Cluster Comput (2008) 11: 17–32
DOI 10.1007/s10586-007-0048-1

Parallel morphological/neural processing of hyperspectral images using heterogeneous and homogeneous platforms

Javier Plaza · Rosa Pérez · Antonio Plaza · Pablo Martínez · David Valencia

Received: 13 March 2007 / Accepted: 29 October 2007 / Published online: 29 November 2007

Abstract The wealth spatial and spectral information available from last-generation Earth observation instruments has introduced extremely high computational requirements in many applications. Most currently available parallel techniques treat remotely sensed data not as images, but as unordered listings of spectral measurements with no spatial arrangement. In thematic classification applications, however, the integration of spatial and spectral information can be greatly beneficial. Although such integrated approaches can be efficiently mapped in homogeneous commodity clusters, low-cost heterogeneous networks of computers (HNOCs) have soon become a standard tool of choice for dealing with the massive amount of image data produced by Earth observation missions. In this paper, we develop a new morphological/neural algorithm for parallel classification of high-dimensional (hyperspectral) remotely sensed image data sets. The algorithm's accuracy and parallel performance is tested in a variety of homogeneous and heterogeneous computing platforms, using two networks of workstations distributed among different locations, and also a massively parallel Beowulf cluster at NASA's Goddard Space Flight Center in Maryland.

Keywords Heterogeneous computing · Parallel performance · Neural networks · Hyperspectral imaging · Mathematical morphology

J. Plaza (✉) · R. Pérez · A. Plaza · P. Martínez · D. Valencia
Neural Networks & Signal Processing Group (GRNPS),
Computer Science department, University of Extremadura,
Avda. de la Universidad s/n, 10071 Cáceres, Spain
e-mail: jplaza@unex.es

1 Introduction

Many international agencies and research organizations are currently devoted to the analysis and interpretation of high-dimensional image data collected over the surface of the Earth [6]. For instance, NASA is continuously gathering hyperspectral images using Jet Propulsion Laboratory's Airborne Visible-Infrared Imaging Spectrometer (AVIRIS) [13], which measures reflected radiation in the wavelength range from 0.4 to 2.5 μm using 224 spectral channels at spectral resolution of 10 nm (see Fig. 1). The incorporation of hyperspectral instruments aboard satellite platforms is now producing a near-continual stream of high-dimensional remotely sensed data, and cost-effective techniques for information extraction and mining from massively large hyperspectral data repositories are highly required [1]. In particular, although it is estimated that several Terabytes of hyperspectral data are collected every day, about 70% of the collected data are never processed, mainly due to the extremely high computational requirements.

Several challenges still remain open in the development of efficient data processing techniques for hyperspectral image analysis [6]. For instance, previous research has demonstrated that the high-dimensional data space spanned by hyperspectral data sets is usually empty [19], indicating that the data structure involved exists primarily in a subspace. A commonly used approach to reduce the dimensionality of the data is the principal component transform (PCT) [23]. However, this approach is characterized by its global nature and cannot preserve subtle spectral differences required to obtain a good discrimination of classes [9]. Further, this approach relies on spectral properties of the data alone, thus neglecting the information related to the spatial arrangement of the pixels in the scene. As a result, there is a need for feature extraction techniques able to integrate the spatial and

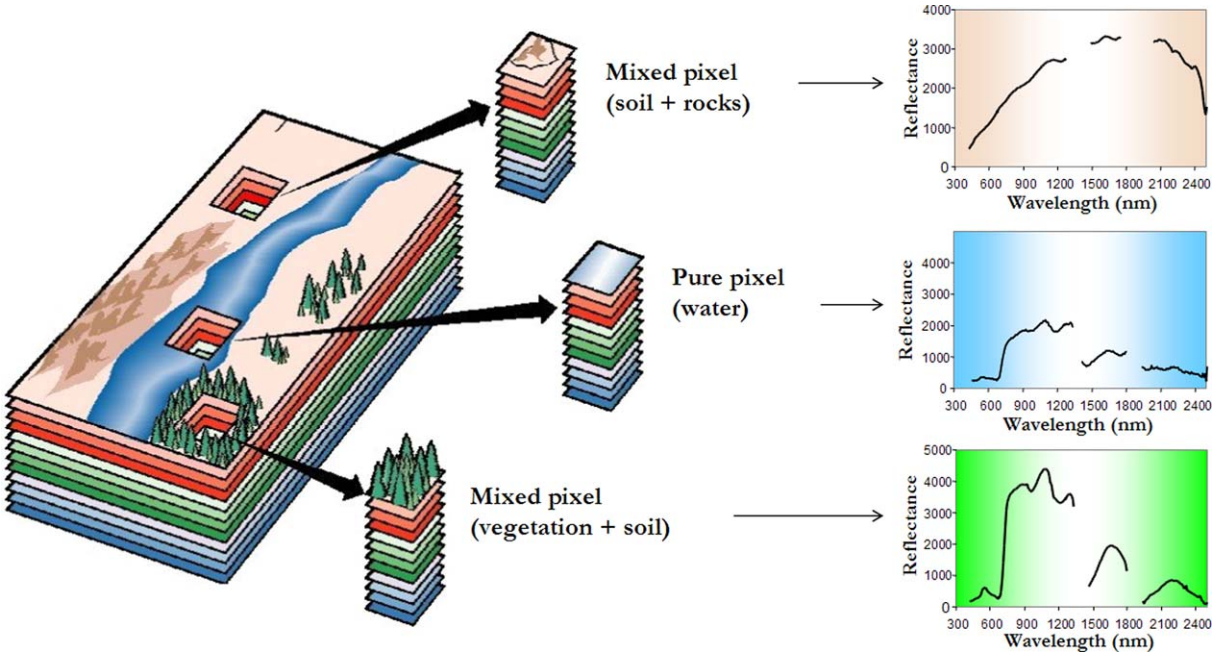

Fig. 1 Concept of hyperspectral imaging using NASA Jet Propulsion Laboratory's AVIRIS system

spectral information available from the data simultaneously [23].

While such integrated spatial/spectral developments hold great promise in the field of remote sensing data analysis, they introduce new processing challenges [24, 34]. The concept of Beowulf cluster was developed, in part, to address such challenges [7, 30]. The goal was to create parallel computing systems from commodity components to satisfy specific requirements for the Earth and space sciences community. Although most dedicated parallel machines employed by NASA and other institutions during the last decade have been chiefly homogeneous in nature, a current trend is to utilize heterogeneous and distributed parallel computing platforms [21]. In particular, computing on heterogeneous networks of computers (HNOCs) is an economical alternative which can benefit from local (user) computing resources while, at the same time, achieve high communication speed at lower prices. The properties above have led HNOCs to become a standard tool for high-performance computing in many ongoing and planned remote sensing missions [1, 21].

To address the need for cost-effective and innovative algorithms in this emerging new area, this paper develops a new parallel algorithm for classification of hyperspectral imagery. The algorithm is inspired by previous work on morphological neural networks, such as autoassociative morphological memories and morphological perceptrons [27], although it is based on different concepts. Most importantly, it can be tuned for very efficient execution on both HNOCs and massively parallel, Beowulf-type commodity clusters.

The remainder of the paper is structured as follows. Section 2 provides an overview of related work in the areas of cluster computing applied to remote sensing problems, heterogeneous computing practices, and parallel implementations of morphological and neural network-oriented algorithms in the same context. Section 3 describes the heterogeneous parallel algorithm, which consists of two main processing steps: (1) parallel morphological feature extraction taking into account the spatial and spectral information, and (2) robust classification using a parallel multilayer neural network with back-propagation learning. Section 4 describes the algorithm's accuracy and parallel performance. Classification accuracy is first discussed in the context of two real applications which respectively make use of hyperspectral data collected by NASA and the German Aerospace Center (DLR) to assess agricultural fields in the valley of Salinas, California, and the status of urban areas in Pavia, Italy. Parallel performance in the context of the above-mentioned applications is then assessed by comparing the efficiency achieved by an heterogeneous parallel version of the proposed algorithm, executed on a fully heterogeneous network, with the efficiency achieved by its equivalent homogeneous version, executed on a fully homogeneous network with the same aggregate performance as the heterogeneous one. For comparative purposes, performance data on Thunderhead, a massively parallel Beowulf cluster at NASA's Goddard Space Flight Center, are also given. Finally, Sect. 5 concludes with some remarks and hints at future research.

2 Related work

This section provides an overview of the evolution of cluster computing architectures in the context of remote sensing applications. The section begins with an introduction the initial developments in Beowulf systems at NASA centers going through more recent computing systems which are being employed for remote sensing data processing, including heterogeneous platforms. The section concludes with an overview of previous research in the development of parallel morphological feature extraction and neural network implementations in remote sensing applications.

2.1 Cluster computing in remote sensing

Beowulf clusters were originally developed with the purpose of creating a cost-effective parallel computing system able to satisfy specific computational requirements in the Earth and space sciences community. Initially, the need for large amounts of computation was identified for processing multispectral imagery with only a few bands [24]. As sensor instruments incorporated hyperspectral capabilities, it was soon recognized that computer mainframes and minicomputers could not provide sufficient power for processing this kind of data. The Linux operating system introduced the potential of being quite reliable due to the large number of developers and users. Later it became apparent that large numbers of developers could also be a disadvantage as well as an advantage.

