Rule Discovery with a Parallel Genetic Algorithm

Dieferson L.A. Araujo
CEFET-PR
CPGEI.
Av. 7 de Setembro, 3165.
Curitiba - PR.
80230-901. Brazil
dief@cpgei.cefetpr.br

Heitor S. Lopes
CEFET-PR
CPGEI.
Av. 7 de Setembro, 3165.
Curitiba - PR.
80230-901. Brazil
hslopes@cpgei.cefetpr.br

Alex A. Freitas
PUC-PR
PPGIA-CCET.
R. Imaculada Conceicao, 1155
Curitiba – PR. 80.215-901. Brazil
alex@ppgia.pucpr.br
http://www.ppgia.pucpr.br/~alex

Abstract

An important issue in data mining is scalability with respect to the size of the dataset being mined. In the paper we address this issue by presenting a parallel GA for rule discovery. This algorithm exploits both data parallelism, by distributing the data being mined across all available processors, and control parallelism, by distributing the population of individuals across all available processors.

1 INTRODUCTION

An important issue in data mining is how a knowledge discovery algorithm scales up with respect to the size of the database being mined [Provost & Kolluri 1999]. Intuitively, parallel processing can be regarded as a natural solution to the problem of scalability in data mining [Freitas & Lavington 1998]. Since genetic algorithms (GAs) tend to be slow, in comparison with most rule induction methods, the design of parallel GAs for data mining is an important research area [Flockhart & Radcliffe 1995], [Giordana & Neri 1995], [Anglano et al. 1997], [Anglano et al. 1998], [Araujo et al. 1999].

This paper presents GA-PVMINER, a parallel GA for rule discovery in data mining. In this paper we focus only on the parallelization aspects of GA-PVMINER. In particular, we report only results concerning the speed up of the parallel version of the algorithm over its sequential version. Results concerning the quality of discovered rules are beyond the scope of this paper, and are described in detail in [Araujo et al. 1999].

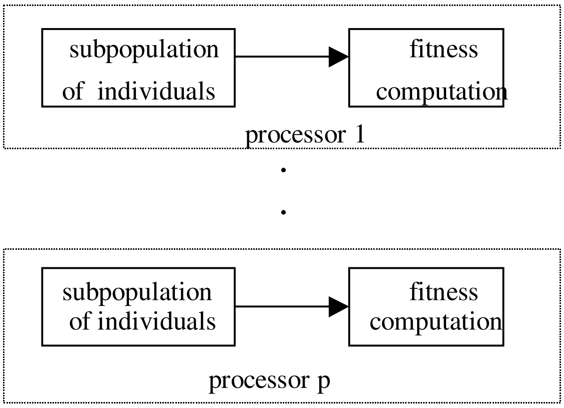
The data mining task addressed in this paper is dependence modeling. This task can be regarded as a generalization of the classification task [Noda et al. 1999]. In dependence modeling, similarly to classification, the aim is to discover rules that predict the value of a goal attribute, given the values of predictor attributes. However, in classification there is a single goal attribute to be predicted, while in dependence modeling there is more than one goal attribute. In our approach for dependence modeling, the user specifies a small set of potential goal attributes, which (s)he is interested in predicting.

The rest of this paper is organized as follows. Section 2 presents an overview of parallel GAs. Section 3 presents an overview of GA-PVMINER. Section 4 discusses how GA-PVMINER exploits parallelism to reduce processing time. Section 5 reports some computational results. Finally, section 6 concludes the paper.

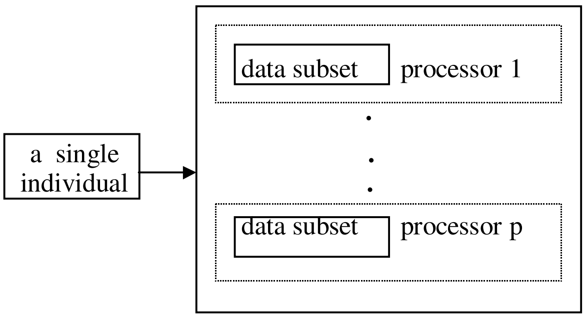
2 AN OVERVIEW OF PARALLEL GENETIC ALGORITHMS

There are two broad sources of parallelism in genetic algorithms. One can exploit parallelism in the application of genetic operators - such as selection, crossover, mutation - and/or in the computation of the fitness of the population individuals (candidate rules). In the context of mining very large databases, the latter tends to be far more important. The reason is that the genetic operators are usually very simple and their application is computationally cheap. Hence, the bottleneck of the algorithm is the computation of the individuals' fitness, whose processing time is proportional to the size of the data being mined.

