

PARALLEL GENETIC ALGORITHMS FOR TUNING A FUZZY DATA MINING SYSTEM

QITAO LIU

Department of Computer Science
Mississippi State University

SUSAN M. BRIDGES

Department of Computer Science
Mississippi State University

IOANA BANICESCU

Department of Computer Science
Mississippi State University

ABSTRACT

In previous work, we have described methods that we have developed for tuning a fuzzy data mining system for intrusion detection using a hierarchical genetic algorithm. Unfortunately, the genetic algorithm approach is very slow due to the computational cost of the evaluation function. In this paper, we describe parallel implementations of the genetic algorithm that were run on both a multiprocessor Unix workstation and a high performance cluster running Linux. Very little speedup was achieved on the Unix workstation because of contention for the single file system; significant speedup was achieved with the cluster in which each node had its own file system. Experimental results of our implementations based on the master-slave parallel model and multiple population parallel model are compared, and the preliminary results indicate that the multiple population model may provide a higher quality solution for a shorter amount of time.

INTRODUCTION

Kuok, Fu, and Wong (1998) introduced the marriage of fuzzy logic and association rule mining to address the sharp boundary problem encountered when discretizing continuous attributes for association rule mining. We have further refined these techniques for intrusion detection applications (Luo and Bridges 2000). One of the difficulties encountered in this approach, however, is defining appropriate membership functions for the fuzzy sets that serve as values of the fuzzy attributes. We have subsequently found that genetic algorithms (GAs) are effective methods for tuning the fuzzy membership functions and for feature selection (Shi 2000; Bridges and Vaughn 2000). Unfortunately, the evaluation function for the genetic algorithm requires that data mining be performed for every member of the population at every iteration of the genetic algorithm. Although, in the intrusion detection domain, the tuning process is an off-line operation, the time required for the GA is very substantial. In order to improve the performance of the GA, we have investigated parallel implementations of the genetic algorithm for data mining on both a multiprocessor Unix workstation and a high performance cluster running Linux.

In the remainder of this paper, we first describe our fuzzy data mining process and the sequential genetic algorithm that we have developed for tuning

the membership functions and for feature selection. We then present our parallel implementations of the algorithm and preliminary experimental results that assess the effectiveness of the parallel implementations. Conclusions and plans for future work are given in the last section.

GENETIC ALGORITHMS FOR FUZZY DATA MINING

We have developed techniques for detecting network intrusions that integrate fuzzy logic and data mining (Luo and Bridges 2000). The intrusion detection system uses fuzzy association rules to represent normal patterns of quantitative attributes recorded in network audit data. When monitoring a new data set for intrusions, one compares the similarity of the rules mined from the new set with the rules mined from “normal” data. If the similarity falls below a specified threshold, the data is flagged as suspicious. Effective application of this method requires that informative attributes of the audit data be used and that the membership functions of the fuzzy sets be appropriately defined. We have subsequently investigated the feasibility of using hierarchical GAs to select optimized feature subsets and to tune the membership functions of the fuzzy sets that serve as attribute values (Shi 2000, Bridges and Vaughn 2000). Figure 1 illustrates the hierarchical structure of chromosomes.

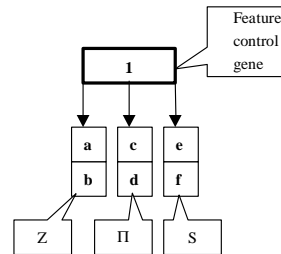


Figure 1. Representation of a single fuzzy attribute; the upper box indicates if “feature” is selected where 1 = yes and 0 = no. The boxes labeled a, b, c, d, e, f are parameters of the Z, Π , and S membership functions.

The fuzzy data mining process uses fuzzy sets as possible values for attribute values. For a particular attribute, we need to determine if this attribute should be used for data mining, and if so, the membership functions for the fuzzy sets to be used as values for the fuzzy attributes. A hierarchical genetic algorithm is used. At the first level, binary control genes are used to represent selection/non-selection of the attribute. At the next level, real values are used for parametric genes that represent the parameters of the Z, S and Π membership functions. In this application, the fitness function was designed to measure the performance of a classification system based on data mining that uses the selected features and the membership functions (Shi 2000). This data mining step makes evaluation of the fitness function a very time-consuming process and thus a prime candidate for parallel implementation.

PARALLEL GENETIC ALGORITHMS

Many research groups have investigated parallel genetic algorithms (PGAs). (See Cantú-Paz (1998) for a review.) Models for parallel GAs are often classified as single-population models or multiple-population models.