In 1994, a team was put together at NASA's Goddard Space Flight Center (GSFC) to build a cluster consisting only of commodity hardware (PCs) running Linux, which resulted in the first Beowulf cluster [25]. It consisted of 16 100 MHz 486DX4-based PCs connected with 2 hub-based Ethernet networks tied together with channel bonding software so that the 2 networks acted like one network running at twice the speed. The next year Beowulf-II, a 16- PC cluster based on 100 MHz Pentium PCs, was built and performed about 3 times faster, but also demonstrated a much higher reliability. In 1996, a Pentium-Pro cluster at Caltech demonstrated a sustained Gigaflop on a remote sensing-based application. This was the first time a commodity cluster had shown high performance potential.

Up until 1997, Beowulf clusters were in essence engineering prototypes, that is, they were built by those who were going to use them. However, in 1997 a project was started at GSFC to build a commodity cluster that was intended to be used by those who had not built it, the HIVE (highly parallel virtual environment) project. The idea was to have workstations distributed among different locations and a large number of compute nodes (the compute core) concentrated in one area. The workstations would share the compute core as though it was apart of each. Although the original HIVE only had one workstation, many users were able to access it from their own workstations over the Internet. The HIVE was also the first commodity cluster to exceed a sustained 10 Gigaflop on a remote sensing algorithm. Currently, an evolution of the HIVE called Thunderhead is being used at GSFC for remote sensing data processing calculations (see http://thunderhead.gsfc.nasa.gov for additional details). In addition, NASA is currently supporting additional massively parallel clusters for remote sensing applications, such as the Columbia supercomputer at NASA Ames Research Center, a 10,240-CPU SGI Altix supercluster, with Intel Itanium 2 processors, 20 terabytes total memory and heterogeneous interconnects including InfiniBand network and 10 gigabit Ethernet. This system is listed as #8 in the November 2006 version of the Top500 list of supercomputer sites available online at http://www.top500.org. Among many other examples of HPC systems included in the list which are currently being exploited for remote sensing and Earth science-based applications, we cite three relevant systems for illustrative purposes:

- *MareNostrum*, an IBM cluster with 10,240 processors, 2.3 GHz Myrinet connectivity and 20,480 GB of main memory available at Barcelona Supercomputing Center (#5 in Top500).
- *Jaws*, a Dell PowerEdge cluster with 3 GHz Infiniband connectivity, 5,200 GB of main memory and 5,200 processors available at Maui High-Performance Computing Center (MHPCC) in Hawaii (#11 in Top500).
- *Earth Simulator*, a 5,120-processor system developed by NEC for Japan's Aerospace Exploration Agency and the Agency for Marine-Earth Science and Technology (#14 in Top500).

In addition to existing platforms, it is highly anticipated that many new supercomputer systems will be specifically developed in forthcoming years to support remote sensing applications. However, the current trend followed by NASA and many other agencies with remote sensing competences is to resort to networks of distributed and highly heterogeneous computing resources as a low-cost alternative to expensive supercomputers which fully complies with the distributed nature of ever-growing remote sensing data repositories [7].

2.2 Heterogeneous parallel computing

With the commercial availability of networking hardware, it soon became obvious that networked groups of machines distributed among different locations could be used together by one single parallel remote sensing code as a distributed-memory machine [32]. Of course, such networks were originally designed and built to connect heterogeneous sets of machines. As a result, heterogeneous networks have soon

become a very popular tool for distributed computations with essentially unbounded sets of machines, in which the number and location of machines may not be explicitly known [21] as opposed to cluster computing, in which the number and location of nodes are known and relatively fixed. An evolution of the concept of distributed computing described above resulted in the current notion of grid computing [12], in which the number and location of nodes is relatively dynamic and have to be discovered at run-time. It should be noted that the work described in this paper specifically focuses on distributed computing environments without meta-computing or grid computing, which aims at providing users access to services distributed over wide-area networks.

There are currently several ongoing research efforts aimed at efficient distributed processing of remote sensing data. Perhaps the most simple example is the use of heterogeneous versions of data processing algorithms developed for homogeneous networks [22], for instance, by resorting to heterogeneous-aware variations of homogeneous algorithms, able to capture the inherent heterogeneity of a network and to load-balance the computation among the available resources [35]. This framework allows one to easily port an existing parallel code developed for a homogeneous system to a fully heterogeneous environment, as will be shown in the following subsection. A recent example of this strategy in the context of remote sensing is the utilization of the Common Component Architecture (CCA) [20] as a plug-and-play environment for the construction of climate, weather, and ocean applications through a set of software components that conform to standardized interfaces. Such components encapsulate much of the complexity of the data processing algorithms inside a black box and expose only well-defined interfaces to other components. Among several other available efforts, another distributed application framework specifically developed for Earth science data processing is the Java Distributed Application Framework (JDAF) [33]. Although the two main goals of JDAF are flexibility and performance, the authors believe that the Java programming language is not yet mature enough for high performance computing of large amounts of data.

2.3 Parallel morphological/neural implementations in remote sensing

Parallelization of low-level image processing algorithms, including morphological operations, has been the subject of several research studies [15, 28]. However, the development of parallel approaches dealing with multi-channel imagery in the context of remote sensing applications is limited to only a few studies in the area of cluster-based computing [16, 24] and distributed processing [32]. In the case of distributed applications, the general approach for the design of parallel implementations is based on the development of effective heuristics for scheduling iterative task computations on the distributed platform [35].

Quite opposite, the development of parallel neural network-oriented methods in general remote sensing applications has been quite popular. Most available approaches rely on statistical assumptions [10], such as the use of a linear mixture model to interpret mixed pixels. However, these assumptions generally provide inaccuracies since efficiently trained neural networks can accurately approximate the nonlinear nature of mixed pixels in hyperspectral imaging [14]. For instance, it is very likely that the *vegetation* pixel in Fig. 1 is actually composed of a mixture of vegetation and soil, depending on the spatial resolution of the sensor. In this case, supervised spectral unmixing approaches via neural networks can be very effective in successfully modeling the inherent mixed nature of remotely sensed image pixels [3–5, 8]. In particular, it has been found in previous work that supervised architectures such as the multi-layer perceptron (MLP) are strongly dependent on the quality of input training patterns, in the sense that the use of highly representative (or discriminative) training patterns may have a more significant impact on the final classification results than the network configuration parameters used during the process (e.g., number of input, hidden and output neurons) [11, 25, 26].

Since neural networks are inherently amenable for parallel implementation, neurocomputing represents a suitable approach to deal with remote sensing imagery. Previous research in this area has addressed the impact of neural network partitioning and mapping onto specific parallel architectures such as systolic arrays [18, 36] and clusters of computers [2, 17]. However, very little effort has been devoted to the development of specific parallel implementations of neural network architectures for hyperspectral image processing. In the following section, we describe a new parallel morphological/neural supervised classification technique, which represents a unique and novel approach for supervised classification of hyperspectral images taking into account both the spectral and the spatial information contained in the data. This represents a novel contribution in the field of hyperspectral image processing.

3 Parallel algorithm

This section describes a new parallel algorithm for analysis of remotely sensed hyperspectral images. Before describing the two main steps of the algorithm, we first formulate a general optimization problem in the context of HNOCs, composed of different-speed processors that communicate through links at different capacities [21]. This type of platform can be modeled as a complete graph $G = (P, E)$ where

each node models a computing resource p_i weighted by its relative cycle-time w_i . Each edge in the graph models a communication link weighted by its relative capacity, where c_{ij} denotes the maximum capacity of the slowest link in the path of physical communication links from p_i to p_j. We also assume that the system has symmetric costs, i.e., $c_{ij} = c_{ji}$. Under the above assumptions, processor p_i will accomplish a share of $\alpha_i \times W$ of the total workload W, with $\alpha_i \geq 0$ for $1 \leq i \leq P$ and $\sum_{i=1}^{P} \alpha_i = 1$. With the above assumptions in mind, an abstract view of our problem can be simply stated in the form of a client-server architecture, in which the server is responsible for the efficient distribution of work among the P nodes, and the clients operate with the spatial and spectral information contained in a local partition. The partitions are then updated locally and the resulting calculations may also be exchanged between the clients, or between the server and the clients. Below, we describe the two steps of our parallel algorithm.

3.1 Parallel morphological algorithm

This section develops a parallel morphological feature extraction algorithm for hyperspectral image analysis. First, we briefly introduce the concept of spectral matching. Then, we describe the morphological algorithm and its parallel implementation for HNOCs.