In a high level of abstraction, there are two basic approaches to exploit parallelism in fitness computation [Freitas & Lavington 1998]. One approach consists of exploiting inter-individual (or inter-fitness-computation) parallelism. This approach is illustrated in Figure 1(a). In this approach the set of individuals in the current population is distributed across all the processors. Different subsets of individuals have their fitness computed in parallel by different processors. Hence, this is a control-parallel approach, in the sense that the flow of control of the algorithm is parallelized. On the other hand, this approach is data-sequential, since the computation of the fitness of a given individual is done by a single processor handling the data being mined in sequential mode. Note that the computation of the fitness of each individual requires access to the entire data being mined. On a distributed-memory machine, in order to avoid high data traffic across the inter-processor network, the entire data being mined would have to be replicated in each processor node, which makes it difficult to scale this approach to very large databases.



(a) Exploiting inter-individual, control parallelism



(b) Exploiting intra-individual, data parallelism

Figure 1: Two approaches for exploiting parallelism in the fitness computation of a genetic algorithm.

Inter-individual parallelism, i.e. control parallelism, is the kind of parallelism exploited by the majority of parallel genetic algorithms. However, these algorithms are typically designed for solving combinatorial optimization problems, which are CPU-bound rather than I/O-bound. The situation is different in the context of data mining, which is a data-intensive application. In this context, an intuitively promising approach to exploit parallelism in fitness computation consists of exploiting intra-individual (or intra-fitness-computation) parallelism. This approach is illustrated in Figure 1(b). In this approach the data being mined is distributed across the processors, and the computation of the fitness of each individual is done by handling the data in parallel. Hence, this is a data-parallel approach. On the other hand, this approach is control-sequential, since fitness computation is done one-individual-at-a-time. The most important advantage of this approach, in the context of data mining, is that, intuitively, it is much more scalable with respect to the size of the data being mined than the control-parallel approach. To put it in simply terms, more data leads to a larger degree of data parallelism to be exploited.

Note that data and control parallelism address different kinds of `large` problems. Data parallelism addresses the problem of very large databases. Control parallelism addresses the problem of very large search spaces. Hence, it would be desirable to exploit both kinds of parallelism in a genetic algorithm for data mining. This is the goal of the algorithm described in the next section.

3 AN OVERVIEW OF GA-PVMINER

In this section we briefly describe the main aspects of GA-PVMINER. In this algorithm each individual represents a single prediction rule of the form `IF *C* THEN *P*` where *C* and *P* represent respectively the condition and the prediction of the rule. The condition *C* is a conjunction of terms. A term is a triple of the form <Attribute = Value>. The current version of GA-PVMINER assumes that all attributes are categorical – i.e. continuous attributes would have to be discretized in a preprocessing step. The *C* part of the rule is encoded as a variable-length conjunction of terms. The length (number of terms) of this part ranges from 1 to *m*, where *m* is the number of predictor attributes. The length of this part is controlled by two genetic operators: term removal and, to a lesser extent, crossover. The details of these operators are beyond the scope of this paper. The interested reader is referred to [Araujo et al. 1999]. The prediction *P* is a triple of the form <Goal_Attribute = Value>, where Goal_Attribute is one of the goal attributes specified by the user. Notice that different rules can have different goal attributes in their *P* part, since GA-PVMINER addresses the task of dependence modeling, as explained in the Introduction.

The population is divided into *n* subpopulations, each of them with *N* individuals. For each subpopulation, all the individuals represent rules with the same goal attribute and the same goal attribute value in the *P* part of the rule. Hence, this part of the individual is fixed and does not undergo the action of genetic operators. The number of subpopulations *n* is a user-specified parameter, and it should be greater than or equal to the number of goal attributes specified by the user. If this constraint is not respected some goal attribute(s) will not occur in any discovered rule.

Our motivation for dividing the population into several subpopulations is twofold. Firstly, an individual can mate only with another individual of the same population. This is a simple solution to the problem of avoiding the exchange of genetic material between individuals (rules) that are being evolved to predict different goal attributes. Secondly, this kind of population partitioning greatly facilitates the exploitation of parallelism, as will be seen in the next section.

We mention in passing that the GA described in this paper was designed for discovering knowledge that is not only accurate and comprehensible, but also *interesting* (novel, surprising). To achieve this goal the fitness function is partially based on a modified version of the J-measure of rule interestingness – again, see [Araujo et al. 1999].

4 EXPLOITING PARALLELISM IN GA-PVMINER

As mentioned above, in GA-PVMINER the global population of individuals (rules) is divided into *n* subpopulations. Each subpopulation is assigned to a distinct *logical* processor node. Let *p* be the number of (*physical*) processor nodes available in the parallel machine or the network of workstations/PCs. We first discuss the case where $n = p$. The cases where $n < p$ or $n > p$ will be discussed later.

