

GRID Oriented Implementation of Self-Organizing Maps for Data Mining in Meteorology

F. Luengo¹, A.S. Cofiño², and J.M. Gutiérrez²

¹ Dept. of Computer Science

Universidad del Zulia, Maracaibo, Venezuela,

² Dept. of Applied Mathematics and Computational Sciences,

Universidad de Cantabria, Spain,

WWW home page: <http://grupos.unican.es/ai/meteo>

Abstract. We study the efficiency of different alternatives for a scalable parallel implementation of the self-organizing map (SOM) in the GRID environment of variable resources and communications. In particular, we consider an application of data mining in Meteorology, which involves databases of high-dimensional atmospheric patterns. In this work, we focus in network partitioning alternatives, analyzing their advantages and shortcomings in this framework. As a conclusion we obtain that there is no optimal alternative, and a combination (hybridization) of algorithms is required for a GRID application.

1 Introduction

CrossGrid is one of the ongoing research projects involving GRID technology. One of the main tasks in the Meteorological applications package is the implementation of data mining systems for the analysis of operational and reanalysis databases of atmospheric circulation patterns. An important problem for many meteorological applications is clustering (or partitioning) these databases to find a set of representative prototypes of the atmospheric patterns in a region of interest. In this paper we analyze an unsupervised data mining clustering technique known as Self-Organizing Map (SOM), and study the suitability of different scalable parallel implementations for the GRID environment. Previous SOM parallel algorithms have focused on parallel computers with predetermined resources (processing units) and high-performance communications (see [1] and references therein). The main goal in this project is designing an adaptive scheme for distributing data and computational load according to the changing resources available for each GRID job submitted. To this aim, some preliminary work is needed to understand the limitations and requirements of different parallel implementations. In this paper, we analyze two different algorithms for distributing computational resources, showing that none of these methods is the most efficient for all situations, but a hybrid strategy is needed.

2 Meteorological Databases

Meteorological data mining applications work with huge meteorological databases created by reanalysis projects such as the ERA/ECMWF reanalysis project, which covers the period from 1978 to 1994 (ERA-15) using a T106L31 model with 1.125° resolution. The total information volume is on the order of Terabytes, since it comprises data from approx. 20 variables (such as temperature, humidity, pressure, etc.) at 30 pressure levels of a 360×360 nodes grid. For instance, Fig. 1(left) shows the grid covering Europe for this model.

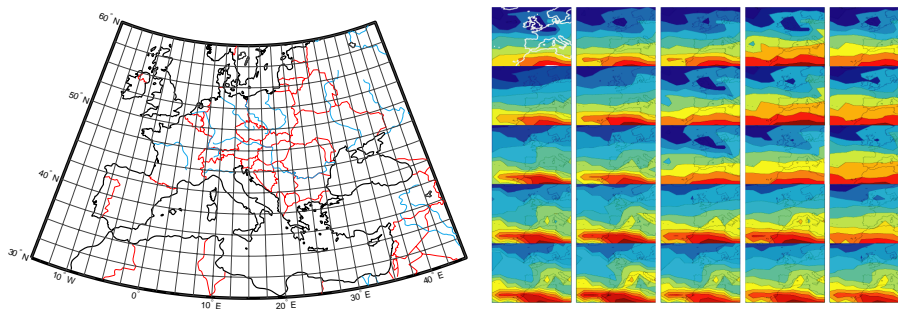


Fig. 1. (left) European region of ERA reanalysis 2.5° grid. (right) Surface temperature fields corresponding to 25 prototypes obtained with the SOM algorithm. For instance the upper-left prototype shows a isolated cold mass of air over Britain.

3 Self-Organizing Maps

Self-Organizing Maps (SOM) is one of the most popular data mining techniques, which is especially suitable for high dimensional data visualization and clustering (see [2]). It uses an unsupervised learning algorithm for creating a set of prototype vectors (cluster centers) representing the data. Moreover, a topology preserving projection of the prototypes from the original input space onto a low-dimensional lattice (usually a 2D lattice) is carried out. Self-Organized maps have been recently applied in several meteorological problems [3].

A SOM is formed by an arbitrary number of clusters C_1, \dots, C_m , located on a 2D lattice ($C_k = (x, y)$ represents the position of the cluster on the lattice); each of the clusters C_k has an associated prototype vector $c_k = (c_{k1}, \dots, c_{kd})$, which describes the position of the cluster's center on the d -dimensional data space. For instance, if we use daily surface temperature fields at 12 UTC from ERA database, then the resulting data vectors, $n = 15 \times 365 = 5500$, of dimension $d = 60 \times 30 = 1800$ characterize the temperature over Europe the period 1979-1993. Fig. 1(right) shows a 5×5 SOM trained with this data.