3.1.1 Spectral matching in hyperspectral imaging

Let us first denote by f a hyperspectral image defined on an N-dimensional (N-D) space, where N is the number of channels or spectral bands in the image. A widely used technique to measure the similarity between spectral signatures in the input data is spectral matching [6], which evaluates the amount of correlation between the signatures. For instance, a widely used distance in hyperspectral analysis is the spectral angle mapper (SAM), which can be used to measure the spectral similarity between two pixels, $f(x, y)$ and $f(i, j)$, i.e., two N-D vectors at discrete spatial coordinates (x, y) and $(i, j) \in Z^2$, as follows:

$$\mathrm{SAM}(f(x, y), f(i, j)) = \cos^{-1} \frac{f(x, y) \cdot f(i, j)}{\|f(x, y)\| \cdot \|f(i, j)\|} \quad (1)$$

3.1.2 Morphological feature extraction algorithm

The proposed feature extraction method is based on mathematical morphology [29] and spectral matching concepts. The goal is to impose an ordering relation (in terms of spectral purity) in the set of pixel vectors lying within a spatial search window (called structuring element) designed by B [23]. This is done by defining a cumulative distance between a pixel vector $f(x, y)$ and all the pixel vectors in the spatial neighborhood given by B (B-neighborhood) as follows: $D_B[f(x, y)] = \sum_i \sum_j \mathrm{SAM}[f(x, y), f(i, j)]$, where (x, y) refers to spatial coordinates in the B-neighborhood. From the above definitions, two standard morphological operations called erosion and dilation can be respectively defined as follows:

$$(f \otimes B)(x, y) = \underset{(s,t) \in Z^2(B)}{\mathrm{argmin}} \sum_s \sum_t \mathrm{SAM}(f(x, y), f(x + s, y + t)) \quad (2)$$

$$(f \oplus B)(x, y) = \underset{(s,t) \in Z^2(B)}{\mathrm{argmax}} \sum_s \sum_t \mathrm{SAM}(f(x, y), f(x - s, y - t)) \quad (3)$$

Using the above operations, the opening filter is defined as $(f \circ B)(x, y) = [(f \otimes C) \oplus B](x, y)$ (erosion followed by dilation), while the closing filter is defined as $(f \bullet B)(x, y) = [(f \oplus C) \otimes B](x, y)$ (dilation followed by erosion). The composition of opening and closing operations is called a spatial/spectral profile, which is defined as a vector which stores the relative spectral variation for every step of an increasing series. Let us denote by $\{(f \circ B)^{\lambda}(x, y)\}, \lambda = \{0, 1, \ldots, k\}$, the *opening series* at $f(x, y)$, meaning that several consecutive opening filters are applied using the same window B. Similarly, let us denote by $\{(f \bullet B)^{\lambda}(x, y)\}, \lambda = \{0, 1, \ldots, k\}$, the *closing series* at $f(x, y)$. Then, the spatial/spectral profile at $f(x, y)$ is given by the following vector:

$$p(x, y) = \{\mathrm{SAM}((f \circ B)^{\lambda}(x, y), (f \circ B)^{\lambda-1}(x, y))\} \cup \{\mathrm{SAM}((f \bullet B)^{\lambda}(x, y), (f \bullet B)^{\lambda-1}(x, y))\} \quad (4)$$

Here, the step of the opening/closing series iteration at which the spatial/spectral profile provides a maximum value gives an intuitive idea of both the spectral and spatial distribution in the B-neighborhood [23]. As a result, the profile can be used as a feature vector on which the classification is performed using a spatial/spectral criterion.

3.1.3 Parallel implementation

Two types of partitioning can be exploited in the parallelization of spatial/spectral algorithms such as the one addressed above [24]. Spectral-domain partitioning subdivides the volume into small cells or sub-volumes made up of contiguous spectral bands, and assigns one or more sub-volumes to each processor. With this model, each pixel vector is split amongst several processors, which breaks the spectral identity of the data because the calculations for each pixel vector (e.g., for the SAM calculation) need to originate from several different processing units. On the other

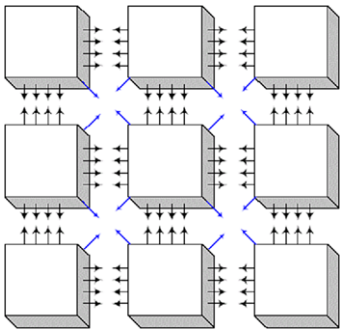

Fig. 2 Communication framework for the morphological feature extraction algorithm

hand, spatial-domain partitioning provides data chunks in which the same pixel vector is never partitioned among several processors. In this work, we adopt a spatial-domain partitioning approach due to several reasons. First, the application of spatial-domain partitioning is a natural approach for morphological image processing, as many operations require the same function to be applied to a small set of elements around each data element present in the image data structure, as indicated in the previous subsection. A second reason has to do with the cost of inter-processor communication. In spectral-domain partitioning, the window-based calculations made for each hyperspectral pixel need to originate from several processing elements, in particular, when such elements are located at the border of the local data partitions (see Fig. 2), thus requiring intensive inter-processor communication.

However, if redundant information such as an overlap border is added to each of the adjacent partitions to avoid accesses outside the image domain, then boundary data to be communicated between neighboring processors can be greatly minimized. Such an overlapping scatter would obviously introduce redundant computations, since the intersection between partitions would be non-empty. Our implementation makes use of a constant structuring element B (with size of 3×3 pixels) which is repeatedly iterated to increase the spatial context, and the total amount of redundant information is minimized. To do so, we have implemented a special 'overlapping scatter' operation that also sends out the overlap border data as part of the scatter operation itself (i.e., redundant computations replace communications). Here, we make use of MPI *derived datatypes* to directly scatter hyperspectral data structures, which may be stored non-contiguously in memory, in a single communication step. A comparison between the associative costs of redundant computations in overlap with the overlapping scatter approach, versus the communications costs of accessing neighboring cell elements outside of the image domain has been presented and discussed in previous work [24].

A pseudo-code of the proposed parallel algorithm, specifically tuned for HNOCs, is given below.

HeteroMORPH algorithm:

Inputs: N-dimensional cube f, Structuring element B.
Output: Set of morphological profiles for each pixel.

1. Obtain information about the heterogeneous system, including the number of processors, P, each processor's identification number, $\{p_i\}_{i=1}^{P}$, and processor cycle-times, $\{w_i\}_{i=1}^{P}$.
2. Using B and the information obtained in step 1, determine the total volume of information, R, that needs to be replicated from the original data volume, V, according to the data communication strategies outlined above, and let the total workload W to be handled by the algorithm be given by $W = V + R$.
3. Set $\alpha_i = \lfloor \frac{(P/w_i)}{\sum_{i=1}^{P}(1/w_i)} \rfloor$ for all $i \in \{1, \ldots, P\}$.
4. For $m = \sum_{i=1}^{P} \alpha_i$ to $(V + R)$, find $k \in \{1, \ldots, P\}$ so that $w_k \cdot (\alpha_k + 1) = \min\{w_i \cdot (\alpha_i + 1)\}_{i=1}^{P}$ and set $\alpha_k = \alpha_k + 1$.
5. Use the resulting $\{\alpha_i\}_{i=1}^{P}$ to obtain a set of P spatial-domain heterogeneous partitions (with overlap borders) of W, and send each partition to processor p_i, along with B.
6. Calculate the morphological profiles $p(x, y)$ for the pixels in the local data partitions (in parallel) at each heterogeneous processor.
7. Collect all the individual results and merge them together to produce the final output.

A homogeneous version of the HeteroMORPH algorithm above can be simply obtained by replacing step 4 with $\alpha_i = P/w_i$ for all $i \in \{1, \ldots, P\}$, where w_i is the communication speed between processor pairs in the network, which is assumed to be homogeneous.

3.2 Parallel neural algorithm

In this section, we describe a supervised parallel classifier based on a multi-layer perceptron (MLP) neural network with back-propagation learning. This approach has been shown in previous work to be very robust for classification of hyperspectral imagery [26]. However, the considered neural architecture and back-propagation-type learning algorithm introduce additional considerations for parallel implementation on HNOCs.