In the case where n = p, each subpopulation is allocated to a distinct processor node and all processor nodes are used. All the subpopulations evolve in parallel. In addition the data being mined is also partitioned across the available processors. This approach has two related advantages. Firstly, it allows the exploitation of data parallelism, as explained below. Secondly, it avoids the problem of replicating the data being mined across all processors, which would reduce scalability for large databases. Each generation of the genetic algorithm consists of two phases, namely fitness evaluation and application of genetic operators. Both these phases exploit parallelism, as follows.

The fitness evaluation phase exploits both data parallelism and control parallelism by having the individuals passing through all the processors in a kind of round-robin scheme. In this scheme the physical interconnection of processor nodes is mapped into a logical ring of processor nodes, so that each processor node has a right neighbour and a left neighbour.

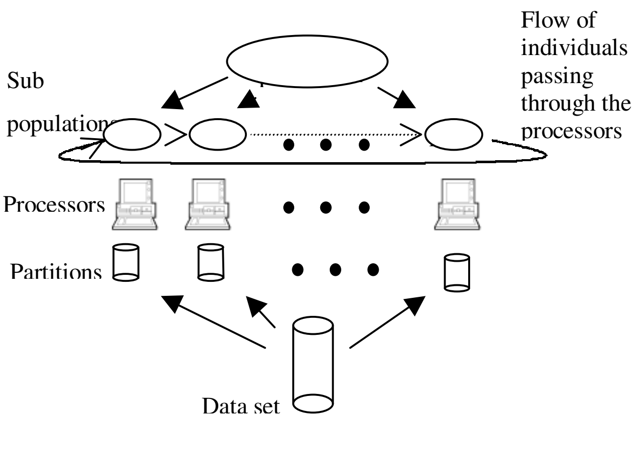


Figure 2: Exploring parallelism in GA-PVMINER

At first each processor nodes computes a partial measure of fitness for all the individuals (rules) in its local subpopulation, by accessing only its local dataset. Then each processor transfer its entire local subpopulation of individuals, as well as the value of their partially-computed fitness function, to its right neighbour. As soon as a processor node receives a subpopulation of individuals from its left neighbour, it performs the following tasks: (a) it computes the partial fitness measure of the incoming individuals on its local data set; (b) it combines this partial fitness measure with the previous one of the incoming individuals to produce a new fitness measure; (c) it forwards the incoming individuals, as well as their updated partial fitness measure, to its right neighbour. This process is repeated until all individuals have passed through all the processors and returned to their original processors, with their final fitness value duly computed. This scheme is illustrated in Figure 2. Note that what is being passed through the processors are only individuals and their partial fitness value, not the data being mined. This minimizes interprocessor communication overhead.

The phase of application of genetic operators exploits control parallelism. In this phase each processor applies genetic operators to the individuals of its local subpopulation. This is done in parallel for all the subpopulations, and the application of genetic operators in each subpopulation is completely independent of the application of genetic operators in the other subpopulations. Hence, this phase requires no interprocessor communication.

The above discussion assumed that n = p. Two other cases are possible, namely $n < p$ and $n > p$. In both cases the basic idea of the above arguments still holds, although of course some opportunity to exploit parallelism will be wasted. The differences are as follows. Firstly, suppose that $n < p$. In this case each subpopulation will be allocated to a distinct processor node, but some processor nodes will be idle, so that the degree of parallelism associated with the parallel machine will be underexploited. Secondly, suppose that $n > p$. In this case the *n* subpopulations will be allocated to the *p* processors in a round-robin scheme, in order to ensure that the subpopulations will be as evenly distributed as possible across the available processors, achieving a good workload balance. In this case a physical processor node will be in charge of evolving more than one subpopulation, so that the degree of parallelism associated with the data mining task will be underexploited.

GA-PVMINER uses PVM (Parallel Virtual Machine), a software environment that allows a cluster of heterogeneous computers to be viewed as a single parallel machine [Geist et al. 1994].

5 COMPUTATIONAL RESULTS

This section reports the results of experiments using two datasets obtained from the UCI repository of datasets at http://www.ics.uci.edu/AI/Machine-Learning.html. The datasets in question are Nursery and Adult. In the absence of a specific benchmark dataset for the dependence modeling task, these datasets were chosen partially because they seem to contain more than one potential goal attribute and partially due to their relatively large size, in comparison with other datasets of the UCI repository.

The Nursery dataset contains 12960 instances (records) and 9 attributes. In our experiments we have specified 3 goal attributes for this dataset, namely Recommendation, Social and Finance. The Adult dataset contains 48844 instances and 15 attributes. We have specified two goal attributes for this data set, namely Workclass and Class (indicating whether or not salary is greater than 50k).

