BIOINFORMATICS

Vol. 23 ISMB/ECCB 2007, pages i167–i174
doi:10.1093/bioinformatics/btm205

GPDTI: A Genetic Programming Decision Tree Induction method to find epistatic effects in common complex diseases

Jesús K. Estrada-Gil[1,2,*] Juan C. Fernández-López[1,2], Enrique Hernández-Lemus[2], Irma Silva-Zolezzi[3], Alfredo Hidalgo-Miranda[3], Gerardo Jiménez-Sánchez[3] and Edgar E. Vallejo-Clemente[1]

[1]Computer Science Department, Instituto Technológico y de Estudios Superiores de Monterrey Campus Estado de Mexico, Mexico, [2]Department of Computational Genomics and [3]Department of Basic Research; Instituto Nacional de Medicina Genómica, Mexico

ABSTRACT

Motivation: The identification of risk-associated genetic variants in common diseases remains a challenge to the biomedical research community. It has been suggested that common statistical approaches that exclusively measure main effects are often unable to detect interactions between some of these variants. Detecting and interpreting interactions is a challenging open problem from the statistical and computational perspectives. Methods in computing science may improve our understanding on the mechanisms of genetic disease by detecting interactions even in the presence of very low heritabilities.

Results: We have implemented a method using Genetic Programming that is able to induce a Decision Tree to detect interactions in genetic variants. This method has a cross-validation strategy for estimating classification and prediction errors and tests for consistencies in the results. To have better estimates, a new consistency measure that takes into account interactions and can be used in a genetic programming environment is proposed. This method detected five different interaction models with heritabilities as low as 0.008 and with prediction errors similar to the generated errors.

Availability: Information on the generated data sets and executable code is available upon request.

Contact: jestrada@inmegen.gob.mx

1 INTRODUCTION

Common complex diseases and traits are characterized by environmental and genetic components, and thus, do not follow a simple Mendelian behavior on its heritability.[1] It has been shown that interactions between two or more genes, or genes and environment interactions can be associated with a predisposition to develop a disease. Traditional methods to find genetic regions linked to a particular phenotype have worked well in models that have a monogenic Mendelian pattern, however, it is known that these procedures have reduced power in the characterization of common complex diseases. A promising strategy to identify genes of modest effect in complex diseases is the association study (Risch and Merikangas, 1996).

The most common class of association study design is the case-control study, in which differences of allele frequency of a genetic variant are compared using classic statistical methods such as $\chi 2$ tests and logistic regression. These methods have little power to detect a special form of interactions known as epistasis (Frankel and Schork, 1996). Epistasis occurs when the combined effect of two or more genes within a phenotype could not have been predicted as the sum of their separate effects. The main effect of a certain locus might be virtually undetectable, because its parts may have an opposite effect (Frankel and Schork, 1996). For example, in epistatic interactions between two loci associated with disease, each with three genotypes, the nine genotype pairs might each be associated with a certain penetrance that is, the probability that the genotype pair leads to disease. From these penetrances and the genotype frequencies, marginal penetrances (MP) might be computed that is, penetrances that are associated with the genotypes at one of the two loci (Hoh and Ott, 2003). A well-characterized model for epistasis, in which disease risk is dependent on whether two deleterious alleles and two normal alleles are present is shown in Table 1. This model represents a two-locus interaction with penetrance functions $P(D|AAbb) = 1.0$, $P(D|AaBb) = 0.5$, $P(D|aaBB) = 1.0$ and $P(D|others) = 0$, where D is disease and A, a, B and b represents the alleles for the loci. In this model MP is 0.25 for all possible genotypes, thus a classical association study measuring the main effect of each genotype one at a time would not see any evidence for association. This is not the only possible model for epistasis with similar MP, a variety of biallelic two-locus possible models have been enumerated (Li and Reich, 2000).

Examination of all two-way interactions in a genome-wide association study including 30 000 SNPs would require 450 million tests; and examination of all three-way interactions would require 4.5 trillion tests (Culverhouse *et al.*, 2002), raising concerns about spurious statistically significant false-positive findings. Machine-learning approaches have

*To whom correspondence should be addressed.

[1]In genetics, heritability is defined as the proportion of the total variation of a trait that is attributable to the average effects of genes.

Table 1. Epistatic model for two biallelic markers with equal marginal penetrance (MP) for all genotypes

	AA(.25)	Aa(.50)	aa(.25)	MP
BB(.25)	0.0	0.0	1.0	0.25
Bb(.50)	0.0	0.5	0.0	0.25
bb(.25)	1.0	0.0	0.0	0.25
MP	0.25	0.25	0.25	

been proposed to address the statistical and computational difficulties of this problem (Carlborg *et al*., 2000; Hahn *et al*., 2003; Ljungberg *et al*., 2004; Pociot *et al*., 2004; Ritchie *et al*., 2001, 2003) and have been useful to detect interactions in real data sets of different common diseases (Brassat *et al*., 2006; Hsieh *et al*., 2006; Manuguerra *et al*., 2007; Motsinger *et al*., 2007; Ritchie *et al*., 2001). However, none of these studies has completely succeeded in describing and quantifying genetic association in the context of common complex diseases. In this article we propose a novel method using Genetic Programming to induce a Decision Tree capable of detecting association even in the case of perfect epistasis.

