Data mining with a parallel rule induction system based on gene expression programming

Wagner Rodrigo Weinert

Federal Institute of Education, Science and Technology of Paraná,
R. Antônio Carlos Rodrigues 453, 83215-750 Paranaguá (PR), Brazil
Fax: +55-41-3721-8300
E-mail: wagner.weinert@ifpr.edu.br

Heitor Silvério Lopes*

Federal University of Technology – Paraná,
Av. Sete de Setembro 3165, 80230-901 Curitiba (PR), Brazil
Fax: +55-41-3310-4694
E-mail: hslopes@utfpr.edu.br
*Corresponding author

Abstract: A parallel rule induction system based on gene expression programming (GEP) is reported in this paper. The system was developed for data classification. The parallel processing environment was implemented on a cluster using a message-passing interface. A master-slave GEP was implemented according to the Michigan approach for representing a solution for a classification problem. A multiple master-slave system (islands) was implemented in order to observe the co-evolution effect. Experiments were done with ten datasets, and algorithms were systematically compared with C4.5. Results were analysed from the point of view of a multi-objective problem, taking into account both predictive accuracy and comprehensibility of induced rules. Overall results indicate that the proposed system achieves better predictive accuracy with shorter rules, when compared with C4.5.

Keywords: evolutionary computation; EC; gene expression programming; GEP; data mining.

Reference to this paper should be made as follows: Weinert, W.R. and Lopes, H.S. (xxxx) 'Data mining with a parallel rule induction system based on gene expression programming', *Int. J. Innovative Computing and Applications*, Vol. X, No. Y, pp.000–000.

Biographical notes: Wagner Rodrigo Weinert received his BSc in Computer Science from Ponta Grossa State University, and his MSc and PhD in Computer Science from the Federal University of Technology – Paraná, Curitiba, Brazil. Currently, he is an Assistant Professor at the Federal Institute of Education Science and Technology of Paraná. His areas of interest are evolutionary computation and data mining.

Heitor Silvério Lopes received his degree in Electrical Engineering (1984) and his MSc in Biomedical Engineering (1990) from the Federal University of Technology – Paraná – UTFPR. He received his PhD in Electrical Engineering (1996) from the Federal University of Santa Catarina. He is presently working as an Associate Professor in the Department of Electronics of UTFPR. He is the Head of the Bioinformatics Laboratory of UTFPR since its foundation in 1997. He is also a Researcher of the Brazilian National Research Council, and his current research interests are: evolutionary computation, data mining and bioinformatics.

1 Introduction

Nowadays, the amount of data stored in databases tends to increase exponentially. Frequently, many important information, implicit to the data, is missed, thus precluding to become high-level knowledge. Such knowledge may be very useful for decision-making in many important areas, such as commerce, science, environment, medicine, etc.

Computer science has contributed with different techniques for extracting relevant knowledge from databases. This area, known as data mining (Freitas, 2002; Witten and Frank, 2005), is interdisciplinary by nature, in which statistics, artificial intelligence, database, evolutionary computation (EC), and other areas, can provide useful tools.

There are many different tasks in data mining, such as clustering, summarisation, dependence modelling and classification. This work is motivated by the increasing amount of data available in organisations and the emerging necessity to explore and understand such data, in such a way to discover interesting and useful knowledge. It is important

Copyright © 200X Inderscience Enterprises Ltd.

to notice that most data mining algorithms proposed in the literature (especially rule-induction methods) take into account only the predictive accuracy of the classifiers. However, when the resulting solutions have to be used and interpreted by a human, the comprehensibility of rules is a very important issue (Freitas, 2002; Johansson et al., 2004).

Although many studies have been carried out about data mining methodologies (Rosset et al., 2010), this work focuses the classification task. In this task, it is aimed to find rules (a logical composition of attributes and values) capable of classifying instances of a dataset according predefined classes. The task of inducing such rules can become excessively complex for deterministic algorithms as the number of instances and the number of attributes per instance in the database increases. Consequently, for real world problems, the computational complexity can grow fast, making traditional algorithms to become inefficient. Therefore, meta-heuristic methods, such as those from EC, can play an important role in the rule induction process.

