Applied Soft Computing 11 (2011) 733–743

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Efficient Distributed Genetic Algorithm for Rule extraction

Miguel Rodríguez*, Diego M. Escalante, Antonio Peregrín

Dept. of Information Technologies, University of Huelva, 21819 Huelva, Spain

ARTICLE INFO

Article history:
Received 15 March 2009
Received in revised form
18 December 2009
Accepted 29 December 2009
Available online 13 January 2010

Keywords:
Classification rules
Rule induction
Distributed computing
Coarse-grained implementation
Parallel genetic algorithms

ABSTRACT

This paper presents an Efficient Distributed Genetic Algorithm for classification Rule extraction in data mining (EDGAR), which promotes a new method of data distribution in computer networks. This is done by spatial partitioning of the population into several semi-isolated nodes, each evolving in parallel and possibly exploring different regions of the search space. The presented algorithm shows some advantages when compared with other distributed algorithms proposed in the specific literature. In this way, some results are presented showing significant learning rate speedup without compromising the accuracy.

1. Introduction

Nowadays the size of datasets is growing quickly due to the widespread use of automatic processes in commercial and public domains and the lower cost of massive storage. Mining large datasets to obtain classification models with prediction accuracy can be a very difficult task because the size of the dataset can make data mining algorithms inefficacy and inefficient.

There are three main approaches to tackling the scaling problem:

- Use as much as possible a priori knowledge to search in subspaces small enough to be explored.
- Perform data reduction.
- Algorithm scalability.

The third approach, algorithm scalability, promotes the use of computation capacity in order to handle the full dataset. The use of computer grids to achieve a greater amount of computational resources has become more popular over the past few years because they are much more cost-effective than single computers of comparable speed. The main challenge when using distributed computing is the need for new algorithms that take the architecture into account. Genetic algorithms are especially well suited for this task because of their implicit parallelism. As a typical population algorithm, there is a direct way of distributing the algorithm by the use of several smaller populations that interchange individuals occasionally. This kind of distributed GA achieves a significant speedup when used on a network of computers and prevents an early convergence by keeping diversity in several populations.

There have been several efforts to make use of models based on distributed genetic algorithms (GA) in data mining emphasising aspects like scalability and efficiency. REGAL [10] and NOW G-Net [1] are well known references of this approach. Both of them increase the computational resources via the use of data distribution on a network of loosely coupled workstations.

In this paper we present an Efficient Distributed Genetic Algorithm for classification Rule extraction (EDGAR) with dynamic data partitioning that shows advantages in scalability for exploring high complexity search spaces with comparable classification accuracy.

The outline of the contribution is as follows: in Section 2 we review the distributed genetic models in rule induction. Section 3 is devoted to analysing the proposed algorithm and the strategies followed to keep the algorithm scalable. The experimental study developed is shown in Section 4 and finally, we reach some conclusions in Section 5. Appendix A is included containing a detailed table of results obtained in our study.

2. Machine learning and parallel genetic algorithms

This section reviews the main streams found in literature about parallel genetic algorithms in rule induction. The first subsection describes the approaches commonly used in the area to achieve machine learning. The second subsection is focused on the

* Corresponding author. Tel.: +34 959217372.
E-mail addresses: miguel.rodriguez@dti.uhu.es (M. Rodríguez), diego.escalante@dti.uhu.es (D.M. Escalante), peregrin@dti.uhu.es (A. Peregrín).

1568-4946/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.12.035

main strategies for parallelising genetic algorithms in data mining related tasks.

2.1. Genetic algorithms in data mining

Genetic algorithms are search algorithms based on natural genetics that provide robust search capabilities in complex spaces, and thereby offer a valid approach to problems requiring an effective search process [15].

GA has achieved a reputation for robustness in rule induction, in common problems associated with real world mining (noise, outliers, incomplete data, etc.). GA can be dedicated to machine learning algorithms [3]. For example, the search space can be seen as the entire possible hypothesis rule base that covers the data and the goodness can be formulated as a coverage function over a number of learning examples.

A key point in GA implementation is the selected representation; the proposals in the specialist literature follow two approaches in order to encode rules within a population of individuals:

The "Chromosome = Set of rules", also called the Pittsburgh approach, in which each individual represents a rule set [23]. The chromosome evolves a complete rule set and they compete among themselves throughout the evolutionary process. GABIL [6] and GA-MINER [9] are proposals that follow this approach.

The "Chromosome = Rule" approach, in which each individual codifies a single rule, and the whole rule set is provided by combining several individuals in a population (rule cooperation) or via different evolutionary runs (rule competition).