2 APPROACH

Genetic Programming (GP) is an automated computational discovery tool that is inspired by Darwinian evolution and natural selection (Koza, 1991, 1992; Koza and Rice, 1991). Genetic Programming starts with a stem of thousands of randomly created computer programs. This population of programs is progressively selected over a series of generations. After proper training, GP generates a computer program capable of resolving a task such as classification. This learning capacity can be used to generate models that fit and explain a training set. Proper validation of the GP generated result can be done using techniques such as cross-validation consistency.

GP computer programs are often represented as symbolic expressions (*S-expressions*) using the LISP functional programming language. S-expressions represent nested compositions of operators of the language using a prefix notation. Consider the following boolean S-expression borrowed from Koza, (1992):

(OR (AND (NOT D1) (NOT D0)) (AND D1 D0))

An alternative representation of S-expressions is given by expression trees. The aforementioned program is represented as an expression tree as shown in Figure 1. GP often operates on computer programs represented as expression trees.

GP emulates the natural evolutionary process by means of the genetic operators: reproduction, recombination and mutation.

The reproduction operator selects a program according to a fitness measure which indicates the ability of the program in solving a particular problem. The selected program is preserved in the next generation of programs.

The recombination operator creates new offspring programs from selected parent programs. Offspring programs are constructed from program fragments taken from their parents. For example, consider the parent programs of Figure 2.

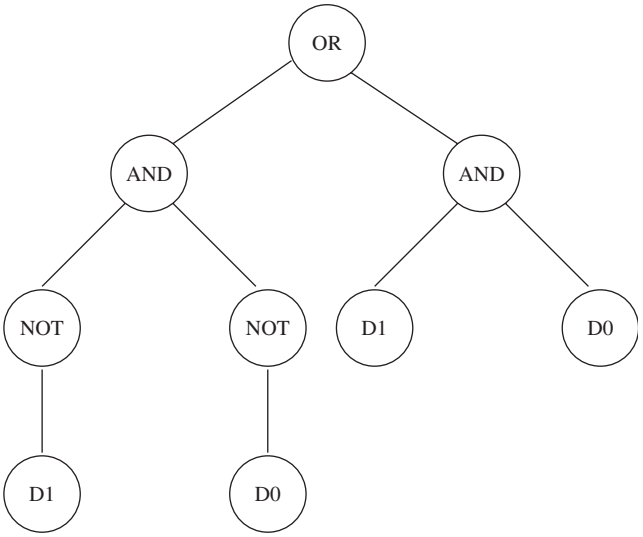


Fig. 1. Expression tree example.

A recombination point is randomly selected in the corresponding expression trees. Figure 3 shows the program fragments obtained from this step.

These fragments are exchanged in the parent trees to obtain the offspring programs as shown in Figure 4.

After the processes of reproduction and recombination, new versions of the offspring programs can be created by the mutation operator. For example, consider the program of Figure 5.

A mutation point is randomly selected in the corresponding expression tree. The program fragment obtained from this step is replaced by the randomly generated program fragment of Figure 6.

Figure 7 shows the mutated version of the program.

There are five major steps in the design of a GP system for solving a particular problem. These involve the determination of the following elements:

(1) The function and terminal sets (internal nodes and leaves of the expression tree)
(2) The initial program structures
(3) The fitness function
(4) The genetic operators
(5) The parameters for controlling the process.

As previously shown, GP represents programs as expression trees. This format can be readily used as one of the most common ways of classification: The Decision Tree. In this method each branch node represents a choice between a number of alternatives, and each leaf node represents a classification or decision. A Decision Tree found by the ID3 algorithm proposed in Quinlan, (1986) to solve a classification problem has been shown to be easily evolved using GP (Koza, 1991), showing its learning and classification capabilities.

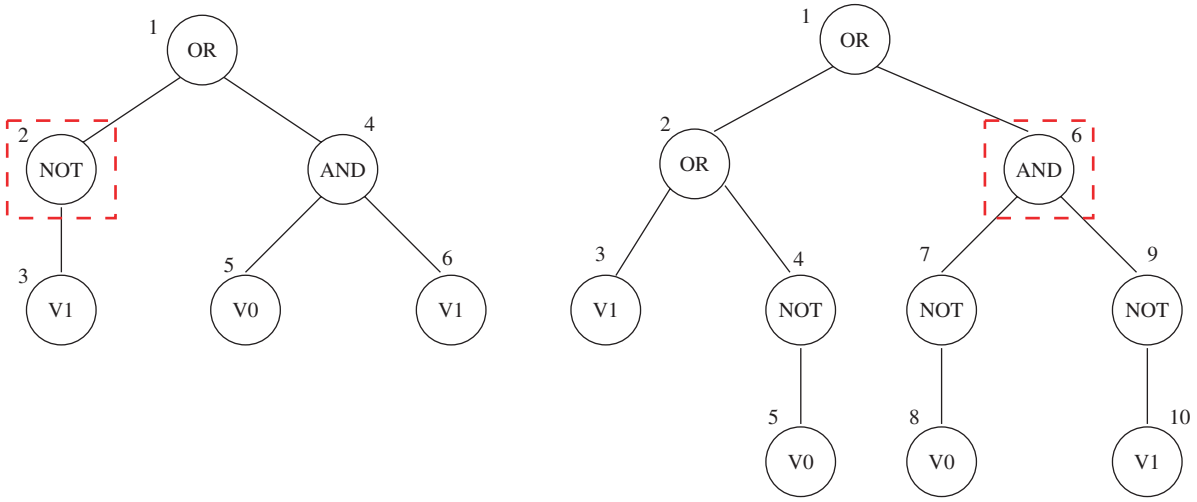


Fig. 2. Parents before crossover. Recombination points are emphasized (dotted-line square).