EC techniques have been applied successfully to many practical problems of natural sciences, engineering and computer science. EC includes methods inspired on the evolution of living beings, based on the Darwinian principle of survival of the fittest. Amongst the several EC methods, it is worth to mention genetic algorithms (GAs) (Goldberg, 1989), genetic programming (GP) (Koza, 1992; Poli et al., 2008) and gene expression programming (GEP) (Ferreira, 2006; Li et al., 2005; Wilson, 2008; Zakaria et al., 2010). These methods evolve a population of individuals (in our case, representing classification rules) for a given number of generations. At each generation, a set of genetic operators are applied to a selected number of individuals so as to generate new solutions. Hopefully, the average quality of individuals, measured by a fitness function, tends to increase at each generation. The evolutionary process is stopped when a satisfactory solution is found or a maximum number of iterations are reached.

GEP is a relatively recent EC method, and it has been poorly explored in the recent literature, when compared with GAs and GP. Some few works reported improvements in GEP aiming at accelerating its convergence and reducing the processing time. For instance, Du et al. (2008) introduced the concept of co-evolution using multiple populations, in a distributed environment, using an asynchronous island model. This work has some similarities with the work reported here, however, it does not take into account the comprehensibility of the obtained solutions. Another related work is from Park et al. (2008), who proposed a client-server architecture, where the clients runs instances of a GEP algorithm, and the server runs a GA that is aimed at finding good running parameters for the GEPs. A migration policy controls the exchange of genetic material between clients, and this is somewhat similar to our work.

Frequently, the computation of the fitness function of an EC method is computationally expensive. In the process of rule induction – see Section 2.2, the computation of the fitness function consists in the evaluation of set of instances (training set) as many times as the product of the number of individuals by the number of generations. Also, the training set can have a large number of instances and, each of them can have a large number of attributes. Consequently, rule induction with using EC requires intensive computation, and so, a parallel processing architecture is necessary to reduce the overall processing time (Weinert and Lopes, 2010).

Therefore, the general objective of this work is to develop a rule induction system for mining classification rules. This system is based on a parallel GEP approach and tackles the problem from a multi-objective optimisation point of view. This paper is organised as follows. Section 2 presents some relevant theoretical issues. The next section describes the methodology for developing the parallel version of a GEP-based rule induction system. Section 4 describes how the experiments were done and Section 5 shows the results of the application of the system to ten databases. Finally, Section 6 presents conclusions and points out future works.

2 Theoretical foundations

2.1 Gene expression programming

GEP (Ferreira, 2006) is an EC approach, considered an extension of GP, hybridised with elements of GAs. The main difference between GA, GP and GEP is in the way solutions (individuals) are represented. In GAs, usually, they are represented as a fixed-sized linear string of bits, while in GP they are non-linear entities in the form of a variable-sized tree. On the other hand, in GEP, individuals are encoded as linear chains of fixed size (genome), as in GAs, but later expressed as trees of arbitrary size and shape (known as expression trees – ETs), as in GP.

After the basic GEP, some improvements were proposed in two specific domains: EGIPSYS (Lopes and Weinert, 2004), for symbolic regression; and GEPCLASS (Weinert and Lopes, 2006), for data classification. In GEPCLASS, an individual is composed by a single chromosome which, in turn, is composed by a number of genes. Each gene is divided into two parts: head and tail. The head is filled by elements belonging to the functions set (i.e., logical and relational functions: *AND*, *OR*, *NOT*, =, ≠, > and <), and tail can elements belonging to the functions set as well as to the terminals set. Terminals include the attributes (and particular values) of the problem in hand.

GEP employs the concept of open reading frame (ORF) in the process of encoding the information in the genes. According to this biological concept, not all genetic material present in the genome is actually used. An ORF, or encoding region of a gene, can activate or deactivate genetic material during the evolutionary process, thus, making some elements of the genes to be considered or not in the construction of a solution. Thanks to this approach, a chromosome is transcribed into an ET of variable size. The transcription rules follow the patterns expressed by the Karva language (Ferreira, 2006), such that each gene is transcribe into a sub-tree. Next, all sub-trees are joined

together into a single ET by a linking function (usually, AND or OR). This ET represents the candidate solution for the problem. Figure 1 shows a full example of the encoding and ET in GEPCLASS. This tree is evaluated by a fitness function (see below) over a set of instances of the problem, thus computing the quality of the individual.

Next, individuals are selected for reproduction according to their quality. The most usual methods used in GEP are: the roulette wheel (Goldberg, 1989) and the stochastic tournament (Koza, 1992). Once individuals are selected, they undergo the action of genetic operators to create new individuals. The genetic operators defined in GEP are: cloning, mutation, recombination, IS transposition, RIS transposition and genic transposition. However, due to space limitations, the detailed description of these operators is omitted, but can be found in Weinert and Lopes (2006).