3.2.1 Network architecture and learning

The architecture adopted for the proposed MLP-based neural network classifier is shown in Fig. 3. As shown by the figure, the number of input neurons equals the number of spectral bands acquired by the sensor. In the case of PCT-based pre-processing or morphological feature extraction commonly adopted in hyperspectral analysis, the number of neurons at the input layer equals the dimensionality

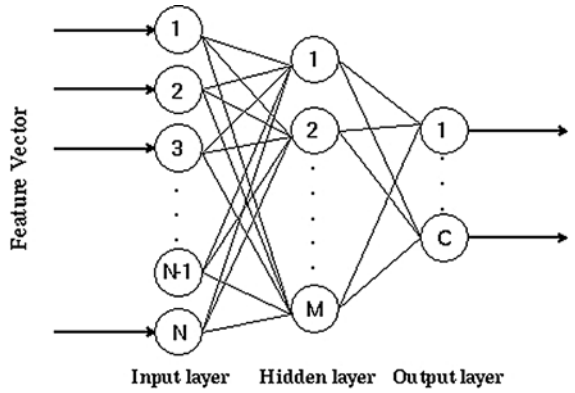

Fig. 3 MLP neural network topology

of feature vectors used for classification. The second layer is the hidden layer, where the number of nodes, M, is usually estimated empirically. Finally, the number of neurons at the output layer, C, equals the number of distinct classes to be identified in the input data. With the above architecture in mind, the standard back-propagation learning algorithm can be outlined by the following steps:

1. *Forward phase*. Let the individual components of an input pattern be denoted by $f_j(x, y)$, with $j = 1, 2, \ldots, N$. The output of the neurons at the hidden layer are obtained as: $H_i = \varphi(\sum_{j=1}^{N} \omega_{ij} \cdot f_j(x, y))$ with $i = 1, 2, \ldots, M$, where $\varphi(\cdot)$ is the activation function and ω_{ij} is the weight associated to the connection between the i-th input node and the j-th hidden node. The outputs of the MLP are obtained using $O_k = \varphi(\sum_{i=1}^{M} \omega_{ki} \cdot H_i)$, with $k = 1, 2, \ldots, C$. Here, ω_{ki} is the weight associated to the connection between the i-th hidden node and the k-th output node.
2. *Error back-propagation*. In this stage, the differences between the desired and obtained network outputs are calculated and back-propagated. The *delta* terms for every node in the output layer are calculated using $\delta_k^o = (O_k - d_k) \cdot \varphi'(\cdot)$, with $i = 1, 2, \ldots, C$. Here, $\varphi'(\cdot)$ is the first derivative of the activation function. Similarly, *delta* terms for the hidden nodes are obtained using $\delta_i^h = \sum_{k=1}^{C} (\omega_{ki} \cdot \delta_i^o) \cdot \varphi'(\cdot)$, with $i = 1, 2, \ldots, M$.
3. *Weight update*. After the back-propagation step, all the weights of the network need to be updated according to the *delta* terms and to η, a learning rate parameter. This is done using $\omega_{ij} = \omega_{ij} + \eta \cdot \delta_i^h \cdot f_j(x, y)$ and $\omega_{ki} = \omega_{ki} + \eta \cdot \delta_k^o \cdot H_i$. Once this stage is accomplished, another training pattern is presented to the network and the procedure is repeated for all incoming training patterns.

Once the back-propagation learning algorithm is finalized, a classification stage follows, in which each input pixel vector is classified using the weights obtained by the network during the training stage [26].

3.2.2 *Parallel multi-layer perceptron classifier*

The parallel classifier presented in this work is based on a hybrid partitioning scheme, in which the hidden layer is partitioned using neuronal level parallelism and weight connections are partitioned on the basis of synaptic level parallelism [31]. As a result, the input and output neurons are common to all processors, while the hidden layer is partitioned so that each heterogeneous processor receives a number of hidden neurons which depends on its relative speed. Each processor stores the weight connections between the neurons local to the processor. Since the fully connected MLP network is partitioned into P partitions and then mapped onto P heterogeneous processors using the above framework, each processor is required to communicate with every other processor to simulate the complete network. For this purpose, each of the processors in the network executes the three phases of the back-propagation learning algorithm described above.

HeteroNEURAL algorithm:

Inputs: N-dimensional cube f, Training patterns $f_j(x, y)$.
Output: Set of classification labels for each image pixel.

1. Use steps 1–4 of HeteroMORPH algorithm to obtain a set of values $(\alpha_i)_{i=1}^{P}$ which will determine the share of the workload to be accomplished by each heterogeneous processor.
2. Use the resulting $(\alpha_i)_{i=1}^{P}$ to obtain a set of P heterogeneous partitions of the hidden layer and map the resulting partitions among the P heterogeneous processors (which also store the full input and output layers along with all connections involving local neurons).
3. *Parallel training.* For each considered training pattern, the following three parallel steps are executed:
   (a) *Parallel forward phase.* In this phase, the activation value of the hidden neurons local to the processors are calculated. For each input pattern, the activation value for the hidden neurons is calculated using $H_i^P = \varphi(\sum_{j=1}^{N} \omega_{ij} \cdot f_j(x, y))$. Here, the activation values and weight connections of neurons present in other processors are required to calculate the activation values of output neurons according to $O_k^P = \varphi(\sum_{i=1}^{M/P} \omega_{ki}^P \cdot H_i^P)$, with $k = 1, 2, \ldots, C$. In our implementation, broadcasting the weights and activation values is circumvented by calculating the partial sum of the activation values of the output neurons.
   (b) *Parallel error back-propagation.* In this phase, each processor calculates the error terms for the local hidden neurons. To do so, *delta* terms for the output neurons are first calculated using $(\delta_k^o)^P = (O_k - d_k)^P \cdot \varphi'(\cdot)$, with $i = 1, 2, \ldots, C$. Then, error terms for the hidden layer are computed using $(\delta_i^h)^P = \sum_{k=1}^{P} (\omega_{ki}^P \cdot (\delta_k^o)^P) \cdot \varphi'(\cdot)$, with $i = 1, 2, \ldots, M/P$.

(c) *Parallel weight update.* In this phase, the weight connections between the input and hidden layers are updated by $\omega_{ij} = \omega_{ij} + \eta^P \cdot (\delta_i^h)^P \cdot f_j(x, y)$. Similarly, the weight connections between the hidden and output layers are updated using the expression: $\omega_{ki}^P = \omega_{ki}^P + \eta^P \cdot (\delta_k^o)^P \cdot H_i^P$.

4. *Classification.* For each pixel vector in the input data cube f, calculate (in parallel) $\sum_{j=1}^P O_k^j$, with $k = 1, 2, \ldots, C$. A classification label for each pixel can be obtained using the winner-take-all criterion commonly used in neural networks by finding the cumulative sum with maximum value, say $\sum_{j=1}^P O_{k^*}^j$, with $k^* = \arg\{\max_{1 \le k \le C} \sum_{j=1}^P O_k^j\}$.

4 Experimental results

This section provides an assessment of the effectiveness of the parallel algorithms described in Sect. 2. The section is organized as follows. First, we describe a framework for assessment of heterogeneous algorithms and provide an overview of the heterogeneous and homogeneous networks used in this work for evaluation purposes. Second, we briefly describe the hyperspectral data set used in experiments. Performance data are given in the last subsection.

4.1 Network description

Following a recent study [22], we assess the proposed heterogeneous algorithms using the basic postulate that they cannot be executed on a heterogeneous network faster than its homogeneous prototype on the equivalent homogeneous cluster network. Let us assume that a heterogeneous network consists of $\{p_i\}_i^P$ heterogeneous workstations with different cycle-times w_i, which span m communication segments $\{s_j\}_{j=1}^m$, where $c^{(j)}$ denotes the communication speed of segment s_j. Similarly, let $p^{(j)}$ be the number of processors that belong to s_j, and let $w_t^{(j)}$ be the speed of the t-th processor connected to s_j, where $t = 1, \ldots, p^{(j)}$. Finally, let $c^{(j,k)}$ be the speed of the communication link between segments s_j and s_k, with $j, k = 1, \ldots, m$. According to [22], the above network can be considered equivalent to a homogeneous one made up of $\{q_i\}_{i=1}^P$ processors with constant cycle-time and interconnected through a homogeneous communication network with speed c if, and only if the following expressions are satisfied:

$$c = \frac{\sum_{j=1}^m c^{(j)} \cdot [\frac{p^{(j)}(p^{(j)}-1)}{2}]}{\frac{P(P-1)}{2}} + \cdots + \frac{\sum_{j=1}^m \sum_{k=j+1}^m p^{(j)} \cdot p^{(k)} \cdot c^{(j,k)}}{\frac{P(P-1)}{2}} \quad (5)$$

and

$$w = \frac{\sum_{j=1}^m \sum_{t=1}^{p^{(j)}} w_t^{(j)}}{P} \quad (6)$$

where the first expression states that the average speed of point-to-point communications between processors $\{p_i\}_{i=1}^P$ in the heterogeneous network should be equal to the speed of point-to-point communications between processors $\{q_i\}_{i=1}^P$ in the homogeneous network, with both networks having the same number of processors. On the other hand, the second expression simply states that the aggregate performance of processors $\{p_i\}_{i=1}^P$ should be equal to the aggregate performance of processors $\{q_i\}_{i=1}^P$.

We have configured two networks of workstations to serve as sample networks for testing the performance of the proposed heterogeneous hyperspectral imaging algorithm. The networks are considered approximately equivalent under the above framework. Their description follows:

- *Fully heterogeneous network.* Consists of 16 different workstations, and four communication segments. Table 1 shows the properties of the 16 heterogeneous workstations, where processors $\{p_i\}_{i=1}^4$ are attached to communication segment s_1, processors $\{p_i\}_{i=5}^8$ communicate through s_2, processors $\{p_i\}_{i=9}^{10}$ are interconnected via s_3, and processors $\{p_i\}_{i=11}^{16}$ share the communication segment s_4. The communication links between the different segments $\{s_j\}_{j=1}^4$ only support serial communication. For illustrative purposes, Table 2 also shows the capacity of all point-to-point communications in the heterogeneous network, expressed as the time in milliseconds to transfer a one-megabit message between each processor pair (p_i, p_j) in the heterogeneous system. As noted, the communication network of the fully heterogeneous network consists of four relatively fast homogeneous communication segments, interconnected by three slower communication links with capacities $c^{(1,2)} = 29.05$, $c^{(2,3)} = 48.31$, $c^{(3,4)} = 58.14$ in milliseconds, respectively. Although this is a simple architecture, it is also a quite typical and realistic one as well.
- *Fully homogeneous network.* Consists of 16 identical Linux workstations $\{q_i\}_{i=1}^{16}$ with processor cycle-time of $w = 0.0131$ seconds per megaflop, interconnected via a homogeneous communication network where the capacity of links is $c = 26.64$ milliseconds.