3 METHODS

3.1 Modeling

3.1.1 Functions Within the context of GP, functions are the structures under adaptation. In the particular approach taken in this work, we modeled functions, for the 1-function per marker coding, as the markers to be evaluated in each test. Take the following configuration set F as an example.

Example : $F = \{BDNF, DRD4, GSTM1, MAOA, HTR1B\}$

In our genetic programming methodology the 1-function per marker coding can take up to $2 + PERMUTATION(N)$ arguments, where N is the number of alleles of the marker. In a biallelic marker $N = 2$ of a diploid species we have three possible genotypes (11,12,22) which we code as 0,1,2. Since we are assuming a discrete binary phenotype for simplicity, our terminals are: $T = (2, 1)$ where 2 is affected and 1 unaffected.

We also tested other coding in which we assign 2 functions per marker (1 function per marker per chromosome), So, instead of having a function named BDNF we now have 2 functions: BDNFA1 and BDNFA2. In this case, each function has exactly 2 arguments one for each possible allele, a reduction of 1 argument compared to the 1-function per marker coding but a 2-fold increase in the number of functions.

3.1.2 Initial structures Depending on problem complexity, usually measured as the number of markers to be included, we assign a fixed number of structures, also called individuals, that will be randomly generated at the beginning of the process. In this work, the size of the initial structures was set to vary from 2 to 6 allowing us to find interactions from the same number of markers.

3.1.3 Fitness function Classification error, defined as the number of incorrectly classified subjects, was set as fitness function. Since the method tends to give large results, also known as bloat, there are several ways to achieve good classification accuracy adjusting by the complexity of the solution (Bleuler, 2001). We chose a simple method called constant parsimony pressure:

$$F_i = E_i + \frac{N_i}{\alpha} \quad (1)$$

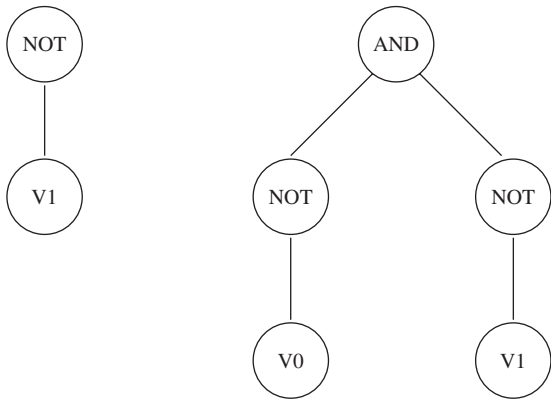


Fig. 3. Crossover segments.

Where E_i is the classification error, N_i is the number of nodes and α is the parsimony constant. Thus a penalty function is applied upon individuals' size. In this sense, the fittest individual will be the one with the best combination of low classification error and small size. Overfitting of the data is possible when using the criteria given by equation (1), nevertheless, a cross-validation procedure showed that in this case, overfitting was not a problem.

3.1.4 Operations We used reproduction, recombination and mutation probabilities of 10, 90 and 0%, respectively as these values are consistently used as an optimal set of parameters. The tournament selection method (Koza, 1992) was used with a tournament size of 7.

3.1.5 Control parameters For this study, we used a population size of 1000 individual programs and the genetic algorithm ran over a 60 generation span. After the last generation is reached, the fittest individual is used as an optimal solution for the prediction step. An optimized calculation trade-off was obtained with this particular combination (1000 initial seeds for 60 generations, in a great number of cases convergence was attained before the 60th generation that indeed was taken as an upper bound).

