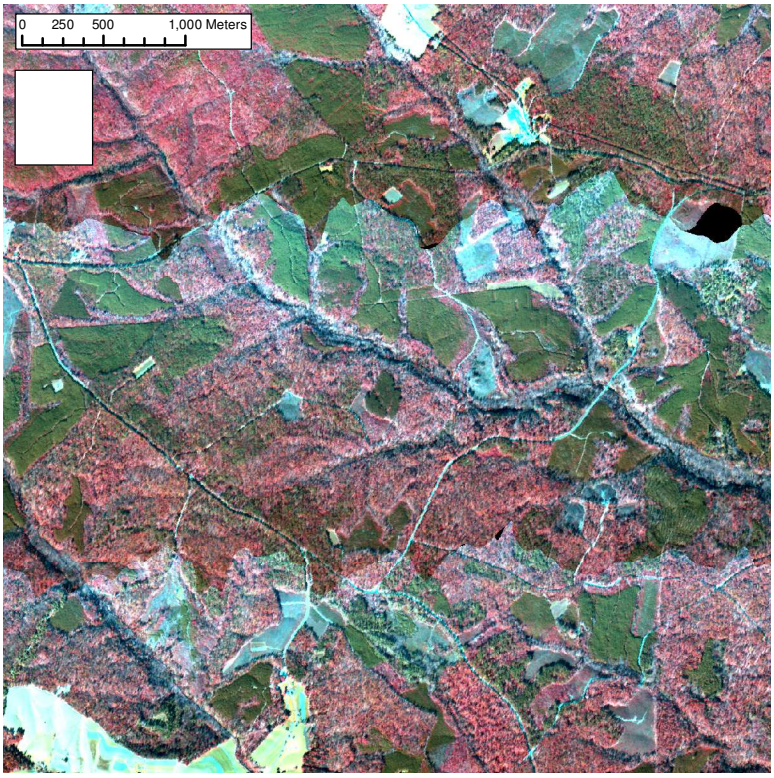

Fig. 6. Landsat ETM+ satellite image, ABpines.

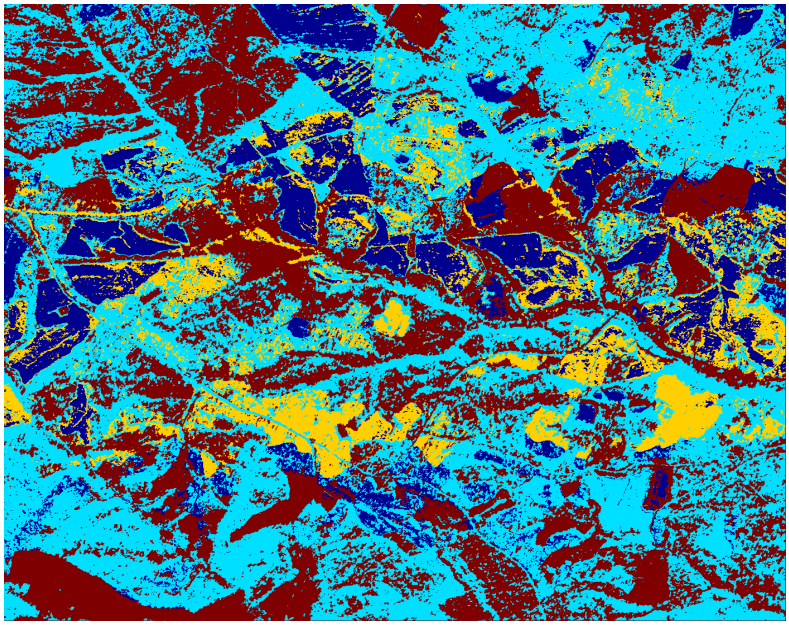

Fig. 7. Final CIGSCR classification, ABpines.