Finally, in order to test the proposed algorithm on a large-scale parallel platform, we have also experimented with Thunderhead, a massively parallel Beowulf cluster at NASA's Goddard Space Flight Center (see Fig. 4). The system is composed of 256 dual 2.4 GHz Intel Xeon nodes, each with 1 GB of memory and 80 GB of main memory. The total peak performance of the system is 2457.6 GFlops. Along with the 512-processor computer core, Thunderhead

Table 1 Specifications of heterogeneous processors

Processor	Architecture	Cycle-time (secs/megaflop)	Main memory (MB)	Cache (KB)
p_1	Free BSD—i386 Intel Pentium 4	0.0058	2048	1024
p_2, p_5, p_8	Linux—Intel Xeon	0.0102	1024	512
p_3	Linux—AMD Athlon	0.0026	7748	512
p_4, p_6, p_7, p_9	Linux—Intel Xeon	0.0072	1024	1024
p_{10}	SunOS—SUNW UltraSparc-5	0.0451	512	2048
$p_{11}-p_{16}$	Linux—AMD Athlon	0.0131	2048	1024

Table 2 Capacity of communication links (time in milliseconds to transfer a one-megabit message)

Processor	p_1-p_4	p_5-p_8	p_9-p_{10}	$p_{11}-p_{16}$
p_1-p_4	19.26	48.31	96.62	154.76
p_5-p_8	48.31	17.65	48.31	106.45
p_9-p_{10}	96.62	48.31	16.38	58.14
$p_{11}-p_{16}$	154.76	106.45	58.14	14.05

Fig. 4 Thunderhead Beowulf cluster at NASA's Goddard Space Flight Center

has several nodes attached to the core with 2 GHz optical fibre Myrinet. In all considered platforms, the operating system used at the time of experiments was Linux Fedora Core, and MPICH was the message-passing library used (see http://www-unix.mcs.anl.gov/mpi/mpich).

4.2 Hyperspectral data

Before empirically investigating the performance of the proposed parallel hyperspectral imaging algorithms in the five considered platforms, we first describe two hyperspectral image scenes that will be used in experiments.

1. ***AVIRIS data***. The first scene was collected by the 224-band AVIRIS sensor over Salinas Valley, California, and is characterized by high spatial resolution (3.7-meter pixels). The relatively large area covered (512 lines by 217 samples) results in a total image size of more than 1 GB. Figure 5a shows the spectral band at 587 nm wavelength and a sub-scene (called hereinafter Salinas A), which comprises 83×86 pixels and is dominated by directional features. Figure 5b shows the ground-truth map, in the form of a class assignment for each labeled pixel with 15 mutually exclusive ground-truth classes. As shown by Fig. 5b, ground truth is available for nearly half of Salinas scene. The data set above represents a very challenging classification problem (due to the spectral similarity of most classes, discriminating among them is very difficult). This fact has made the scene a universal and widely used benchmark to validate classification accuracy of hyperspectral algorithms [23].
2. ***DAIS 7915 data***. The second hyperspectral data set used in experiments was collected by the DAIS 7915 airborne imaging spectrometer of the German Aerospace Agency (DLR). It was collected by a flight at 1500 meters altitude over the city of Pavia, Italy. The resulting scene has spatial resolution of 5 meters and size of 400 lines by 400 samples. Figure 6a shows the image collected at 639 nm by DAIS 7915 imaging spectrometer, which reveals a dense residential area on one side of the river, as well as open areas and meadows on the other side. Ground-truth is available for several areas of the scene (see Fig. 6b). Following a previous research study on this scene [23], we take into account only 40 spectral bands of reflective energy, and thus skip thermal infrared and middle infrared bands above 1958 nm because of low signal to noise ratio in those bands.

In order to test the accuracy of the proposed parallel morphological/neural classifier, a random sample of less than

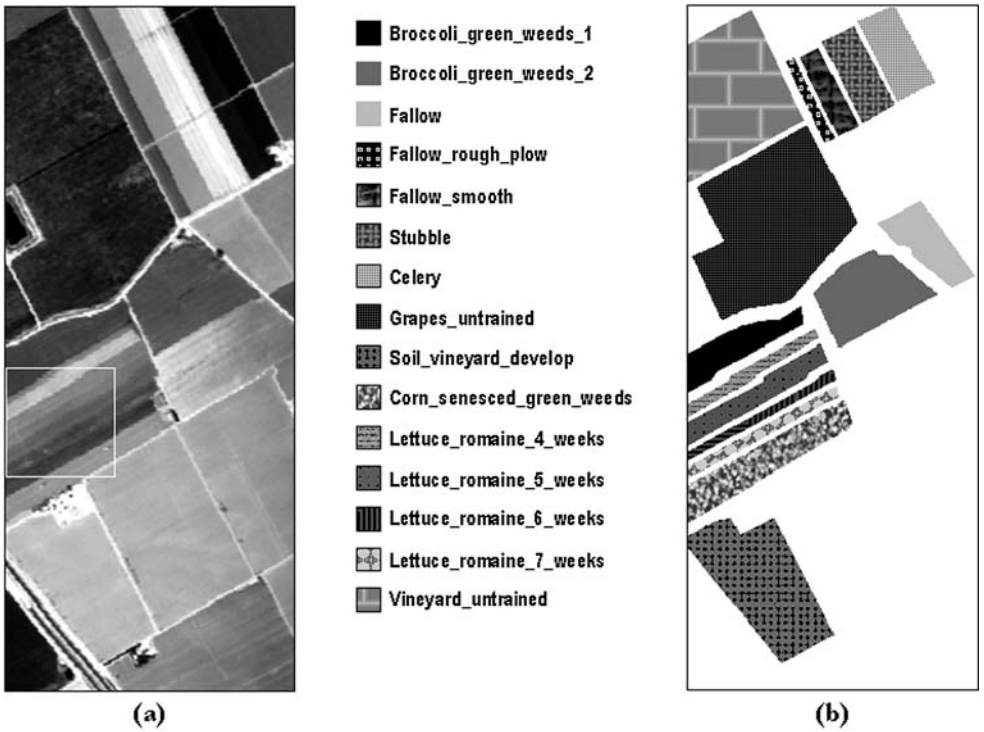

Fig. 5 (**a**) AVIRIS scene of Salinas Valley, California, and (**b**) Land-cover ground classes

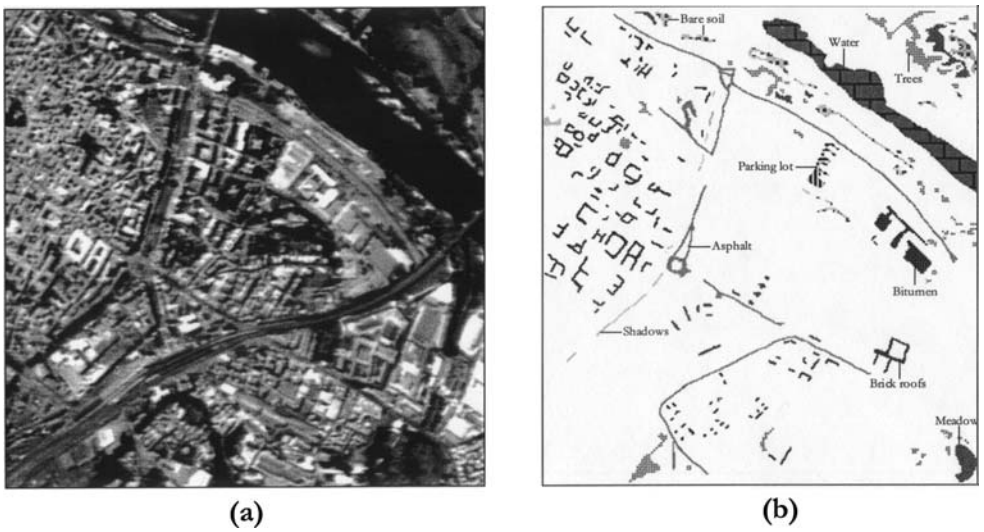

Fig. 6 (**a**) DAIS 7915 scene of Pavia city, Italy, and (**b**) Land-cover ground classes

2% of the pixels was chosen from the known ground truth of the two images described above. Morphological profiles were then constructed in parallel for the selected training samples using 10 iterations, which resulted in feature vectors with dimensionality of 20 (i.e., 10 structuring element iterations for the *opening series* and 10 iterations for the *closing series*). The resulting features were then used to train the parallel back-propagation neural network classifier with one hidden layer, where the number of hidden neurons was selected empirically as the square root of the product of the number of input features and information classes (several configurations of the hidden layer were tested and the one that gave the highest overall accuracies was reported). The trained classifier was then applied to the remaining 98% of labeled pixels in each considered scene, yielding the classification accuracies shown in Tables 3 and 4.