Single-population GAs are generally implemented using a master-slave model. In the master-slave model, a single population resides in the master processor and the master processor does the selection, crossover, and mutation; only evaluation of the fitness function is distributed among slave processors. In a multiple-population model, the population is divided into several subpopulations that are assigned to different processors. Each processor applies a traditional sequential GA (SGA) independently to its own subpopulation. Occasionally, individuals are exchanged between subpopulations in a process called migration. We have investigated both single population and multiple population genetic algorithms.

The global single-population master-slave model that we use is illustrated in Figure 2. A portion of the population is distributed to each slave processor for evaluation of the fitness value of individuals. The master processor also retains a portion of the population so that it can carry out evaluation in parallel with the slave processors. Genetic operations other than evaluation are performed only by the master processor. The master processor assigns a fraction of the population to each slave processor for each generation.

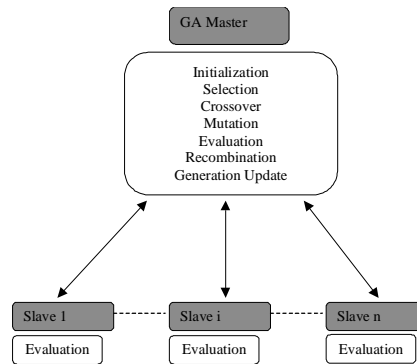


Figure 2. Global single-population master-slave PGA. The master stores the population, executes GA operations, and sends subsets of the population to the slaves. Each slave only evaluates the fitness of the individuals in its subpopulation, and sends the fitness value back to master.

Multiple-population GAs are also widely used parallel methods, but they are more complex than single population methods. A key characteristic of multiple-population PGAs is the migration of individuals among subpopulations. Each subpopulation is managed by an independent SGA except that the processors periodically exchange individuals. Some important parameters in multiple-population GAs are 1) which other processors a processor exchanges individuals with, 2) how often processors exchange

individuals (the rate for the migration), 3) how many individuals processors exchange with each other (the number of individuals to migrate), and 4) what method is used to exchange (the topology for the exchange) (Cantú-Paz 1998). The most commonly used communication topology is a ring (see Figure 3). Individuals are exchanged for two reasons: to increase the fitness of the other population and to maintain diversity in the subpopulations. Many different approaches for migration have been developed. A schematic of the multiple-population PGA we have used is given in Figure 3. The subpopulations are virtually positioned on an oriented ring. Every time a new generation is computed, at the migration generation (every two generations) a copy of the best individual, i.e., the one with the greatest fitness value in the current subpopulation, is sent to the next subpopulation on the oriented ring. Each subpopulation thus receives a new individual that replaces the worst individual with the lowest fitness value.

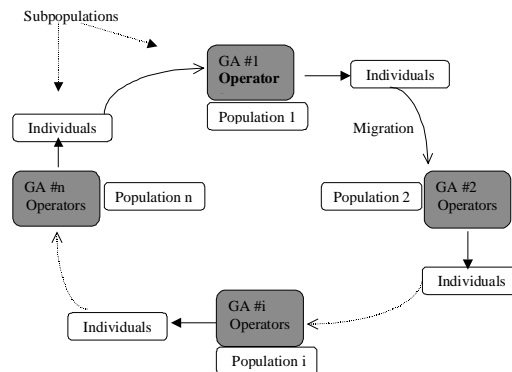


Figure 3. A schematic of a multiple-population GA. Each processor executes a SGA for its subpopulation, and there is communication (migration) between the subpopulations. Operators include all GA operators, initialization, selection, crossover, mutation, evaluation.

EXPERIMENTS AND RESULTS

A master-slave model of the GA was implemented and run on both a multi-processor Sun workstation and a Linux cluster. The workstation was a Sun E450 with 4 processors (300 MHz UltraSparc II CPU, 1024 MB Ram, one 9GB hard drive, SunOS 5.7). The Linux cluster consists of 14 machines (Dual PIII 450 or 550 processors, 512 MB SDRAM, Dual 6 GB IDE HD, Redhat Linux 6.2) which are connected by full duplex 100Mb Ethernet. A multiple population model was also implemented and run on the cluster.