6. Discussion

This implementation of parallel CIGSCR efficiently uses a shared memory architecture to achieve better than ideal parallel speedup. The measured speedup in Figures 5 and 8 is higher than ideal speedup and is called superlinear speedup. Ideal speedup indicates that using p processors instead of one should reduce the execution

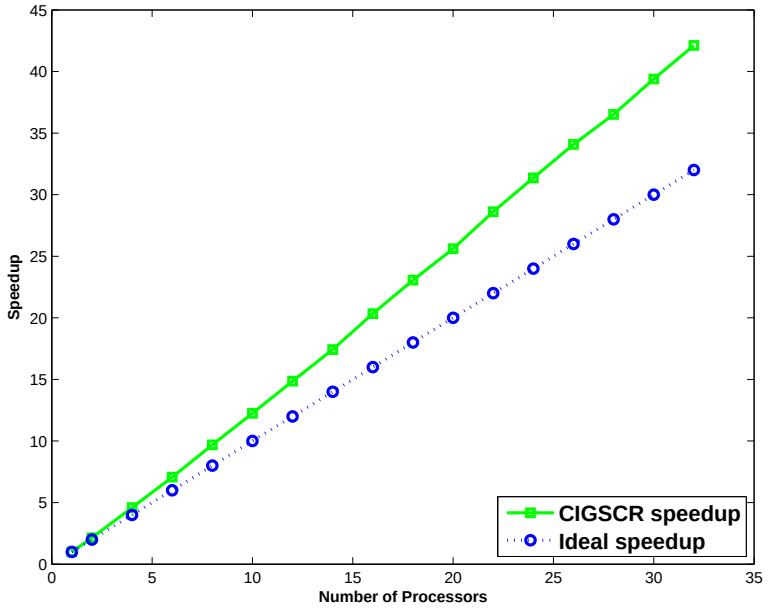

Fig. 8. CIGSCR parallel speedup compared to execution time of serial IGSCR, ABpines.

time by a factor of p. Speedup that is better than ideal is caused by factors other than additional processors such as memory. The SunFire X4600 M2 has a nonuniform memory access (NUMA) architecture, meaning that even though there is global memory that is accessible from each processor, there are physical sections of memory that are closest to each processor, which can affect memory access times. Furthermore, each processor has local cache memory, and additional processors used for classification contribute additional cache memory.

In order to take advantage of nonuniform global memory and individual processors' cache memory, it is necessary to consider the physical data layout when implementing an algorithm. This consideration is mandatory in distributed memory programming as data is manually divided and distributed across processors. One advantage of shared memory programming is that consideration of data distribution is not required. However, using shared memory effectively often requires understanding how the data is divided and placed into memory. In distributed memory programming, this is explicit. On most shared memory systems including the SunFire X4600, data is physically stored closest to the processor that first accesses it. Good memory access requires that shared data is accessed each time in code using the same data access patterns, and each process/local memory is bound to a particular processor. The superlinear speedup achieved in this paper results from leveraging these coding techniques. Namely, the parallel processes that fork for the CLUSTER, ISCLASS, AND DRCLASS loops access X, IS, and DR the same way. Finally, in order to take full advantage of cache memory, all multidimensional data such as the image X is accessed in the order it is stored in memory.

Although superlinear speedup is desirable, it can obscure scalability issues. In order to determine if the algorithm will scale beyond 32 processors, compare execution times on p processors to execution times for two processors. The intuition behind this analysis is that two processors have more cache than one processor, and therefore speedup is due to the increased processing power and not increased cache memory. Once the number of processors used has enough cache to efficiently access and process the data, adding more cache does not decrease execution times. The comparison of execution times to what is observed using two processors is shown in Figures 9 and 10 for the data sets Virginia17 and ABpines, respectively.