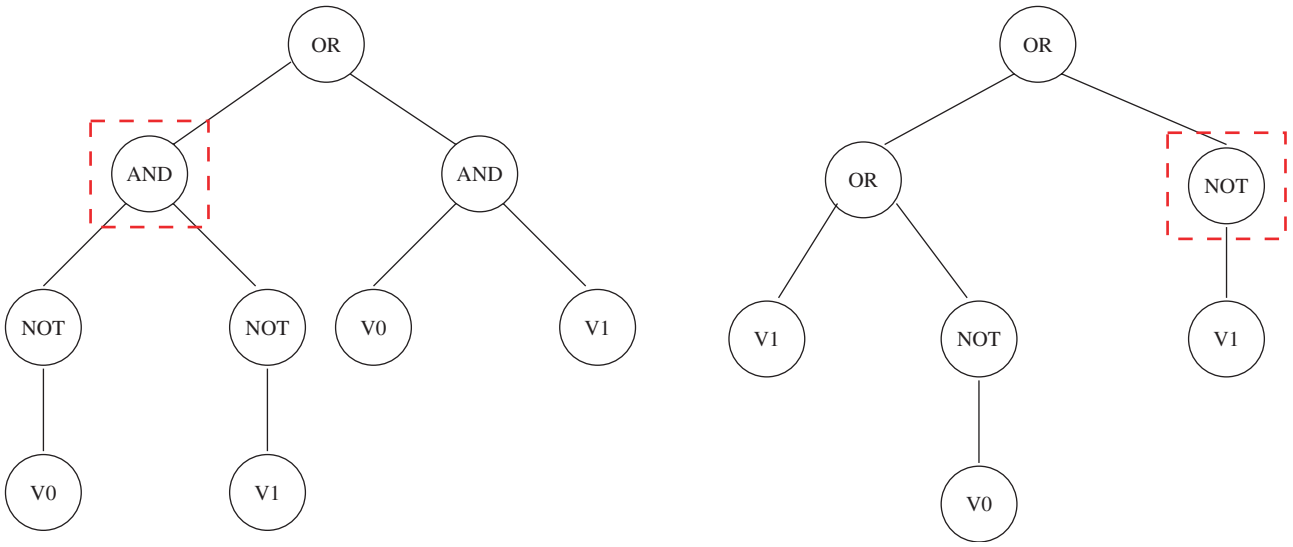


Fig. 4. Offspring after crossover. Recombination points are emphasized (dotted-line square).

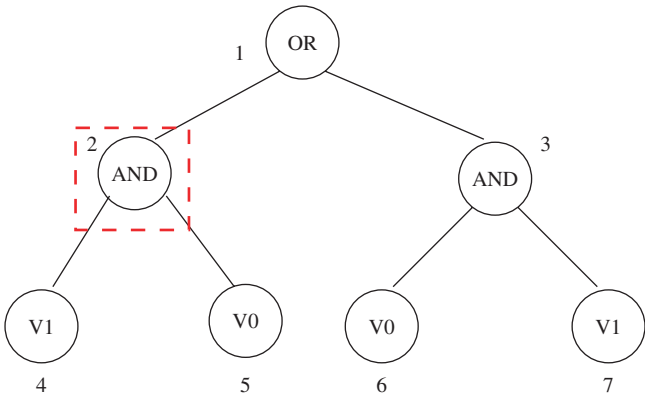


Fig. 5. Offspring before mutation. The mutation point is emphasized (dotted-line square).

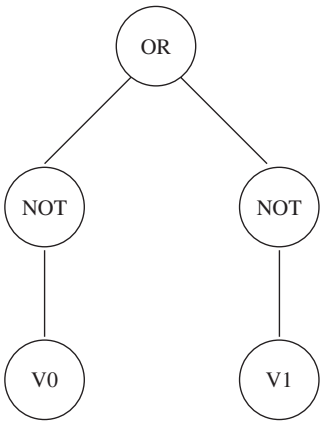


Fig. 6. Mutated subtree.

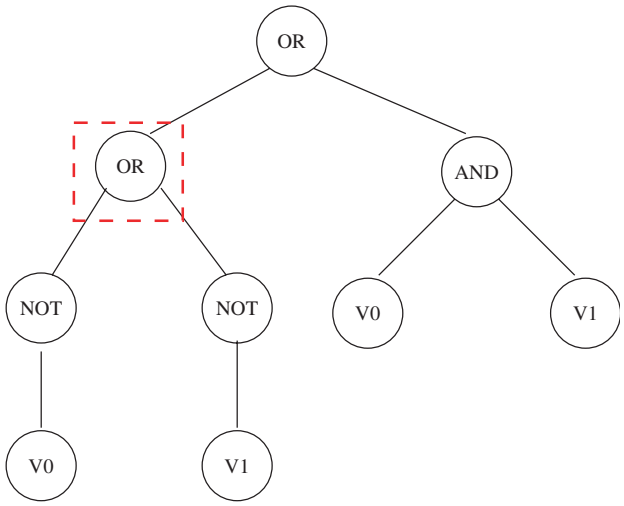


Fig. 7. Offspring after mutation. The mutation point is emphasized (dotted-line square).

3.2 Process

3.2.1 Training After the creation of an initial population, each individual, defined as a particular problem solution represented by a Decision Tree made of functions (markers) and terminals (affection status) (see Table 2), is evaluated by the fitness function. For each subject in the training data set, the individual was used to classify the sample as affected or unaffected by the considered trait. For example, one unaffected sample with genotypes $M2=0$ and $M4=0$ would be correctly classified by the individual in Table 2, however, using the same individual on an affected sample with genotypes $M2=1$ and $M4=0$ would be incorrectly classified. This step is iterated until all training samples have been processed. Following the genetic algorithm, only the fittest individuals will be selected and recombined to create novel solutions into the next generation. The training ends when the specified number of generations is reached. Only the fittest individual at the end of the generations is used in the prediction step.

3.2.2 Prediction The best result obtained from the training step was used to classify the subjects in the prediction data set. Since the best result comes from an optimized procedure it is the natural choice to predict outcomes. The prediction error was measured estimating the proportion of incorrectly classified subjects out of the total. This is a stringent criteria to test the generalizability of the GP algorithm.

3.2.3 Cross-validation To avoid overfitting, prediction error was estimated by a 10-fold cross-validation. The subjects were randomly divided into 10 equal parts. A GPDTI run was done for 9/10 of the subjects and the results were used to make predictions about the 1/10 of excluded subjects. This process was repeated 10 times to have a better estimate of the real prediction error.