Since the problem dealt in this work is data classification, when classifying a given instance of the problem by a rule, four outcomes are possible:

- *tp*: true positive – The rule predicts that the instance belongs to a given class, and it really does.
- *fp*: false positive – The rule predicts that the instance belongs to a given class, but it actually does not.
- *tn*: true negative – The rule predicts that the instance does not belong to a given class, and it really does not.
- *fn*: false negative – The rule predicts that the instance does not belong to a given class, but it actually does.

The outcomes above are the basis for the fitness function proposed by Lopes et al. (1997). The goal of the fitness function is to maximise two measures frequently used in classification tasks, namely, specificity ($Sp = tn / (tn + fp)$) and sensitivity ($Se = tp / (tp + fn)$), by taking their product, as follows:

$$fitness = Sp.Se \tag{1}$$

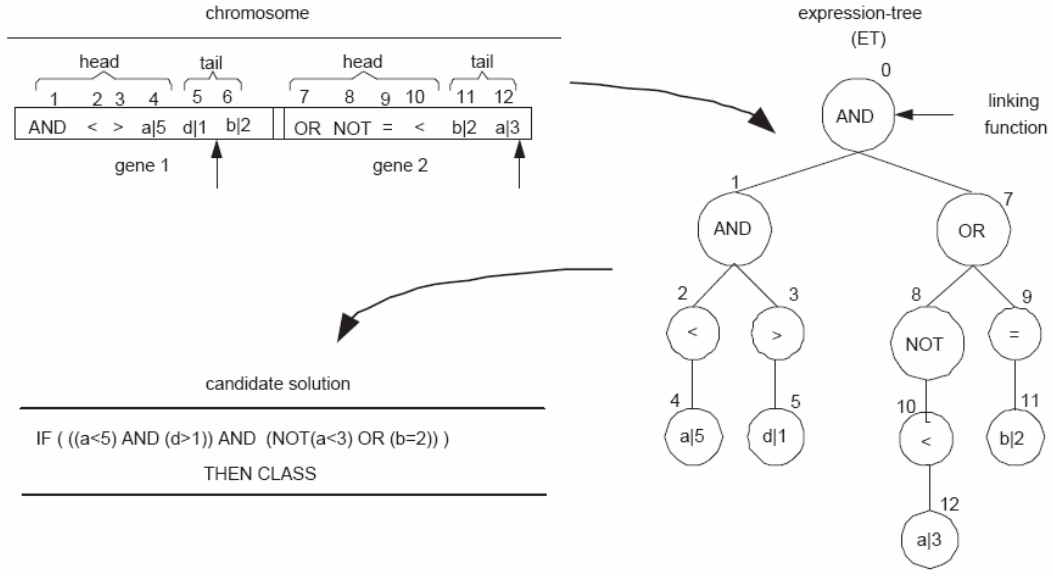
2.2 The rule induction problem

The rule induction problem can be defined as follows: given a set of instances (training set, with known classification), find a set of rules composed by a logical combination of attributes pertaining to the dataset, such that unseen instances can be correctly classified (Flach and Lavrac, 2003). In general, classification rules are expressed in the form IF A THEN C. The antecedent A is a composition of the attributes of the problem with logical and relational functions, and the consequent C represents the class to which the instance should belong to.

More specifically, in this work, the antecedent of rules are defined as *t*-uplets in the form A_i *Op* V_{ij}, where A_i is the *i*th attribute, *Op* is a relational operator (=, ≠, > or <), and V_{ij} is the *j*th value pertaining to the domain of attribute A_i. Logical operators (AND, OR and NOT) are used to combine *t*-uplets forming a complex condition.

In this work, the encoding of individuals of GEP follows the Michigan approach (Freitas, 2002). According this approach, an individual represents a single rule in the classification system. Therefore, the set of all sub-trees expressed by a chromosome represents a single (and complex) rule for a given class.