The parallel scalability (defined as the speedup for p processors with respect to two processors, divided by $p/2$) of parallel CIGSCR depends on clustering scalability, and clustering scalability in turn depends on data size. Notice in Figure 9 that scalability (compared to two processors) is ideal through 24 processors, but then tapers off through 32 processors. However, scalability in Figure 10 is superideal (> 1) through 32 processors with no indication of a dropoff. The main differences between these two datasets is that ABpines corresponding to Figure 10 has many more bands and the resulting classification uses many more clusters. The increases the dimensionality of both the input image, X, and the cluster memberships, w. Modest increases in either the number of input bands or the number of clusters require additional memory. The ABpines classification problem requires large amounts

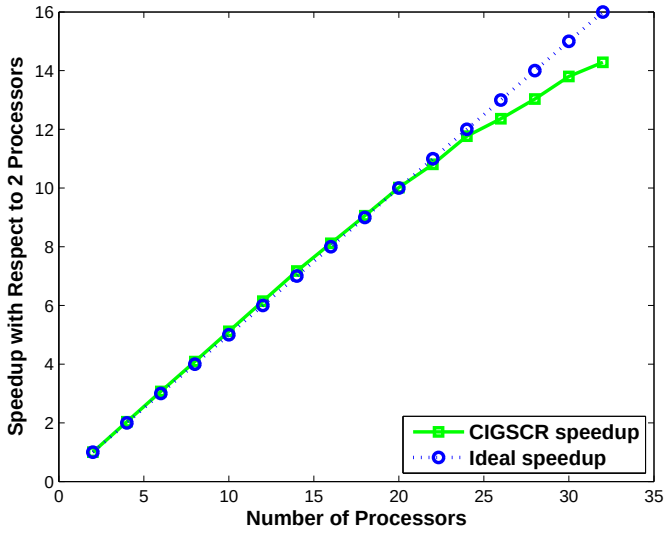

Fig. 9. CIGSCR parallel speedup compared to execution time of two processors, Virginia17.

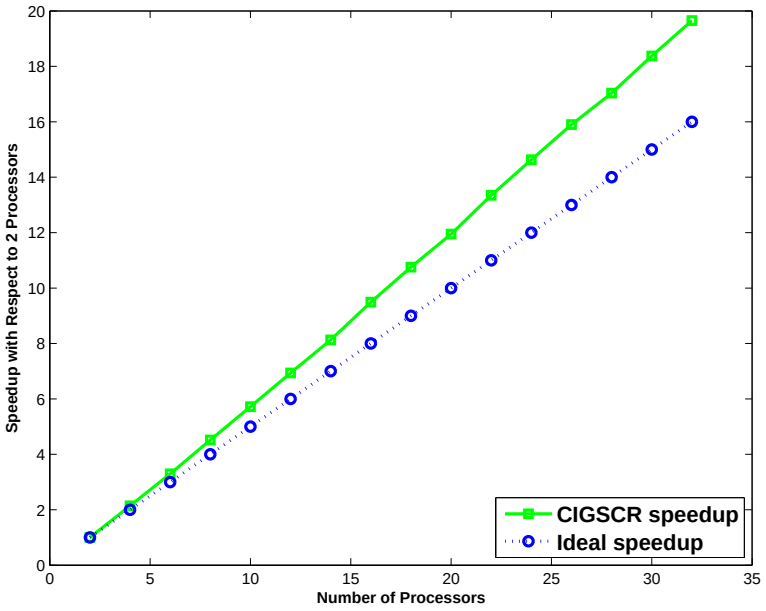

Fig. 10. CIGSCR parallel speedup compared to execution time of two processors, ABpines.

of RAM and cache, fully utilizing this kind of shared memory architecture. Parallel CIGSCR is likely to scale efficiently for many more processors for any representative hyperspectral classification problem. Furthermore, as spectral and spatial resolution of sensors will continue to increase and the number of processors included in computers will increase, parallel CIGSCR will continue to be relevant.

Perhaps more important than potential speedup beyond 32 processors is the potential scientific value of Parallel CIGSCR on a modest multicore computer. 32 processors produce a two class soft classification in about four minutes, but as few as eight processors bring the execution time down to 15 minutes (see Table 1) from 2.5 hours. This drastic time reduction will allow scientists to perform more classifications with different parameter values to gain better insight into the classification. Scientists can spend more time analyzing results rather than waiting for results.