Cross-validation was also used to measure how many times the same solution was identified in the 10 runs. A true signal should be present in the data regardless of how they are divided. This hypothesis can be tested by evaluating the consistency of the model across cross-validation data sets (Ritchie *et al*., 2001). However, GP algorithms usually generate results that are not exactly the same, although they have similar power to classify. Take for example the figures in Table 3, those four solutions have converged to solve the same problem but in different ways. If we were to measure the cross-validation consistency (CVC) of this example, it would be zero since all models are different. A new version of the CVC overcomes this limitation, CVC is now defined as the number of times each variable appears in the model across 10 cross-validations (Moore *et al*., 2002; Moore, 2003; Ritchie *et al*., 2003). However, this measure only tell us the consistency with which each single marker was identified by its own, but does not account for the size of the solutions.

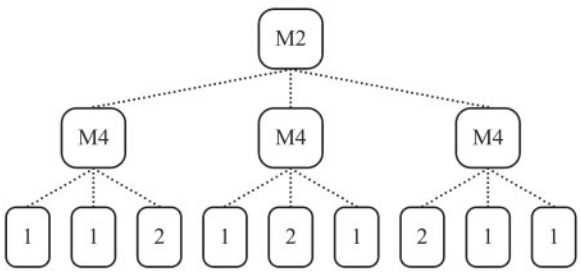
We can classify a GP solution by the set of markers that it is using and the number of nodes that it is made of. We have empirically seen that solutions with the same number of nodes and functions tend to have similar classification power. If we see Table 3 carefully, each solution contains the same function set (M4A2, M2A2, M4A1, M2A1) and has the same number of nodes (17). Thus we can use these two parameters to estimate the consistency of the model through different cross-validations. We have called this measure the extended cross validation consistency (ECVC).

Table 2. Three different representations for the same individual solution

(a) List format
(M2(M4 1 1 2)(M4 1 2 1)(M4 2 1 1))

(b) Rules format
If M2=0 And
If M4=0
then 1
If M4=1
then 1
If M4=2
then 2
If M2=1 And
If M4=0
then 1
If M4=1
then 2
If M4=2
then 1
If M2=2 And
If M4=0
then 2
If M4=1
then 1
If M4=2
then 1

(c) Tree Format



In each case M2 and M4 represents the markers and their 3 possible genotype options (0,1,2). The terminal nodes 1 and 2 represents the classes 'Unaffected' and 'Affected' respectively. (a) The list format is used by the program on its internal processing, (b) the rules format is a more intuitive display set and is best fitted for user reading and (c) the tree is a graphical representation of the list format.

3.3 Data simulation

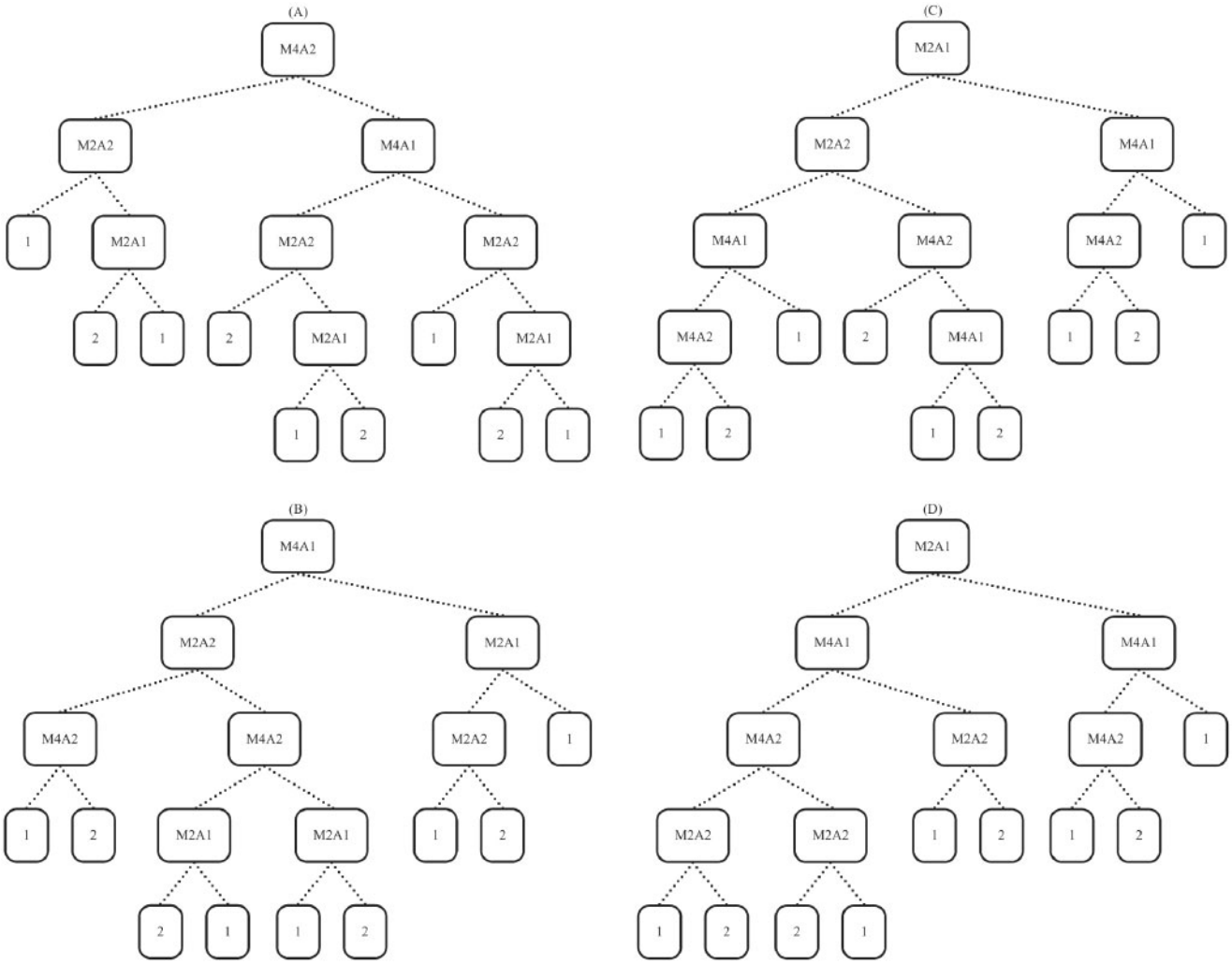
To simulate data using different epistasis models with minimal independent main effects, but with association with disease through interactions, we used two functional biallelic markers in five different previously described models (Culverhouse *et al*., 2002; Li and Reich, 2000; Ritchie *et al*., 2003). The first model is based on the XOR function that generates an interaction effect dependent on having a heterozygous genotype (Aa) from one SNP or from the other (Bb) but not both. Incomplete penetrance is modeled by setting the probability of disease given the combinations of genotypes (such as Aabb, AaBB, AABb, aaBb) to 0.1 and not 1.0 (Table 4). All models in Table 4 were chosen to be able to compare the results with those obtained with the GP neural network (GPNN) method proposed in Ritchie *et al*. (2003). The heritabilities for models 1, 2, 3, 4, 5 are 0.053, 0.051, 0.026, 0.012 and 0.008, respectively, representing the worst-case scenario for a disease-detection method, (Ritchie *et al*., 2003) so it is expected that the method will also work with higher heritabilities. For each described model, 50 different data sets were randomly generated consisting in 200 affected and 200 unaffected subjects each with two functional markers (representing the epistasis model) and eight non-functional markers. Following the cross-validation procedure, training was done using 360 subjects leaving 40 for prediction.