Figure 1 An example of a genome, its ET, and the corresponding rule



3 Methodology

3.1 The parallel processing environment

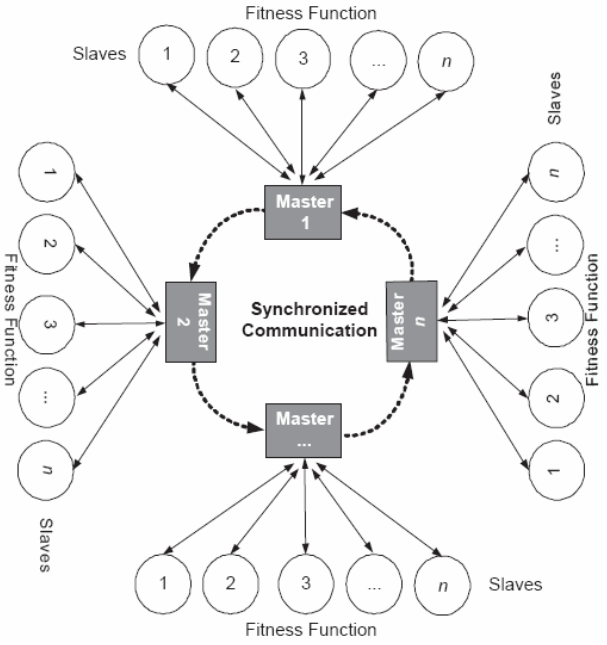
The system was developed in C programming language for the Linux Suse 10.0 operating system. Parallelism was implemented using the MPICH 2.0 Library (Gropp et al., 1999)[1] that provides the message-passing interface (MPI) support. This library creates a virtual environment composed by several elements. Each element corresponds to a core processor of a multi-core workstation networked as a cluster. A set of low-level functions for communication and synchronism allows an efficient way to develop parallel applications. Processors in the cluster do not share memory, since all interaction between them is accomplished by means of messages through the MPI environment (Snir et al., 1997). Such parallel environment was developed especially for using in a cluster of networked computers, homogeneous or not. For all experiments reported in Section 4, we used a cluster of 31 workstations (with Pentium ©Quadcore EM64T processor, running at 2.4 GHz, and 2 GB of RAM), delivering 124 homogeneous processing cores. Although not essential, the homogeneity of processors in the MPI environment is important, because it facilitates load balance, an important issue in parallel computing applications.

All computers are interconnected by an exclusive gigabit Ethernet switch, and only one has an external access (gateway), so as to minimise the traffic within the network. Within the MPI environment, all tasks are subdivided into processes. Each process is run in a specific processing core. Although more than a process can be run in a core, this is avoided to allow a better load balance in the cluster.

In our implementation, a logic parallel architecture can be dynamically dimensioned to deal with a specific problem. Such architecture is composed by a master process that controls slave processes. The set of a master and *n* slave processes defines a processing island. The developed system is also able to work with multiple islands that cooperates each other for the solution of a classification problem, as shown in Figure 2.

The master process is responsible for running the enhanced GEP algorithm (Weinert and Lopes, 2006) and control all the communication with its slave processes. Every slave process runs exactly the same task, in this case, the computation of the fitness function. In a multi-island environment, master processes become collaborative each other, by managing the transmission and reception of messages between them. Messages between master processes are migrant individuals between the islands. The objective here is to enhance the quality of solutions evolved in a given island by introducing new genetic material from an external source. This procedure may induce the co-evolution phenomenon (Freitas, 2002), which is highly beneficial to improve the overall quality of solutions. Co-evolution emerges from a symbiotic process where symbiosis defined as relationships between individuals of different populations, and the ability of an individual directly affects fitness of another one.

Figure 2 Architecture of the parallel processing system



3.2 Parallel rule induction with gene expression programming

The developed system was named parallel rule induction with gene expression programming (PRIGEP). Initially, the logical structure of PRIGEPs computing environment is defined by two parameters: the number of master processes (islands) and the number of slave processes subordinated to each master.

In the experiments reported in the next section, two versions were tested: PRIGEP-MS (one master with ten slaves), and PRIGEP-Islands (five masters, each one with ten slaves). In PRIGEP-Islands, aiming at enabling co-evolution, each island is logically connected to another island in a ring topology.

The communication between master processes always takes place unidirectionally and synchronously under the coordination of one of the master processes. Since the MPI environment does not provide any load balancing mechanism, it is entirely responsibility of the developer. Since the processing cores are exactly the same, load balancing is accomplished by equalising the number of individuals processed by each slave at time, as well as the number of slaves per island. Therefore, once the GEP algorithm generates individuals of a new generation, they are evenly divided among the corresponding slaves of the island, so as to evaluate the fitness function. Due to the inherent differences between individuals generated by GEP, a small difference in the processing time can appear by the end of a batch processing of a slave. Since the communication is synchronous, in the next communication cycle all differences disappear, thus automatically balancing again the system.

In each island, the initial population is randomly created, using the elements of the functions set and

terminals drawn from the attributes of the dataset. The rules for filling head and tail are the same as used before (Weinert and Lopes, 2006). The random number generator used in our implementation was the Mersenne Twister (Matsumoto and Nishimura, 1998), known as one of the most efficient for evolutionary algorithms.