In all the experiments the genetic algorithm had 200 individuals in each subpopulation, and it was run for 100 generations.

The experiments were performed on a parallel virtual machine (PVM) consisting of four 350-MHz Pentium II computers, each with 32 MB of main memory and 6 GB of disk, with operating system Linux RedHat 5.2 and PVM 3.3.11. The interconnection network was Ethernet with 10 Mbps. In our system one of the four processors runs the master program, which controls the slave programs (each running a subpopulation). The processor running the master program also runs one slave program.

The experiments have measured the Speed up (Sp) of the parallel version of the algorithm over its sequential counterpart, defined as: $Sp = T_s/T_p$, where T_s is the sequential processing time (on a single processor) and T_p is the parallel processing time (on *p* processors). Tables 1 and 2 show the Sp results for the Nursery and Adult datasets, respectively. In both these tables the parallel processing time shown in the fourth column is the sum of the processing time itself and the initialization time – required to distribute data and individuals across the processors.

Table 1: Speed up results for the Nursery data set

Number of Processors	Number of sub-populations	Sequential processing time	Parallel processing time	Speed up
1	1	134 sec.	-	-
3	3	402 sec.	252 sec.	1.595
4	4	536 sec.	320 sec.	1.675

Table 2: Speed up results for the Adult data set

Number of Processors	Number of sub-populations	Sequential processing time	Parallel processing time	Speed up
1	1	1052 sec.	-	-
3	3	3156 sec.	1364 sec.	2.313

As shown in Tables 1 and 2, the parallel version of GA-PVMINER achieved a reasonable speed up over the sequential version. As expected, the speed up was greater in the case of the Adult dataset. The reason is that this dataset is larger than the Nursery dataset, so there is more opportunity for the exploration of data parallelism in the former. Of course, real-world databases can be much larger than the two public domain datasets used in our experiments. Therefore, we can expect that our system will achieve even higher speed ups in large real-world databases, which should be investigated in further research.

6 CONCLUSIONS AND FUTURE RESEARCH

We have described a hybrid parallelization strategy for a GA that discovers prediction rules in the dependence modeling task. The proposed strategy exploits both data parallelism, by distributing the data being mined across all available processors, and control parallelism, by distributing the population of individuals across all available processors. The results show that a good speed up can be achieved provided that the data being mined has a relatively large size. This is due to the fact that the degree of data parallelism is proportional to the size of the data being mined. As mentioned above, future work should include a more extensive set of experiments with larger, real-world databases, to further validate the empirical results reported in this paper.

We emphasize that, due to space limitations, in this paper we have focused on the parallel-processing aspects of GA-PVMINER and on scalability issues. As mentioned above, a more detailed description of the algorithm as well as an analysis of the quality of discovered rules can be found in [Araujo et al. 1999].

REFERENCES

Anglano, C; Giordana, A.; Lo Bello, G. and Saitta, L. (1997) A network genetic algorithm for concept learning. *Proc. 7th Int. Conf. Genetic Algorithms*, 434-441. Morgan Kaufmann.

Anglano, C; Giordana, A.; Lo Bello, G. and Saitta, L. (1998) An experimental evaluation of coevolutive concept learning. *Machine Learning: Proc. Fifteenth Int. Conf.*, 19-27. Morgan Kaufmann.

Araujo, D.L.A.; Lopes, H.S. and Freitas, A.A. (1999) A parallel genetic algorithm for rule discovery in large databases. *Proc. 1999 IEEE Systems, Man and Cybernetics Conf.*, v. III, 940-945. Tokyo, Japan.

Flockhart, I.W. and Radcliffe, N.J. (1995) GA-MINER: parallel data mining with hierarchical genetic algorithms – final report. *EPCC-AIKMS-GA-MINER-Report 1.0.* University of Edinburgh, UK.

Freitas, A.A. and Lavington, S.H. (1998) *Mining Very Large Databases with Parallel Processing*. Kluwer.

Geist, A.; Benguelin, A.; Dongarra, J.; Jiang, W.; Manchek, R. and Sunderam, V. (1994) *PVM – Parallel Virtual Machine – A user's guide and tutorial for networked parallel computing.* MIT Press.

Giordana, A. and Neri, F. (1995) Search-intensive concept induction. *Evolutionary Computation 3(4)*: 375-416.

Noda, E.; Freitas, A.A. and Lopes, HS. (1999) Discovering interesting prediction rules with a genetic algorithm. *Proc. CEC-99*, 1322-1329. Washington D.C., USA.

Provost, F. and Kolluri, V. (1999) A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery 3(2)*, 131-169.