4 RESULTS

An implementation of the GPDTI was achieved using the GP framework *Java-based Evolutionary Computation and Genetic Programming Research System (ECJ)*. Results obtained after comparing GPNN and GPDTI (Table 5) showed that classification error is always smaller when using the latter. For Model 1 both methods have similar prediction errors (0.2370 and 0.2379, respectively). For Models 2 and 3, GPDTI had better prediction errors than GPNN, and in average GPDTI had a similar prediction power than GPNN (0.3464 and 0.35, respectively).

We also compared the consistency of results from GPNN and GPDTI by using CVC of functional markers in the solutions (Table 6). For Models 1, 2 and 3, GPNN and GPDTI had 100% CVC for both functional markers. In Model 4,

Table 3. A, B, C and D are four equivalent Decision Trees that can classify the same problem



GPNN had better CVC values and in Model 5 GPDTI outperformed GPNN. The average generated errors in the 100 data sets reported in Ritchie *et al*. (2003) were 24, 18, 27, 36 and 40% for Models 1, 2, 3, 4 and 5, respectively, which differs from our simulation results that were 23.55, 19.86, 29.36, 47.18 and 47.36%, respectively for the same models. The difference in generated errors for Models 4 and 5 between Ritchie *et al*. (2003) and our simulations (11.18 and 7.36%), could explain why GPNN had less prediction errors than GPDTI. The difference between the generated error and the prediction error was always smaller in GPDTI, suggesting that it has better prediction power than GPNN (Table 5). From these two results, we can see that either GPNN and GPDTI have similar prediction capabilities and consistencies across runs to detect epistatic effects in the proposed models.

In the comparison of the ECVC against the prediction error in our method we found that for models with prediction error <30%, we obtained high ECVC values (86.0 and 90.80%). However, for those models with more than 30% of prediction error, the ECVC values were much smaller (31.4–3.2%). As expected, these results showed that there is an inverse correlation of ECVC and prediction errors and suggests that ECVC is more informative than CVC (Fig. 8).

All previous results were obtained with a 1-function per marker coding. We reran all the analysis for the five models using a 2-functions per marker coding. In average, the 1-function per marker coding had lower prediction errors than the 2-function per marker coding (Table 7). In contrast, the latter had better CVC but not ECVC values (data not shown). This result also suggests that in these situations ECVC is more informative than CVC.

5 DISCUSSION

We have implemented a GP algorithm to induce a Decision Tree. Using simulated data, we demonstrated that GPDTI was able to find interactions even in the presence of incomplete penetrance in models with heritabilities as low as 0.008. Predicted errors in GPDTI were only marginally above the generated error, suggesting that our method is not overfitting to the data.

In this study we tested ECVC, a new measure to evaluate the consistency of the method, and found that it is more informative and had better inverse correlation with prediction error than CVC. We modeled two different function codings, one considered 1-function per marker meanwhile a second one considered 1-function per marker per chromosome (2-function per marker coding). The former had better results in our simulations than the latter, we attribute this result to the reduction on the size of the search space when using 1-function per marker coding.

Results using GPDTI and GPNN were similar as compared with prediction error and CVC. It is well known that a neural network (NN) is usually a good classifier, however, a disadvantage of this method is the interpretation of the GPNN models since its output is a NN in the form of a binary expression tree that can get up to 500 nodes (Ritchie *et al.*, 2003). In contrast, our method produces a Decision Tree that can be transformed to a rules -format that can be easily interpreted.