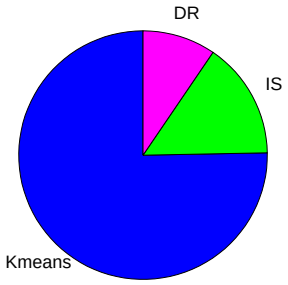

Fig. 11. Proportion of time devoted to each main operation in CIGSCR with 32 processors.

6. Conclusions

This work introduced a parallel version of the CIGSCR classification algorithm. CIGSCR uses soft clustering to produce soft classifications, which are more computationally expensive than hard classifications but provide more information. This parallel version of CIGSCR achieves superlinear (better than ideal) speedup using up to 32 processors, bringing the execution time required for a six-band image containing approximately 10^8 pixels from over 2 hours to about 4 minutes. Additional processors are likely to reduce the execution time even more. Regardless, 32 processors substantially reduce the execution time required to produce a soft classification of such a large image, enabling scientists to spend more time on science and less time waiting for classification results.

Acknowledgements

This research was supported by NASA (NAG5-10548), Department of Energy (DE-FGO2-O6ER2572O), and the Department of Computer Science, Virginia Polytechnic Institute and State University. The authors gratefully acknowledge remote sensing technical assistance by Dr. Christine E. Blinn, Department of Forestry, Virginia Polytechnic Institute and State University and provision of testing and validation data by Dr. John A. Scrivani, Department of Forestry, Commonwealth of Virginia.

References

Bauer, M.E., Burk, T.E., Ek, A.R., Coppin, P.R., Lime, S.D., Walsh, T.A., Walters, D.K., Befort, W., and Heinzen, D.F., 1994, "Satellite inventory of Minnesota forest resources," Photogrammetric Engineering & Remote Sensing, 60(3), 287–298.

Bernstein, A.J., 1966, "Analysis of programs for parallel processing," IEEE Transactions on Electronic Computers, 15(5), 757–763.

Bruzzone, L. and Prieto, D.F., 2001, "Unsupervised retraining of a maximum likelihood classifier for the analysis of multitemporal remote sensing images," IEEE Transactions on Geoscience and Remote Sensing, 39(2), 456–460.

Cossu, R., Chaudhuri, S., Bruzzone, L., 2005, "A context-sensitive Bayesian technique for the partially supervised classification of multitemporal images," IEEE Geoscience and Remote Sensing Letters, 2(3), 352–356.

Dhillon, I.S. and Modha, D.S., 2002, "A data-clustering algorithm on distributed memory multiprocessors," Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence, 1759, 245–260.

Dhodhi, M.K., Saghri, J.A., Ahmad, I., and Ul-Mustafa, R., 1999, "D-ISODATA: A distributed algorithm for unsupervised classification of remotely sensed data on network of workstations," Journal of Parallel and Distributed Computing, 59, 280–301.

Fleming, M.D., Berkebile, J.S., Hoffer, R.M., 1975, Computer-aided analysis of LANDSAT-1 MSS data: A comparison of three approaches, including a "modified clustering" approach, LARS Technical Reports, Paper 96.

Gomez-Chova, L., Fernandez-Prieto, D., Calpe, J., Soria, E., Vila, J., and Camps-Valls, G., 2004, Partially supervised hierarchical clustering of SAR and multispectral imagery for urban areas monitoring, in Proc. of the Society of Photo-optical Instrumentation Engineers (SPIE), Conference on Image and Signal Processing for Remote Sensing X, 138–149.

Jeon, B., and Landgrebe, D.A., 1999, "Partially supervised classification using weighted unsupervised clustering," IEEE Transactions on Geoscience and Remote Sensing, 37(2), 1073–1079.

Jiang, H., Strittholt, J.R., Frost, P.A., and Slosser, N.C., 2004, "The classification of late seral forests in the Pacific Northwest, USA using Landsat ERM+ imagery," Remote Sensing of Environment, 91(3–4), 320–331.