The vectors of the SOM are first initialized to random values. The goal of the training algorithm is iteratively adapting the prototype vectors. The batch implementation of the training proceeds in cycles; on each cycle, all data vectors are considered iteratively, one at a time (v_i), and the best-matching (or "winning")

prototype c_{k_i} is obtained as the one minimizing the Euclidean distance to the data vector:

$$\|v_i - c_{k_i}\| = \min_k \|v_i - c_k\|, k = 1, \dots, m. \quad (1)$$

After each cycle the prototypes are moved according to the closer data vectors and also to its neighbors:

$$c_j = \frac{\sum_{i=1}^n v_i h(\|C_j - C_{k_i}\|)}{\sum_{i=1}^n h(\|C_j - C_{k_i}\|)}, j = 1, \dots, m. \quad (2)$$

where the function $h(\|c_1 - c_2\|)$ is a neighborhood kernel, which determines the rate of change around the winner unit (usually a Gaussian function is considered: $h(x) = \exp(-x/s(t))$). The variance of the Gaussian neighborhood kernel $s(t)$ decrease monotonically with time, softening the topologic constrains – a linear decay to zero is usually chosen for these functions. – The neighborhood adaptation mechanism is what makes SOM different from other clustering algorithms, so neighboring clusters in the 2D lattice space are quite similar, while more distant clusters become increasingly diverse. For a detailed description of different implementations of the method, the reader is referred to Oja et al. [2].

4 Parallelization in the GRID Environment

We have used a cluster of 80 x220 IBM servers, Pentium III, 1.26 Ghz with 512 MB RAM, and 90 GB hard disks, connected with 100 Mbps Ethernet LAN (see <http://grid.ifca.unican.es/> for details). All the programs have been developed in C language using the MPICH-chp4 implementation of Message Passing Interface (in the future, experiments in the GRID will be carried out using Globus). Due to the inhomogeneous cluster communications in the GRID, a centralized master-slave architecture is chosen to avoid intensive message passing among computing units. The simplest form for parallelizing the SOM algorithm is splitting up the data, and hence the sum in (2), between different processors. However, this is not an efficient implementation for the GRID, since it requires intensive message passing of high-dimensional data.

Figures 2(a) and (b) shows two different alternatives for distributing computational resources with replicated and centralized prototype vectors, respectively. The different messages required for each of the schemes are shown in the figure using dashed lines, which may correspond to either an iteration of the algorithm, or a whole cycle. In Fig. 2(a), messages are minimized in size, but the prototypes are replicated and updated in each computing element (we shall denote this algorithm SOM_R). On the other hand, in Fig. 2(b) calculations are only performed at the master, but messages to update the centers are sent to the slaves after each cycle (SOM_C).

To check the efficiency of these algorithms, we performed several experiments varying the number of clusters m , the dimension of data d , the size of the database n , and the number of cycles. Figure 3 shows the speedup curves obtained for $d = 500$, $n = 5500$ and variable SOM size (from $10 \times 5 = 50$ to

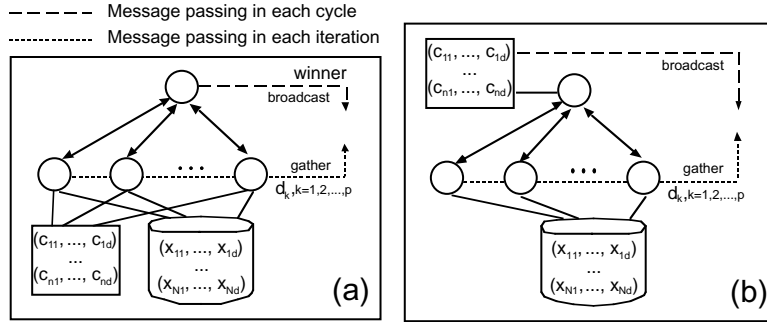


Fig. 2. Two different parallel schemes for the SOM training algorithm for distributing computational resources with: (a) replicated prototype vectors; (b) centralized prototype vectors.

$40 \times 40 = 1600$ clusters, or prototypes); the cluster prototypes are distributed in two, four, up to twenty processing units, obtaining the speedup curves for SOM_R and SOM_C schemes. This figure indicates that none of the schemes is the most efficient for all situations, since SOM_R achieves better speedup curves for small and medium network sizes, up to a given number of processing units, but SOM_C is more robust for large network sizes. Therefore, the distribution of data and computing resources has to be done according to the changing resources available for each GRID job submitted.

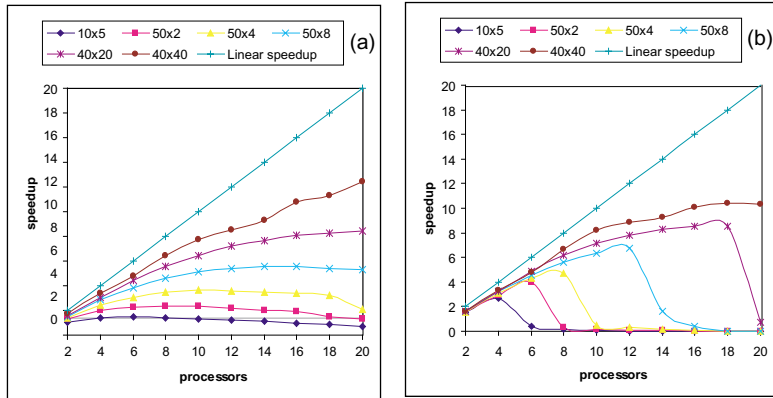


Fig. 3. Speedup for (a) SOM_R (b) SOM_C . $d = 500$, $n = 5500$.

References

1. Lawrence, R.D., Almasi, G.S., and Rushmeier, H.E.: A scalable parallel algorithm for self-organizing maps with applications to sparse data mining problems, *Data Mining and Knowledge Discovery* **3** (1999) 171–195.
2. Oja, E., and Kaski, S.: *Kohonen Maps*. Amsterdam, Elsevier, 1999.
3. Cavazos, T.: Using self-organization maps to investigate extreme climate event, *Journal of Climate* **13** (2000) 1718–1732.