It has been suggested that 300 000 selected biallelic markers might be enough to capture all common variation in a Caucasian population (Consortium, 2005). By applying GP for attribute selection and expert knowledge it has been shown that it is possible to find a 2-marker interaction with heritabilities as low as 0.3 with 100% in power in a data set of 1000 markers (Moore and White, 2006). Preliminary experiments using GPDTI have been able to find a 3-marker interaction in a data set of 1000 markers and a sample size of 600 subjects (data not shown). Theoretically, there is no limit on the number and size of initial structures in a GP system. However, in practice they are often bounded by the amount of memory available in the underlying computational platform. Efficient implementations such as parallel GP could be used in order to exploit the capabilities of modern computer technologies (Folino, 2001). This is especially important as genetic studies now spawn several orders of magnitude. In effect, whole-genome association studies using over 500 000 markers are increasingly typical and high order interactions are expected to be found in common complex diseases. Further research and experiments are necessary to evaluate the capabilities of GPDTI in this context.

Table 4. Penetrance functions for five different models

	AA(.25)	Aa(.50)	aa(.25)	MP
		Model 1		
BB(.25)	0.0	0.1	0.0	0.05
Bb(.50)	0.1	0.0	0.1	0.05
bb(.25)	0.0	0.1	0.0	0.05
MP	0.05	0.05	0.05	
		Model 2		
BB(.25)	0.0	0.0	0.1	0.025
Bb(.50)	0.0	0.05	0.0	0.025
bb(.25)	0.1	0.0	0.0	0.025
MP	0.025	0.025	0.025	
		Model 3		
BB(.25)	0.0	0.4	0.0	0.02
Bb(.50)	0.04	0.02	0.0	0.02
bb(.25)	0.0	0.0	0.08	0.02
MP	0.02	0.02	0.02	
		Model 4		
BB(.25)	0.0	0.02	0.08	0.03
Bb(.50)	0.05	0.03	0.01	0.03
bb(.25)	0.02	0.04	0.02	0.03
MP	0.03	0.03	0.03	
		Model 5		
BB(.25)	0.0	0.04	0.08	0.04
Bb(.50)	0.06	0.04	0.02	0.04
bb(.25)	0.04	0.04	0.04	0.04
MP	0.04	0.04	0.04	

Each cell represents the probability of disease given the particular combination of genotypes. Model 1 has a heritability of 0.053 and an average generated error of 23.55%. Model 2 has a heritability of 0.051 and an average generated error of 19.86%. Model 3 has a heritability of 0.026 and an average generated error of 29.36%. Model 4 has a heritability of 0.012 and an average generated error of 47.18%. Model 5 has a heritability of 0.008 and an average generated error of 47.36%

Table 5. Classification and prediction errors for GPNN and GPDTI

	GPNN				GPDTI			
Model	Classification error	Prediction error	Generated error	Prediction−generated	Classification error	Prediction error	Generated error	Prediction−generated
1	0.2370	0.2370	0.2400	0.0030	0.2342	0.2379	0.2355	0.0024
2	0.2420	0.2430	0.1800	0.0630	0.2007	0.2034	0.1986	0.0048
3	0.3350	0.3600	0.2700	0.0900	0.2870	0.3192	0.2936	0.0256
4	0.3870	0.4310	0.3600	0.0710	0.3326	0.4825	0.4718	0.0107
5	0.4010	0.4790	0.4000	0.0790	0.3303	0.4892	0.4736	0.0157
Average	0.3204	0.3500	0.2900	0.0612	0.2770	0.3464	0.3346	0.0118

Table 6. Cross-validation consistency for GPNN and GPDTI

	GPNN		GPDTI	
Model	Marker 1	Marker 2	Marker 1	Marker 2
1	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%
4	92.00%	87.00%	83.00%	74.00%
5	44.00%	47.00%	76.00%	77.00%

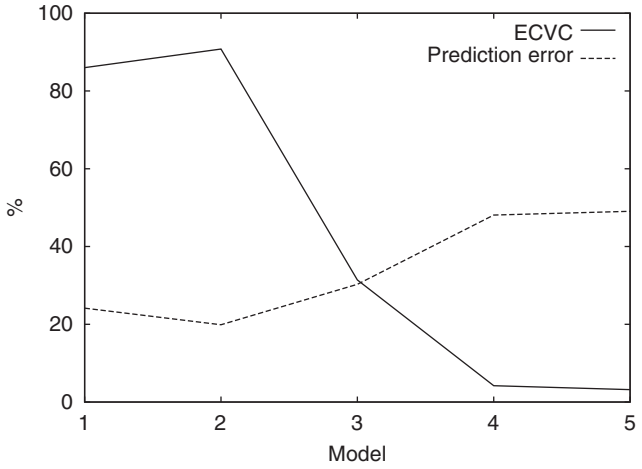


Fig. 8. Extended cross-validation consistency anticorrelates with prediction error.