Creating the initial population, as well as determining the ORFs is responsibility of the master process. Once this is done, individuals are distributed to slaves by means of a MPI function. Slaves compute the fitness of individuals and wait for a request of the master to send results. Once the master has collected all results from slaves, the evolutionary process inherent to the GEP algorithm takes place. That is, individuals are selected based on their fitness and undergo the action of the genetic operators. The stop criterion is checked (maximum number of generations) and, if not met, the cycle is repeated, sending individuals to the slaves.

4 Computational experiments

PRIGEP was tested with classification problems using ten publicly available datasets from the machine learning repository (Frank and Asunción, 2010). The datasets were randomly chosen from the repository. The only criterion was no missing data in any instance. Table 1 shows the most relevant information about the features of the datasets. In this table, it is shown the number of classes (2..16), number of attributes (4..16) and number of instances (150..20,000). It is interesting to observe that the features of these datasets cover a large range of possibilities, and so, they are useful to test the proposed algorithm in different situations.

Table 1 Features of the ten datasets used in the experiments

Database	*# clas.*	*# attr.*	*# inst.*
Abalone	3	8	4,177
Balance scale	3	4	625
Car evaluation	4	6	1,728
Haberman	2	3	306
Blood transfusion	2	4	748
Ljubljana breast cancer	2	9	277
Wisconsin breast cancer	2	9	683
Iris	3	4	150
Nursery	4	8	12,958
Letter recognition	16	16	20,000

Since the number of classes for each dataset is different, for each one, a hierarchical classification model is created. For instance, the *Abalone* dataset has three classes with 4,177 instances, with 1,407, 1,323 and 1,447 instances for classes 1, 2 and 3, respectively. The hierarchical classification consists of, first inducing a classification rule for class 1 (positive class), considering classes 2 and 3 as negative examples. Next, all instances of class 1 are eliminated from the dataset and a rule for class 2 is induced, considering class 3 as negative examples. The last class is considered as 'default' and no rule is induced for it. For each hierarchical level, three experiments are done aiming to evaluate the performance of the induced rules.

The first experiment is accomplished by the well-known C4.5 decision-tree induction algorithm (Quinlan, 1993). This algorithm is considered the baseline for data classification and it is frequently used for comparison purposes in data mining (Witten and Frank, 2005).

The second experiment encompasses the use of the master-slave (MS) approach (a single processing island) of the GEP algorithm for rule induction (named PRIGEP-MS). The third experiment (PRIGEP-Islands) expands the number of islands to five, and aims at verifying the possible benefits of co-evolution. Recall that at each island there is the same PRIGEP-MS.

Although parallel processing models suggest the comparison of performance regarding processing time, this was not the focus of our experiments because C4.5 is deterministic and much faster than any heuristic algorithm, evolutionary or not. In the three experiments, reported below, performance was measured regarding the predictive accuracy that is the product $Se \times Sp$. This product returns values in the range [0..1], where 1 is the best possible performance (100% of correct classifications). We also evaluated a very important issue in data mining: the comprehensibility of induced rules (Freitas, 2002). Considering that the knowledge extracted from the data mining process is to be used by humans in a decision-making process, it is essential that the user can understand it. Therefore, the complexity of a rule is measured by the number of nodes of the corresponding ET. The smaller the number of nodes, the easier to comprehend.

All experiments were done using the cross-validation procedure with five folds (Hand, 1997; Witten and Frank, 2005). The same proportion of classes as the original dataset was maintained for each fold, not only for the training subset, but also for the evaluation subsets. This was done to avoid any classification bias in the process.

Since there is no conventional procedure for adjusting parameters of the GEP algorithm for classification problems, the control parameters of PRIGEP were adjusted according to the default values stated in the current literature. Independently of the number of islands, all algorithms (GEPs) use the same parameters (Ferreira, 2006; Lopes and Weinert, 2004; Weinert and Lopes, 2006), as follows: 50 individuals per island, 50 generations, three genes per chromosome, 15 elements in the head of the gene, stochastic tournament using 10% of population size as selection method. The function set: {AND, OR, NOT}, and linking function AND. The probabilities of application of operators were: mutation (2%), recombination (80%), IS, RIS and genic transpositions (70%). The parallel architecture included five islands with ten slave processes per island, interconnected by a unidirectional ring. The migration policy was defined as one individual from each island migrates to the next neighbour island, every five generations. The emigrant is selected by tournament

selection and substitutes the worst individual in the arriving island.