For comparative purposes, the accuracies obtained using the full spectral information and PCT-reduced features as input to the neural classifier are also reported in Tables 3 and 4. As shown by both tables, morphological input features substantially improve individual and overall classification accuracies with regards to PCT-based features and the full spectral information (e.g., for the directional "lettuce" classes contained in the Salinas A sub-

Table 3 Classification accuracies (in percentage) achieved by the parallel neural classifier for the AVIRIS Salinas scene using morphological features, PCT-based features and the original spectral information (processing times in a single Thunderhead node are given in the parentheses)

AVIRIS Salinas Class label	Spectral information (2981)	PCT-based features (3256)	Morphological features (3679)
Fallow rough plow	96.51	91.90	96.78
Fallow smooth	93.72	93.21	97.63
Stubble	94.71	95.43	98.96
Celery	89.34	94.28	98.03
Grapes untrained	88.02	86.38	95.34
Soil vineyard develop	88.55	84.21	90.45
Corn senesced green weeds	82.46	75.33	87.54
Lettuce romaine 4 weeks	78.86	76.34	83.21
Lettuce romaine 5 weeks	82.14	77.80	91.35
Lettuce romaine 6 weeks	84.53	78.03	88.56
Lettuce romaine 7 weeks	84.85	81.54	86.57
Vineyard untrained	87.14	84.63	92.93
Overall accuracy	87.25	86.21	95.08

Table 4 Classification accuracies (in percentage) achieved by the morphological/neural classifier for the DAIS 7915 Pavia scene using morphological features, PCT-based features and the original spectral information (processing times in a single Thunderhead node are given in the parentheses)

DAIS 7915 Pavia Class label	Spectral information (274)	PCT-based features (297)	Morphological features (335)
Water	87.30	86.17	100
Trees	94.64	97.62	98.72
Asphalt	97.79	84.48	98.88
Parking lot	83.82	81.93	71.77
Bitumen	86.11	75.48	98.68
Brick roofs	83.69	82.36	99.37
Meadow	88.88	89.86	92.61
Bare soil	79.85	84.68	95.11
Shadows	89.64	92.81	96.19
Overall accuracy	88.65	89.75	96.16

scene or the directional "brick roofs" class in the Pavia scene). This is not surprising since morphological operations use both spatial and spectral information as opposed to the other methods which rely on spectral information alone. For illustrative purposes, Tables 3 and 4 also include (in the parentheses) the algorithm processing times in seconds for the different approaches tested, measured on a single processor in the Thunderhead system. Experiments were performed using the GNU-C/C++ compiler in its 4.0 version. As shown by both tables, the computational cost was slightly higher when morphological feature extraction was used.

Finally, we can note from Tables 3 and 4 that the processing times measured for the DAIS 7915 scene in a single Thunderhead node were always about eleven times smaller than those obtained for AVIRIS scene. This is not surprising since the proportion in size is similar (137 MB for the AVIRIS scene versus 12.5 MB for the DAIS 7915 scene). Since the proposed parallel morphological/neural classification algorithm is dominated by regular computations, and given the satisfactory classification results reported in two completely different application scenarios, only parallel performance results for the larger (AVIRIS) scene will be provided in the following subsection for simplicity.

Table 5 Execution times (in seconds) and performance ratios reported for the homogeneous algorithms versus the heterogeneous ones on the two considered networks

Algorithm	Homogeneous network		Heterogeneous network	
	Time	Homo/Hetero	Time	Homo/Hetero
HeteroMORPH	221	1.11	206	10.98
HomoMORPH	198		2261	
HeteroCOM	289	1.12	242	11.86
HomoCOM	258		2871	
HeteroNEURAL	141	1.12	130	9.70
HomoNEURAL	125		1261	

4.3 Assessment of the parallel algorithm

To investigate the properties of the parallel morphological/neural classification algorithm developed in this work, the performance of its two main modules (HeteroMORPH and HeteroNEURAL) was first tested by timing the program using the heterogeneous network and its equivalent homogeneous one. For illustrative purposes, an alternative implementation of HeteroMORPH without 'overlapping scatter' was also tested, i.e., in this implementation the overlap border data are not replicated between adjacent processors but communicated instead. This approach is denoted as HeteroCOM, with its correspondent homogeneous version designated by HomoCOM.

As expected, the execution times reported on Table 5 for the three considered heterogeneous algorithms and their respective homogeneous versions indicate that the heterogeneous implementations were able to adapt much better to the heterogeneous computing environment than the homogeneous ones, which were only able to perform satisfactorily on the homogeneous network. For the sake of comparison, Table 5 also shows the performance ratio between the heterogeneous algorithms and their respective homogeneous versions (referred to as Homo/Hetero ratio in the table and simply calculated as the execution time of the homogeneous algorithm divided by the execution time of the heterogeneous algorithm).

From Table 5, one can also see that the heterogeneous algorithms were always several times faster than their homogeneous counterparts in the heterogeneous network, while the homogeneous algorithms only slightly outperformed their heterogeneous counterparts in the homogeneous network. The Homo/Hetero ratios reported in the table for the homogeneous algorithms executed on the homogeneous network were indeed very close to 1, a fact that reveals that the performance of heterogeneous algorithms was almost the same as that evidenced by homogeneous algorithms when they were run in the same homogeneous environment. The above result demonstrates the flexibility of the proposed heterogeneous algorithms, which were able to adapt efficiently to the two considered networks.

Interestingly, Table 5 also reveals that the performance of the heterogeneous algorithms on the heterogeneous network was almost the same as that evidenced by the equivalent homogeneous algorithms on the homogeneous network (i.e., the algorithms achieved essentially the same speed, but each on its network). This seems to indicate that the heterogeneous algorithms are very close to the optimal heterogeneous modification of the basic homogeneous ones. Finally, although the Homo/Hetero ratios achieved by HeteroMORPH and HeteroCOM are similar, the processing times in Table 5 seem to indicate that the data-replication strategy adopted by HeteroMORPH is more efficient than the data-communication strategy adopted by HeteroCOM in our considered application.

To further explore the above observations in more detail, an in-depth analysis of computation and communication times achieved by the different methods is also highly desirable. For that purpose, Table 6 shows the total time spent by the tested algorithms in communications (labeled as COM in the table) and computations in the two considered networks, where two types of computation times were analyzed, namely, sequential (those performed by the root node with no other parallel tasks active in the system, labeled as SEQ in the table) and parallel (the rest of computations, i.e., those performed by the root node and/or the workers in parallel, labeled as PAR in the table). The latter includes the times in which the workers remain idle. It is important to note that our parallel implementations have been carefully designed to allow overlapping of communications and computations when no data dependencies are involved.

It can be seen from Table 6 that the COM scores were very low when compared to PAR scores in both HeteroMORPH and HeteroNEURAL. This is mainly due to the fact that these algorithms involve only a few inter-processor communications, which led to almost complete overlapping between computation and communications in most cases. In the case of HeteroMORPH, it can be observed that SEQ and PAR scores are slightly increased with regards to those obtained for HeteroCOM as a result of the data replication

Table 6 Communication (COM), sequential computation (SEQ) and parallel computation (PAR) times for the homogeneous algorithms versus the heterogeneous ones on the two considered networks after processing the AVIRIS Salinas hyperspectral image

	Homogeneous network			Heterogeneous network		
	COM	SEQ	PAR	COM	SEQ	PAR
HeteroMORPH	7	19	202	11	16	190
HomoMORPH	14	18	180	6	16	2245
HeteroCOM	57	16	193	52	15	182
HomoCOM	64	15	171	69	13	2194
HeteroNEURAL	4	27	114	7	24	106
HomoNEURAL	9	27	98	3	24	1237

Table 7 Load-balancing rates for the parallel algorithms on the homogeneous and heterogeneous network

Algorithm	Homogeneous network		Heterogeneous network	
	D_{All}	D_{Minus}	D_{All}	D_{Minus}
HeteroMORPH	1.03	1.02	1.05	1.01
HomoMORPH	1.05	1.01	1.59	1.21
HeteroCOM	1.06	1.04	1.09	1.03
HomoCOM	1.07	1.03	1.94	1.52
HeteroNEURAL	1.02	1.01	1.03	1.01
HomoNEURAL	1.03	1.01	1.39	1.19

strategy introduced by the former algorithm. However, Table 6 also reveals that the COM scores measured for HeteroCOM were much higher than those reported for HeteroMORPH, and could not be completely overlapped with computations due to the high message traffic resulting from communication of full hyperspectral pixel vectors across the heterogeneous network. This is the main reason why the execution times measured for HeteroCOM where the highest in both networks, as already reported by Table 5. Finally, the fact that the PAR scores produced by the homogeneous algorithms executed on the heterogeneous network are so high is likely due to a less efficient workload distribution among the heterogeneous workers. Therefore, a study of load balance is highly required to fully substantiate the parallel properties of the considered algorithms.