When the master slave implementation was run on the workstation, little improvement in performance occurred with the use of additional processors. The fuzzy data mining algorithm used for the evaluation of the fitness function reads and writes files for every evaluation. Because the Unix system has a single file system, only one processor can access the file system at any one time. The result is that the parallel implementation runs in

approximately the same time as the sequential version due to contention for the file system. In order to solve this problem, we moved to the Linux cluster. We used an MPI-Shell program written by Bruce Wooley (2001) to distribute files to every machine's file system at the beginning of a run and to clean the file systems at the end of the run. At the beginning of a run, all data files are distributed to every machine. Since every machine has its own file system, the evaluation function of individuals on that machine can read and write files without affecting other evaluations. This allows all machines to concurrently evaluate a subset of the population. Tables 1 and 2 give experimental results for the intrusion detection system using the master-slave model for the GA. It is evident that execution time increases as the population size increases. When the number of processors is small, the speedup and efficiency do not change a great deal as the population size increases. As the population size increases, the speedup increases but does not achieve the ideal. The efficiency decreases because of increased communication overhead as the number of processors increases. The single-population model yields the same solution as the sequential version of the GA.

Table 1 Execution time for master-slave parallel implementation of fuzzy data mining.

Pop. Size	Execution time (mins) (30 generations)					
	Processors					
	1	2	3	4	5	6
35	18.34	9.89	7.11	5.62	4.51	4.06
70	47.27	25.37	18.05	14.47	11.79	10.73
140	87.31	44.54	31.34	24.67	20.27	17.83

Table 2 Speedup and efficiency for master parallel implementation.

Pop. Size	Speedup (Processors)						Efficiency (Processors)					
	1	2	3	4	5	6	1	2	3	4	5	6
35	1	1.9	2.6	3.3	4.1	4.5	1	.93	.86	.82	.82	.76
70	1	1.9	2.6	3.3	4.0	4.4	1	.93	.87	.82	.80	.74
140	1	2.0	2.8	3.6	4.3	4.9	1	.98	.93	.89	.86	.82

A multiple-population model for the fuzzy data mining GA was also implemented and run on the Linux cluster. The subpopulations are on a virtual oriented ring and the best individual from each subpopulation migrates to a neighbor where it replaces the worst individual. Migration is done every two generations. Unlike the master-slave model, the multiple-population explores a different search space than that explored by the sequential model and will find a different solution. Preliminary results indicate that the multiple population model may provide a higher quality solution in a shorter amount of time. Further investigation is required to confirm these initial findings.

CONCLUSIONS AND FUTURE WORK

The genetic algorithm that we have developed for tuning a fuzzy data mining application has a very time-consuming evaluation function since the evaluation of each individual involves a data mining process. In addition, the data mining algorithm must read and write data files every time it is executed. The costly evaluation function makes this genetic algorithm a prime candidate for parallel implementation since the cost of evaluation should dominate the communication costs. We have implemented both single-population master-slave and multiple-population versions of the GA. When the parallel GAs are run on a Sun workstation with a single file system, little speedup is obtained due to contention for the file system. This problem was not encountered with the Linux cluster because each node has its own file system. Since all of the nodes needed exactly the same files for all iterations, the files could be distributed one time only. The near linear speedup achieved with the single-population master-slave GA is comparable to results reported by other researchers. The advantage of the cluster system is that different processes can concurrently access to different file systems and this makes the concurrent evaluation of a subset of the population possible. Our initial results with a multiple population GA indicate that this model may have advantages over the single population model. However, this model is more complex than the single population model and further investigation is required to confirm the initial results and to determine appropriate model parameters such as migration rate, subpopulation size, etc.

ACKNOWLEDGEMENT

This work was partially supported by NSF Grant #9818489.

REFERENCES

- Bridges, Susan M., and Rayford B. Vaughn. 2000. Fuzzy Data Mining And Genetic Algorithms Applied to Intrusion Detection. *Proceedings of the National Information Systems Security Conference (NISSC), October 16-19, 2000, Baltimore, MD.*
- Cantú-Paz, Erick. 1998. A survey of parallel genetic algorithms. *Calculateurs Paralleles*. Vol. 10, No. 2. Paris: Hermes. <http://www-illigal.ge.uiuc.edu/publications.php3> (Accessed 18 May 2000).
- Cantú-Paz, Erick. 1999a. Topologies, migration rates, and multi-population parallel genetic algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*. Edited by Banzhaf, W. et al. San Francisco, CA: Morgan Kaufmann Publishers.
- Kuok, C., A. Fu, and M. Wong. 1998. Mining fuzzy association rules in databases, *SIGMOD Record* 27(1): 41-6.
- Luo, Jianxiong, and Susan M. Bridges. 2000. Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection. *International Journal of Intelligent Systems* 15 (8): 687-703.
- Shi, Fajun. 2000. Genetic algorithms for feature selection in an intrusion detection application. M.S. Thesis, Mississippi State University.
- Wooley, Bruce. 2001. MPI-shell and other software. <http://www.cs.msstate.edu/~bwooley/>.