5 Results and discussion

For the ten above mentioned datasets, a total of 17 comparisons were done between the three algorithms: C4.5, PRIGEP-MS and PRIGEP-Islands. Each comparison reflects a specific hierarchical level of the data classification.

Some issues raised from the analysis of the 17 comparisons. Considering only the quality measure, in 12 out of 17 comparisons the PRIGEP-Islands model was the most effective, followed by C4.5, which was the best performing algorithm in four comparisons, and PRIGEP-MS that won in only one comparison. On the other hand, considering only the complexity measure, in nine out of 17 comparisons, C4.5 presented solutions with the smallest complexity, while PRIGEP-Islands and PRIGEP-MS won in 6 and 2 comparisons, respectively. When combining both measures, the following scenario was found: PRIGEP-Islands showed the best trade-off in five comparisons, and C4.5 in three comparisons. The PRIGEP-MS model did not fit properly this scenario. The remaining nine comparisons need a more careful analysis, as follows.

In the *Abalone* database, 2nd hierarchical level, C4.5 algorithm achieved the best quality. However, this measure exceeds only 0.007 from that of PRIGEP-Islands. Moreover, the solution presented by C4.5 is approximately five times more complex than that presented by PRIGEP-Islands. Therefore, in this case, the simplest solution can be considered more appropriate for the problem.

In the *balance scale* database, there are two cases to be checked. At 1st hierarchical level, it is evident that no method was able to reach a satisfactory solution. Probably, this is due to the imbalance of samples between the positive and negative classes. The negative class has 11.75 times more samples than the positive class. At 2nd hierarchical level, despite the solution provided by PRIGEP-Islands has nine nodes more than the one found by PRIGEP-MS, it has approximately twice the quality of the latter. Therefore, it is fair to consider here PRIGEP-Islands as the best performing algorithm.

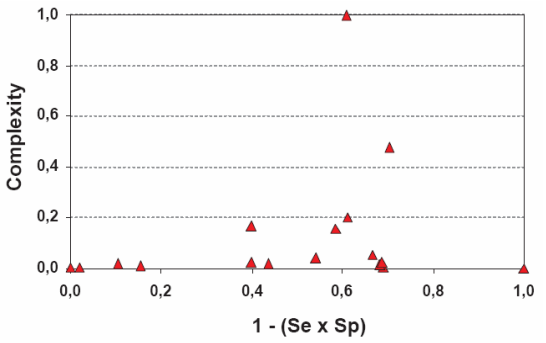
In the *car evaluation* database, two particular cases raised. At 1st hierarchical level, the best quality was obtained by PRIGEP-Islands, and the smallest complexity obtained by PRIGEP-MS. Since the difference is only two nodes, regarding complexity, the algorithm that achieved the best quality was chosen. At 3rd hierarchical level, we should consider another issue when choosing between PRIGEP-Islands and C4.5, since the first won in quality and the second in complexity. If the resulting rules will be manipulated by a computer, we should choose PRIGEP-Islands' solution; otherwise, if a human will interpret the rules, we should choose C4.5's solution, since complexity is an important issue.

In the *Haberman* database, at 1st hierarchical level, we faced the same situation previously mentioned at 3rd level of *car evaluation* database. However, at this time the choice concerns C4.5 (regarding simplicity) or PRIGEP-MS (regarding quality).

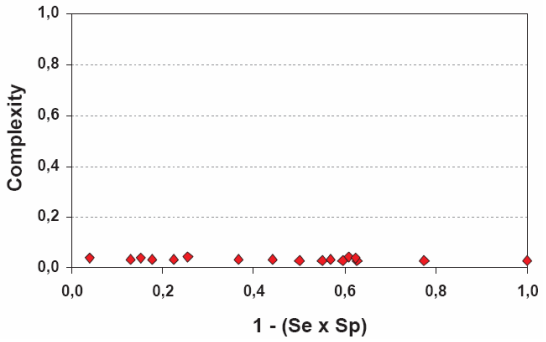
In the *blood transfusion* database, 1st hierarchical level, PRIGEP-Islands obtained a solution with quality 30% above C4.5. In this case, the difference in complexity, 12 nodes, does not justify the choice of C4.5.

In the *Wisconsin Breast Cancer* database, 1st hierarchical level, the quality of PRIGEP-Islands and C4.5 were equivalent. However, C4.5 obtained the simplest solution, and this was the choice.