In order to measure load balance, Table 7 shows the imbalance scores achieved by the parallel algorithms on the two considered networks. The imbalance is defined as $D = R_{max}/R_{min}$, where R_{max} and R_{min} are the maxima and minima processor run times, respectively. Therefore, perfect balance is achieved when $D = 1$. In the table, we display the imbalance considering all processors, D_{All}, and also considering all processors but the root, D_{Minus}. As we can see from Table 7, both HeteroMORPH and HeteroNEURAL algorithms were able to provide values of D_{All} close to 1 in the two considered networks, which indicates that the proposed heterogeneous data partitioning algorithm is effective. Further, the above algorithms provided almost the same results for both D_{All} and D_{Minus} while, for the homogeneous versions, load balance was much better when the root processor was not included. While the homogeneous algorithms executed on the heterogeneous network provided the highest values of D_{All} and D_{Minus} (and hence the highest imbalance), the heterogeneous algorithms executed on the homogeneous network resulted in values of D_{Minus} which were close to optimal.

Despite the fact that conventional feature extraction algorithms (such as those based on PCT) do not take into account the spatial information explicitly into the computations—a fact that has traditionally been perceived as an advantage for the development of parallel implementations—and taking into account that both HeteroMORPH and HeteroNEURAL introduce redundant information expected to slow down the computation *a priori*, results in Table 7 indicate that the two heterogeneous algorithms are effective in finding an appropriate workload distribution among the heterogeneous processors. On the other hand, the higher imbalance scores measured for HeteroCOM (and its homogeneous version) are likely due to the impact of inter-processor communications. In this case, further research is required to adequately incorporate the properties of the heterogeneous communication network into the design of the heterogeneous algorithm.

Taking into account the results presented above, and with the ultimate goal of exploring issues of scalability (con-

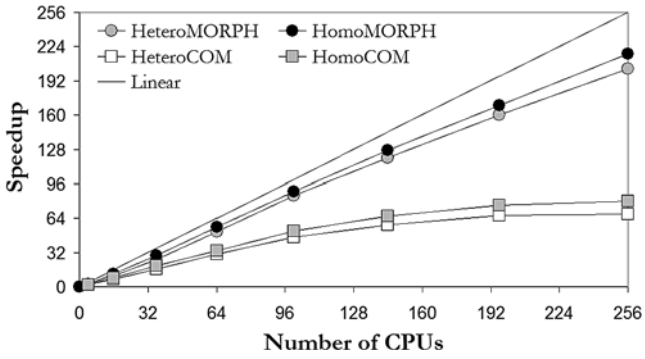

Fig. 7 Scalability of parallel morphological feature extraction algorithms on Thunderhead

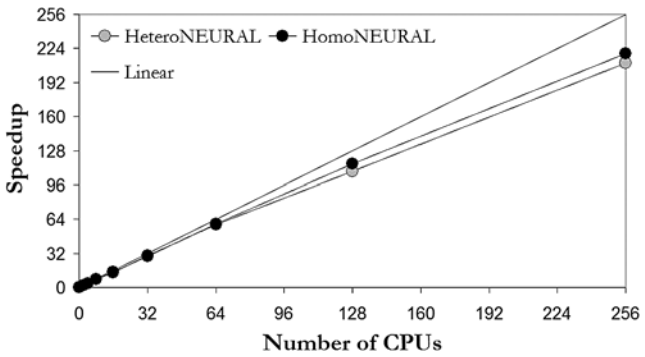

Fig. 8 Scalability of parallel neural classifier on Thunderhead

sidered to be a highly desirable property in the design of heterogeneous parallel algorithms), we have also compared the performance of the heterogeneous algorithms and their homogeneous versions on the Thunderhead Beowulf cluster. Figure 7 plots the speedups achieved by multi-processor runs of the heterogeneous parallel implementations of the morphological feature extraction algorithm over the corresponding single-processor runs of each considered algorithm on Thunderhead. For the sake of comparison, Fig. 7 also plots the speedups achieved by multi-processor runs of the homogeneous versions on Thunderhead. On the other hand, Fig. 8 shows similar results for the parallel neural network classifier. As Figs. 7 and 8 show, the scalability of heterogeneous algorithms was essentially the same as that evidenced by their homogeneous versions, with both HeteroNEURAL and HeteroMORPH showing scalability results close to linear in spite of the fact that the two algorithms introduce redundant computations expected to slow down the computation a priori. Quite opposite, Fig. 7 shows that the speedup plot achieved by HeteroCOM flattens out significantly for a high number of processors, indicating that the ratio of communications to computations is progressively more significant as the number of processors is increased, and parallel performance is significantly degraded. The above results clearly indicate that the proposed data-replication strategy is more appropriate than the tested data-communication strategy in the design of a parallel version of morphological feature extraction in the context of remote sensing applications.

For the sake of quantitative comparison, Table 8 reports the measured execution times achieved by all tested algorithms on Thunderhead, using different numbers of processors. The times reported on Table 8 reveal that the combination of HeteroMORPH for spatial/spectral feature extraction, followed by HeteroNEURAL for robust classification is able to provide highly accurate hyperspectral classification results (in light of Table 3), but also quickly enough for practical use. For instance, using 256 Thunderhead processors, the proposed classifier was able to provide a highly accurate classification for the Salinas AVIRIS scene in less than 20 seconds. In this regard, the measured processing times represent a significant improvement over commonly used processing strategies for this kind of high-dimensional data sets, which can take up to more than one hour of computation for the considered problem size, as indicated by the single-processor execution times reported on Table 3.

Overall, experimental results in our study reveal that the proposed heterogeneous parallel algorithms offer a relatively platform-independent and highly scalable solution in the context of realistic hyperspectral image analysis applications. Contrary to common perception that spatial/spectral feature extraction and back-propagation learning algorithms are too computationally demanding for practical use and/or (near) real time exploitation in hyperspectral imaging, results in this paper demonstrate that such approaches are indeed appealing for parallel implementation, not only due to the regularity of the computations involved in such algorithms, but also because they can greatly benefit from the incorporation of redundant information to reduce sequential computations at the master node and involve minimal communication between the parallel tasks, namely, at the beginning and ending of such tasks.

5 Conclusions and future research

In this paper, we have presented several innovative parallel algorithms for hyperspectral image analysis and implemented them on both heterogeneous and homogeneous networks and clusters. As a case study of specific issues involved in the exploitation of heterogeneous algorithms for hyperspectral image information extraction, this paper provided a detailed discussion on the effects that platform heterogeneity has on degrading parallel performance of a highly innovative morphological/neural classification algorithm, able to exploit the spatial and spectral information in simultaneous fashion. The parallel performance evaluation strategy conducted in this work was based on experimentally assessing the heterogeneous algorithm by comparing

Table 8 Processing times (in seconds) achieved by multi-processor runs of the considered parallel algorithms on Thunderhead

Processors:	1	4	16	36	64	100	144	196	256
HeteroMORPH	2041	797	203	79	39	23	17	13	10
HomoMORPH		753	170	70	36	22	16	12	9
HeteroCOM	2204	1177	339	146	81	53	42	37	36
HomoCOM		1144	303	127	74	48	38	35	34
Processors:	1	2	4	8	16	32	64	128	256
HeteroNEURAL	1638	985	468	239	122	61	30	18	9
HomoNEURAL		973	458	222	114	55	27	15	7

its efficiency on a fully heterogeneous network (made up of processing units with different speeds and highly heterogeneous communication links) with the efficiency achieved by its equivalent homogeneous version on an equally powerful homogeneous network. Scalability results on a massively parallel commodity cluster are also provided. Experimental results in this work anticipate that the combination of the (readily available) computational power offered by heterogeneous architectures and the recent advances in the design of last-generation Earth observation sensors and new parallel algorithms (such as those presented in this work) may introduce substantial changes in the systems currently used for exploiting the sheer volume of hyperspectral data which is now being collected worldwide, on a daily basis.

Acknowledgements The authors would like to thank J. Dorband, J.C. Tilton and J.A. Gualtieri for many helpful discussions, and also for their support with experiments on NASA's Thunderhead system. They also state their appreciation for Profs. M. Valero and F. Tirado.