Table 7. 1-Function versus 2-function per marker coding comparison

	2-Function per marker		1-Function per marker	
Model	Classification error	Prediction error	Classification error	Prediction error
1	0.2467	0.2844	0.2342	0.2379
2	0.2	0.21	0.2007	0.2034
3	0.2918	0.3282	0.2870	0.3192
4	0.3308	0.4943	0.3326	0.4989
5	0.3281	0.4852	0.3303	0.4892
Average	0.2797	0.3600	0.2770	0.3497

ACKNOWLEDGEMENTS

This work was supported by the Consejo Nacional de Ciencia y Tecnología (CONACYT) under SEP-CONACYT award No. SEP-2004-C01-47434. Any opinions, findings and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsoring agency.

Conflicts of Interest: The authors declare that they have no competing financial interests.

REFERENCES

Bleuler,S. *et al*. (2001) Multiobjective genetic programming: reducing bloat using spea2. In: *Proceedings of the Congress on Evolutionary Computation 2001*. Vol. 1, pp. 536–543.

Brassat,D. *et al*. (2006) Multifactor dimensionality reduction reveals gene-gene interactions associated with multiple sclerosis susceptibility in African Americans. *Genes Immun*., **7**, 310–315.

Carlborg,O. *et al*. (2000) The use of a genetic algorithm for simultaneous mapping of multiple interacting quantitative trait loci. *Genetics*, **155**, 2003–2010.

Consortium,I.H. (2005) A haplotype map of the human genome. *Nature*, **437**, 1299–1320.

Culverhouse,R. *et al*. (2002) A perspective on epistasis: limits of models displaying no main effect. *Am. J. Hum. Genet*., **70**, 461–471.

Folino,G. and Pizzuti,C.S.G. (2001) Parallel genetic programming for decision tree induction In *Tools with Artificial Intelligence, Proceedings of the 13th International Conference on*, pp. 129–135.

Frankel,W.N. and Schork,N.J. (1996) Who's afraid of epistasis? *Nat. Genet*., **14**, 371–373 comment.

Hahn,L.W. *et al*. (2003) Multifactor dimensionality reduction software for detecting gene-gene and gene-environment interactions. *Bioinformatics*, **19**, 376–382.

Hoh,J. and Ott,J. (2003) Mathematical multi-locus approaches to localizing complex human trait genes. *Nat. Rev. Genet*., **4**, 701–709.

Hsieh,C.-H. *et al*. (2006) Analysis of epistasis for diabetic nephropathy among type 2 diabetic patients. *Hum. Mol. Genet*., **15**, 2701–2708.

Koza,J.R. and Rice,J.P. (1991) Genetic generation of both the weights and architecture for a neural network. In *International Joint Conference on Neural Networks, IJCNN-91*, IEEE Computer Society Press, Washington State Convention and Trade Center, Seattle, WA, USA. Vol.II, pp. 397–404.

Koza,J.R. (1991) Concept formation and decision tree induction using the genetic programming paradigm. In *PPSN I: Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, Springer-Verlag, London, UK, pp.124–128.

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

Li,W. and Reich,J. (2000) A complete enumeration and classification of two-locus disease models. *Hum. Hered*., **50**, 334–349.

Ljungberg,K., Holmgren,S. and Carlborg,O. (2004) Simultaneous search for multiple QTL using the global optimization algorithm DIRECT. *Bioinformatics*, **20**, 1887–1895. Comparative Study.

Manuguerra,M. *et al*. (2007) Multi-factor dimensionality reduction applied to a large prospective investigation on gene-gene and gene-environment interactions. *Carcinogenesis*, **28**, 414–422.

Moore,J.H. and White,B.C. (2006) Exploiting expert knowledge in genetic programming for genome-wide genetic analysis. In Runarsson T. P., Beyer H.-G., Burke E., Merelo-Guervos J. J., Whitley L. D. and Yao X. (eds) *Parallel Problem Solving from Nature - PPSN IX*, Springer-Verlag, Reykjavik, Iceland. Vol. 4193 of LNCS, pp. 969–977.

Moore,J.H. and Raidl,G. (2003) Cross validation consistency for the assessment of genetic programming results in microarray studies. In Raidl G.R., Cagnoni S., Cardalda J.J.R., Corne D.W., Gottlieb J., Guillot A., Hart E., Johnson C.G., Marchiori E., Meyer J.-A. and Middendorf M. (eds) *Applications of Evolutionary Computing, Evo- Workshops2003: EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, EvoSTIM*, Springer-Verlag, University of Essex, UK. Vol. 2611 of LNCS, pp. 99–106.

Moore,J.H. *et al*. (2002) Symbolic discriminant analysis of microarray data in autoimmune disease. *Genet Epidemiol*, **23**, 57–69.

Motsinger,A.A. *et al*. (2007) Complex gene-gene interactions in multiple sclerosis: a multifactorial approach reveals associations with inflammatory genes. *Neurogenetics*, **8**, 11–20.

Pociot,F. *et al*. (2004) Novel analytical methods applied to type 1 diabetes genome-scan data. *Am. J. Hum. Genet*., **74**, 647–660. Comparative Study.

Quinlan,J. (1986) Induction of decision trees. *Machine Learning*, **1**, 81–106.

Risch,N. and Merikangas,K. (1996) The future of genetic studies of complex human diseases. *Science*, **273**, 1516–1517.

Ritchie,M.D. *et al*. (2001) Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. *Am. J. Hum. Genet*., **69**, 138–147.

Ritchie,M.D. *et al*. (2003) Optimization of neural network architecture using genetic programming improves detection and modeling of gene-gene interactions in studies of human diseases. *BMC Bioinformatics*, **4**, 28.