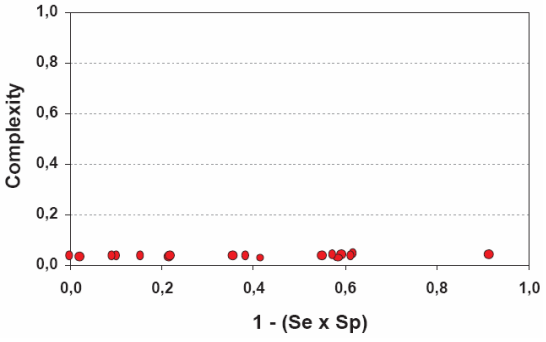
Figure 3 Pareto plot: (a) C4.5, (b) PRIGEP-MS and (c) PRIGEP-Islands (see online version for colours)



(a)



(b)



(c)

In the *nursery* database, 3rd hierarchical level, PRIGEP-Islands obtained a solution which quality was 61% better than that of C4.5. In such situation, the difference in

complexity between algorithms does not justify the choice for C4.5.

Figure 3 summarises the performance of the algorithms for all the datasets using Pareto plots. In these plots, the *x* axis is the complement of the fitness function (that is, $1 - (Se \times Sp)$), and the *y* axis is the normalised number of nodes (complexity of the induced rules), obtained by dividing the actual number of nodes by the largest value obtained in the experiments. Pareto plots are useful for the analysis of multi-objective problems, in which two or more contradictory criterions (or objectives) need to be satisfied at the same time. Recall that in a Pareto plot, the best solutions are those closest to the origin, meaning that they tend to satisfy both criterions (represented by the x and y axes).

In this figure, it is shown that the C4.5 algorithm achieved the worst performance, not only in the quality of solutions (average = 0.448), but also, in the complexity of the induced rules (average = 0.131). PRIGEP-MS improved significantly the results, in quality (average = 0.449), and mainly regarding the complexity (average = 0.035). Finally, PRIGEP-Islands achieved around the same level of complexity of PRIGEP-MS (average = 0.034), but improved the quality further (average = 0.377).

A statistical analysis of the results revealed that there is no clear correlation between the number of instances, number of attributes, and number of classes with the quality or complexity of induced rules, for all three algorithms. The only exception was a direct relationship between the number of instances with the complexity of induced rules for C4.5 ($R^2 = 0.97$). This is explained by the nature of the algorithm (Quinlan, 1993). Overall, this means that PRIGEP (in both versions) is not biased by any feature of the datasets under analysis.

6 Conclusions and future works

This work reported the development of a parallel rule induction system based on the gene-expression programming paradigm. The algorithm was applied to ten datasets and results were compared with C4.5. Results were analysed from the point of view of a multi-objective problem that is, taking into account quality and complexity of rules at the same time.

PRIGEP, in both versions (MS and island), obtained rules with higher predictive accuracy than the baseline C4.5 algorithm, for most cases. Considering that C4.5 is a well-established deterministic algorithm, and the proposed GEP-based system is experimental, results are very satisfactory in general. In special, the effect of co-evolution of PRIGEP-Islands was very beneficial for improving the predictive quality. Generally speaking, considering the average performance, PRIGEP-MS achieved better predictive accuracy with less complex rules when compared with C4.5. Also, PRIGEP-Islands extended the quality and comprehensibility of results achieved by PRIGEP-MS. A remarkable feature of PRIGEP is the comprehensibility of induced rules, making it a more 'human-friendly' algorithm than other regular rule-induction algorithms.

Both versions of PRIGEP were shown to be independent of the features of datasets (number of instances, classes, and attributes). This fact suggests the generality of the method, and future research will investigate in depth this issue.

The PRIGEP-Islands model was tested with a fixed topology and migration policy. Current literature indicates that both issues are relevant for parallel evolutionary algorithms, and this is will be also explored in future work. Similarly, it is possible that the running parameters of the GEP algorithm are not the optimal for the rule-induction problem. The automatic adjustment of these parameters would make the algorithm easier to use and possibly lead to even better results. Therefore, self-adjustment of parameters will be included in the next version of PRIGEP.

PRIGEP is intended to be applied to other complex problems of rule induction, especially in the domain of bioinformatics, such as spreading of diseases, simulation of tumour growth, modelling enzymatic reactions, etc.

References

Du, X., Ding, L. and Jia, L. (2008) 'Asynchronous distributed parallel gene expression programming based on estimation of distribution algorithm', in *Proceedings of the 4th International Conference on Na tural Computation*, pp.433–437, IEEE Computer Society, Jinan, China.

Ferreira, C. (2006) *Gene Expression Programming: Mathematical Modeling by an A rtificial Intelligence*, 2nd ed., Springer-Verlag, Berlin.