References

1. Aloisio, G., Cafaro, M.: A dynamic earth observation system. Parallel Comput. **29**, 1357–1362 (2003)
2. Ayoubi, R.A., Bayoumi, M.A.: Efficient mapping algorithm of multilayer neural network on torus architecture. IEEE Trans. Parallel Distributed Syst. **14**, 932–943 (2003)
3. Baraldi, A., Binaghi, E., Blonda, P., Brivio, P.A., Rampani, A.: Comparison of the multilayer perceptron with neuro-fuzzy techniques in the estimation of cover class mixture in remotely sensed data. IEEE Trans. Geosci. Remote Sens. **39**, 994–1005 (2001)
4. Braswell, B.H., Hagen, S.C., Frolking, S.E., Salas, W.A.: A multivariable approach for mapping sub-pixel land cover distributions using misr and modis: application in the Brazilian amazon region. Remote Sens. Environ. **87**, 243–256 (2003)
5. Brown, M., Lewis, H.G., Gunn, S.R.: Linear spectral mixture models and support vector machines for remote sensing. IEEE Trans. Geosci. Remote Sens. **38**, 2346–2360 (2000)
6. Chang, C.-I.: Hyperspectral Imaging: Techniques for Spectral Detection and Classification. Kluwer, New York (2003)
7. Dorband, J., Palencia, J., Ranawake, U.: Commodity clusters at Goddard Space Flight Center. J. Space Commun. **3**, 227–248 (2003)
8. Du, Q., Chang, C.-I.: An interference rejection-based radial basis function neural network for hyperspectral image classification. IEEE Trans. Geosci. Remote Sens. **38**, 2346–2360 (1999)
9. El-Ghazawi, T., Kaewpijit, S., Moigne, J.L.: Parallel and adaptive reduction of hyperspectral data to intrinsic dimensionality. In: Proceedings of the IEEE International Conference on Cluster Computing, pp. 102–110 (2001)
10. Foody, G.M.: Applications of the self-organising feature map neural network in community data analysis. Ecol. Model. **120**, 97–107 (1999)
11. Foody, G.M., Mathur, A.: Toward intelligent training of supervised image classifications: directing training data acquisition for svm classification. Remote Sens. Environ. **93**, 107–117 (2004)
12. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Kaufman, San Francisco (1999)
13. Green, R.O.: Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS). Remote Sens. Environ. **65**, 227–248 (1998)
14. Guilfoyle, K.J., Althouse, M.L., Chang, C.-I.: A quantitative and comparative analysis of linear and nonlinear spectral mixture models using radial basis function neural networks. IEEE Trans. Geosci. Remote Sens. **38**, 2314–2318 (2001)
15. Itoh, K.: Massively parallel Fourier-transform spectral imaging and hyperspectral image processing. Opt. Laser Technol. **25**, 202–206 (1993)
16. Kalluri, S., Zhang, Z., JaJa, J., Liang, S., Townshend, J.: Characterizing land surface anisotropy from AVHRR data at a global scale using high performance computing. Int. J. Remote Sens. **22**, 2171–2191 (2001)
17. Kumar, V., Shekhar, S., Amin, M.B.: A scalable parallel formulation of the backpropagation algorithm for hypercubes and related architectures. IEEE Trans. Parallel Distributed Syst. **5**, 1073–1090 (1994)
18. Kung, S.Y., Hwang, J.N.: A unified systolic architecture for artificial neural networks. J. Parallel Distributed Comput. **6**, 357–387 (1989)
19. Landgrebe, D.A.: Signal Theory Methods in Multispectral Remote Sensing. Wiley, Hoboken (2003)
20. Larson, J.W.: Components, the common component architecture, and the climate/weather/ocean community. In: Proceedings of the 20th International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology, pp. 123–130 (2004)
21. Lastovetsky, A.: Parallel Computing on Heterogeneous Networks. Wiley-Interscience, Hoboken (2003)
22. Lastovetsky, A., Reddy, R.: On performance analysis of heterogeneous parallel algorithms. Parallel Comput. **30**, 1195–1216 (2004)

23. Plaza, A., Martinez, P., Plaza, J., Perez, R.M.: Dimensionality reduction and classification of hyperspectral image data using sequences of extended morphological transformations. IEEE Trans. Geosci. Remote Sens. **43**, 466–479 (2005)
24. Plaza, A., Valencia, D., Plaza, J., Martinez, P.: Commodity cluster-based parallel processing of hyperspectral imagery. J. Parallel Distributed Comput. **66**, 345–358 (2006)
25. Plaza, J., Chang, C.-I., Plaza, A., Perez, R.M., Martinez, P.: On the generation of training samples for neural network-based mixed pixel classification. In: Proceedings of the SPIE International Conference on Algorithms and Technologies for Multispectral, Hyperspectral and Ultraspectral Imagery, pp. 149–160 (2005)
26. Plaza, J., Plaza, A., Perez, R.M., Martinez, P.: Automated generation of semi-labeled training samples for nonlinear neural network-based abundance estimation in hyperspectral data. In: Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, pp. 345–350 (2005)
27. Ritter, G.X., Sussner, P., Diaz, J.L.: Morphological associative memories. IEEE Trans. Neural Netw. **9**, 281–293 (2004)
28. Seinstra, F.J., Koelma, D., Geusebroek, J.M.: A software architecture for user transparent parallel image processing. Parallel Comput. **28**, 967–993 (2002)
29. Soille, P.: Morphological Image Analysis: Principles and Applications. Springer, Berlin (2003)
30. Sterling, T.: Cluster computing. In: Encyclopedia of Physical Science and Technology, vol. 3 (2002)
31. Suresh, S., Omkar, S.N., Mani, V.: Parallel implementation of back-propagation algorithm in networks of workstations. IEEE Trans. Parallel Distributed Syst. **16**, 24–34 (2005)
32. Tehranian, S., Zhao, Y., Harvey, T., Swaroop, A., Mckenzie, K.: A robust framework for real-time distributed processing of satellite data. J. Parallel Distributed Comput. **66**, 403–418 (2006)
33. Votava, P., Nemani, R., Golden, K., Cooke, D., Hernandez, H.: Parallel distributed application framework for earth science data processing. In: Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, pp. 1161–1166 (2002)
34. Wang, P., Liu, K.Y., Cwik, T., Green, R.O.: MODTRAN on supercomputers and parallel computers. Parallel Comput. **28**, 53–64 (2002)
35. Yang, T., Fu, C.: Heuristic algorithms for scheduling iterative task computations on distributed memory machines. IEEE Trans. Parallel Distributed Syst. **8**, 608–622 (1997)
36. Zhang, D., Pal, S.K.: Neural Networks and Systolic Array Design. World Scientific, Singapore (2002)

Javier Plaza is currently an Assistant Professor with the Computer Science Department, University of Extremadura, Spain, where he is pursuing the Ph.D. degree. His current research work is focused on the development of efficient implementations of neural network-based algorithms for analysis and classification of hyperspectral scenes. He is also involved in the design and configuration of homogeneous and fully heterogeneous parallel computing architectures for high-performance scientific applications. Other major research interests include networking configuration and training of neural network architectures for specific applications. He has served as a reviewer for the IEEE Transactions on Geoscience and Remote Sensing and the IEEE Transactions on Image Processing journals.

Rosa Pérez received the Ph.D. degree in computer science from the Polytechnic University of Madrid, Madrid, Spain, in 1995. She is currently a Professor of Computer Science with the Computer Science Department, University of Extremadura, Caceres, Spain. She has conducted research and published extensively in many computer science-related areas, including neural networks, systolic array- and FPGA-based design, pattern recognition, and signal/image processing. She has served as a reviewer for the IEEE Transactions on Neural Networks and the IEEE Transactions on Geoscience and Remote Sensing.

Antonio Plaza received his Ph.D. degree in Computer Science from the University of Extremadura, Spain, in 2002, where he is currently an Associate Professor. He has been Visiting Researcher with the University of Maryland, NASA Goddard Space Flight Center and Jet Propulsion Laboratory. His main research interests include the development and efficient implementation of high-dimensional data processing algorithms on parallel homogeneous and heterogeneous computing systems and hardware-based computer architectures such as FPGAs and GPUs. He has authored or co-authored more than one hundred publications, and currently serves as regular manuscript reviewer for more than 15 highly cited journals in the areas of parallel and distributed computing, computer architectures, pattern recognition and image processing. He is an Associate Editor for the IEEE Transactions on Geoscience and Remote Sensing journal in the areas of Hyperspectral Image Analysis and Signal Processing, and is currently editing a book on High-Performance Computing in Remote Sensing (with Prof. Chein-I Chang) for Chapman&Hall/CRC Press. His web page address is http://www.umbc.edu/rssipl/people/aplaza.

Pablo Martínez received the Ph.D. degree in Physics from the University of Granada, Granada, Spain, in 1992. He has been a Professor of Computer Science with the University of Extremadura, Caceres, Spain, since 1985. He is currently the Head Scientist of the Neural Networks and Signal Processing Group (GRNPS). He has held Visiting Researcher positions at the Applied Information Sciences Branch, Goddard Space Flight Center, Greenbelt, MD, the Department of Electrical Engineering, University of Maryland, College Park, and the AVIRIS Group, Jet Propulsion Laboratory, Pasadena, CA. His main research interests include remote sensing, digital image analysis, hardware-based architectures, operating systems management and configuration, and neural-network-based pattern recognition. He has served as a reviewer for the IEEE Transactions on Geoscience and Remote Sensing journal.

David Valencia is an Assistant Professor at the University of Extremadura, Spain, where he is currently pursuing the Ph.D. degree. His research interests are in the development of parallel implementations of algorithms for high-dimensional data analysis, with particular emphasis on commodity cluster-based (homogeneous and heterogeneous) systems and hardware-based architectures, including systolic arrays and FPGAs. He is also involved in the design and configuration of large-scale distributed heterogeneous computing platforms.