In turn, within the "Chromosome = Rule" approach, there are three generic proposals:

- The Michigan approach, in which each individual encodes a single rule. These kinds of systems, usually called learning classifier systems [14], are rule-based, message-passing systems that employ reinforcement learning and a GA to learn rules that guide their performance in a given environment. The GA is used to detect new rules that replace the bad ones via the competition between the chromosomes in the evolutionary process.
- The IRL (Iterative Rule Learning) approach uses several GA executions to obtain the rule set. Chromosomes compete in every GA run, choosing the best rule per run. The global solution is formed by the best rules obtained when the algorithm is run multiple times. SIA [26] is a proposal that follows this approach.
- The GCCL (genetic cooperative–competitive learning) approach encodes the rule set as the complete population or a subset of it. The chromosomes compete and cooperate simultaneously. This strategy requires the conservation of species in the same population to avoid a final solution consisting of clones of the best individual. COGIN [13], REGAL [10] and NOW G-Net [1] are examples using this representation.

2.2. Parallel genetic algorithms in data mining

The definition of scalable in computing could be something like "able to support the required quality of service as the system load increases". Applied to data mining and more precisely to supervised classification, the system load is provided by the complexity and the size of the dataset (in attributes or training examples), and the quality of service is relative to the processing time for producing a similar classifier mainly in terms of accuracy and interpretability.

As complexity of the dataset increases, GAs exhibit high computational cost and degradation of the quality of the solutions. Efforts towards solving these shortcomings have been made in several directions, and parallel GAs is one of the most significant. We stress four approaches that represent the main parallelisation strategies:

- Global parallelisation [8]. Only the evaluation of individuals' fitness values is parallelised by assigning a fraction of the population to each processor to be evaluated. This is an equivalent algorithm that will produce the same results as the sequential one.
- Coarse-grained [8] and fine-grained parallelisation [20]. In the former, the entire population is partitioned into subpopulations (demes). A GA is run on each subpopulation, and exchange of information between demes (migration) takes place occasionally [8] in analogy with the natural evolution of spatially distributed populations such as the island model (Fig. 1). Fine-grained course has just one individual per processor and rules to perform crossover in the closest neighbourhood defined by a topology.
- Supervised data distribution [11]. A master process uses a group of processors (slaves) by sending them partial tasks and a smaller data partition. Each node has a complete GA or a part of it. The master process uses the partial results to reassign data and tasks [10,1] to the processors until some condition is met.
- Not supervised data distribution [18]. The full dataset is shared out in several processors and moved to the next processor in the topology after a pre-specified number of generations without removing the existing population. The individuals will try to cover the newly arrived training data.

The proposal presented in this work follows a GCCL approach and as a parallelisation strategy uses a coarse-grained implementation and a master process to build up the final classifier on the basis of partial results.

3. Genetic Learning Proposal: EDGAR algorithm

This section describes the characteristics of an *Efficient Distributed Genetic Algorithm for classification Rules extraction*, from now on designated EDGAR. The proposed algorithm distributes population (rules in a GCCL approach) and training data in a coarse-grained model to achieve scalability.

We start by explaining the distributed model in Section 3.1. Sections 3.2–3.5 describe the components of the genetic algorithm: representation, genetic operators, genetic search and data reduction. Finally, Section 3.6 is devoted to the strategy used to determine the best set of rules that will make up the classifier from the redundant population of rules generated by the GCCL algorithm.

3.1. Distributed model

This subsection explains the main properties of the distributed framework. First, in Section 3.1.1 we describe the use of the coarse-grained implementation with data partition.

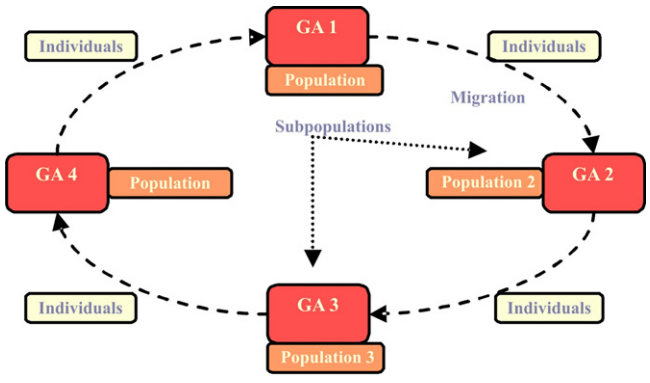

Fig. 1. Island model.