Flach, P. and Lavrac, N. (2003) 'Rule induction', in Berthold, M. and Hand, D. (Eds.): *Intelligent Data A nalysis: an Introduction*, 2nd ed., Chapter 7, pp.229–267, Springer, Heidelberg.

Frank, A. and Asunción, A. (2010) 'UCI machine learning repository', available at http://archive.ics.uci.edu/ml.

Freitas, A. (2002) *Data Mining and Knowledge Discovery with Evolutionary Algorithms*, Springer-Verlag, Berlin.

Goldberg, D. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, USA.

Gropp, W., Lusk, E. and Thakur, R. (1999) *Using MPI-2: Advanced Features of the Message-Passing Interface*, MIT Press, Cambridge, USA.

Hand, D. (1997) *Construction and Assessment of Classification Rules*, John-Wiley & Sons Ltd., Chichester, England.

Johansson, U., Niklasson, L. and König, R. (2004) 'Accuracy vs. comprehensibility in data mining models', in Svensson, P. and Schubert, J. (Eds.): *Proceedings of th e Seventh International Conference on Information Fusion*, Vol. 1, pp.295–300, International Society of Information Fusion, Mountain View, CA.

Koza, J. (1992) *Genetic Programming: on the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, USA.

Li, X., Zhou, C., Xiao, W. and Nelson, P.C. (2005) 'Prefix gene expression programming', in F. Rothlauf et al. (Eds.): *Proceedings of Genetic and Evolutionary Computation Conference GECCO*, pp.25–29, USA, Washington DC.

Lopes, H., Coutinho, M. and de Lima, W. (1997) 'An evolutionary approach to simulate cognitive feedback learning in medical domain', in Sanchez, E., Shibata, T. and Zadeh, L. (Eds.): *Genetic Algorithms and Fuzzy Logic Systems*, pp.193–207, World Scientific, Singapore.

Lopes, H.S. and Weinert, W.R. (2004) 'EGIPSYS: an enhanced gene expression programming approach for symbolic regression problems', *International Journal of Applied Mathematics and Computer Science*, Vol. 14, No. 3, pp.375–384.

Matsumoto, M. and Nishimura, T. (1998) 'Mersenne twister: a 623-dimensionally equidistributed uniform pseudorandom number generator', *ACM Transactions on Modeling and Computer Simulation*, Vol. 8, No. 1, pp.3–30.

Park, H., Grings, A., dos Santos, M.V. and Soares, A.S. (2008) 'Parallel hybrid evolutionary computation: automatic tunning of parameters for parallel gene expression', *Applied Mathematics and Computation*, Vol. 201, Nos. 1–2, pp.108–120.

Poli, R., Langdon, W.B. and McPhee, N.F. (2008) *A Field Guide to Genetic Programming*, Lulu Enterprises, Raleigh, USA.

Quinlan, J.R. (1993) *C4.5: Programs fo r Machine Learning*, Morgan Kaufmann, San Mateo, USA.

Rosset, S., Perlich, C., Świrszcz, G., Melville, P. and Liu, Y. (2010) 'Medical data mining: insights from winning two competitions', *Data Mining and Knowledge Discovery*, Vol. 20, pp.439–468.

Snir, M., Otto, S., Huss-Lederman, S., Walker, D. and Dongarra, J. (1997) *MPI: The Complete Reference*, 2nd ed., MIT Press, Cambridge.

Weinert, W. and Lopes, H. (2006) 'GEPCLASS: a classification rule discovery tool using gene expression programming', *Lecture Notes in Computer Science*, Vol. 4093, pp.871–880.

Weinert, W. and Lopes, H. (2010) 'Evaluation of dynamic behavior forecasting parameters in the process of transition rule induction of unidimensional cellular automata', *Biosystems*, Vol. 99, No. 1, pp.6–16.

Wilson, S.W. (2008) 'Classifier conditions using gene expression programming', in Bacardit, J., Bernadó-Mansilla, E., Butz, M.V., Kovacs, T., Llorà, X. and Takadama, K. (Eds.): *Learning Classifier Systems*, pp.206–217, Springer-Verlag, Berlin, Heidelberg.

Witten, I. and Frank, E. (2005) *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation*, 2nd ed., Morgan Kaufmann, San Francisco.

Zakaria, N.A., Azamathulla, H.M., Chang, C.K. and Ghani, A.A. (2010) 'Gene expression programming for total bed material load estimation – a case study', *Science of the Total Environment*, Vol. 408, pp.5078–5085.

Notes

1 Available at http://www.mcs.anl.gov/research/projects/mpich2/.