Kelly, M., Shaari, D., Guo, Q.H., and Liu, D.S., 2004, "A comparison of standard and hybrid classifier methods for mapping hardwood mortality in areas affected by "sudden oak death"," Photogrammetric Engineering & Remote Sensing, 70(11), 1229–1239.

Kwok, T., Smith, K., Lozano, S., and Taniar, D., 2002, "Parallel fuzzy c-means clustering for large data sets" in Lecture Notes in Computer Science vol. 2400, Springer, Berlin, 365–374.

Kumar, A., Ghosh, S.K., and Dadhwal, V.K., 2007, "Full fuzzy land cover mapping using remote sensing data based on fuzzy c-means and density estimation," Canadian Journal of Remote Sensing, 33, 81–87.

Mantero, P., Moser, G., Serpico, S.B., 2005, "Partially supervised classification of remote sensing images through SVM-based probability density estimation," IEEE Transactions on Geoscience and Remote Sensing, 43(3), 559–570.

Musy, R.F., Wynne, R.H., Blinn, C.E., Scrivani, J.A. and McRoberts, R.E., 2006, "Automated forest area estimation via iterative guided spectral class rejection," Photogrammetric Engineering & Remote Sensing, 72, 949–960.

Pepe, M., Boschetti, L., Brivio, P.A., and Rampini, A., 2007, Accuracy benefits of a fuzzy classifier in remote sensing data classification of snow, in Proc. IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '07), VOLS 1–4, 492–497.

Phillips, R.D., Watson, L.T., and Wynne, R.H., 2007, "Hybrid image classification and parameter selection using a shared memory parallel algorithm," Computers & Geosciences, 33, 875–897.

Phillips, R.D., Watson, L.T., Wynne, R.H., and Ramakrishnan, N., 2012, "Continuous iterative guided spectral class rejection classification algorithm," IEEE Transactions on Geoscience and Remote Sensing, 50(6), 2303–2317.

Rajan, S., Ghosh, J., and Crawford, M.M., 2006, "Exploiting class hierarchies for knowledge transfer in hyperspectral data," IEEE Transactions on Geoscience and Remote Sensing, 44(11), 3408–3417.

Rajan, S., Ghosh, J., and Crawford, M.M., 2008, "An active learning approach to hyperspectral data classification," IEEE Transactions on Geoscience and Remote Sensing, 46(4), 1231–1242.

Richards, J.A. and Jia, X., 1999, Remote Sensing Digital Image Analysis, Springer, New York.

Sanchez-Hernandez, C., Boyd, D.S., and Foody, G.M., 2007, "One-class classification for mapping a specific land-cover class: SVDD classification of fenland," IEEE Transactions on Geoscience and Remote Sensing, 45(4), 1061–1073.

Sha, Z., Bai, Y., Xie, Y., Yu, M., and Zhang, L., 2008, “Using a hybrid fuzzy classifier (HFC) to map typical grassland vegetation in Xilin River Basin, Inner Mongolia, China,” International Journal of Remote Sensing, 29, 2317–2337.

Sivanpillai, R., Smith, C.T., Srinivasan, R., Messina, M.G., and Ben Wu, X., 2005, “Estimating regional forest cover in East Texas using enhanced thematic mapper (ETM plus) data,” Forest Ecology and Management, 218(1–3), 342–352.

Wayman, J.P., Wynne, R.H., Scrivani, J.A., and Reams, G.A., 2001, “Landsat TM-based forest area estimation using iterative guided spectral class rejection,” Photogrammetric Engineering & Remote Sensing, 67, 1155–1166.

Wynne, R.H., Joseph, K.A., Browder, J.O., and Summers, P.M., 2007, “Comparing farmer-based and satellite-derived deforestation estimates in the Amazon basin using a hybrid classifier,” International Journal of Remote Sensing, 28(6), 1299–1315.