Protein-Protein Functional Association Prediction Using Genetic Programming

Beatriz Garcia, Ricardo Aler, Agapito Ledezma, and Araceli Sanchis
Universidad Carlos III de Madrid, Computer Science Department
Avda. de la Universidad 30, 28911, Leganes, Madrid, Spain
{beatrizg, aler, ledezma, masm}@inf.uc3m.es

ABSTRACT
Determining if a group of proteins are functionally associated among themselves is an open problem in molecular biology. Within our long term goal of applying Genetic Programming (GP) to this domain, this paper evaluates the feasibility of GP to predict if a given pair of proteins interacts. GP has been chosen because of its potential flexibility in many aspects, such as the definition of operations. In this paper, the *if-unknown* operation is defined, which semantically is the most appropriate in this domain for handling missing values. We have also used the Tarpeian bloat control method to decrease the computational time and the solution size. Our results show that GP is feasible for this domain and that the Tarpeian method can obtain large improvements in search efficiency and interpretability of solutions.

Categories and Subject Descriptors
I.2.2 [**Artificial Intelligence**]: Automatic Programming*;* J.3 [**Life and Medical Sciences**]: Biology and genetics.

General Terms
Algorithms, Measurement, Performance, Experimentation.

1. INTRODUCTION
Understanding the protein interaction networks is essential to identify, explain and regulate the biological process dynamics in living systems. In recent years, many different computational prediction methods have arisen [3] with the aim of reducing costs. Related data exist distributed among multiple databases.

We intend to approach the problem of protein-protein functional association prediction from attributes obtained from different sources and methods, as a binary classification problem. This problem can be tackled by traditional Machine Learning (ML) methods. But our long-term aim is to apply GP [1] to this biological domain because of GP's potential flexibility.

One of the reasons for choosing GP is that this technique allows us to define the primitives most appropriate for the problem (for example, the *if_unknown (if_?)* operator, which manages the missing values). Also, we try to improve the accuracy and readability of equations evolved by GP by using the well founded Tarpeian bloat control mechanism [2], which biases evolution towards simple solutions, avoiding GP individuals growing in size without apparent gain in fitness. We also expected that the Tarpeian method will speed-up the evolution of solutions.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.
ACM 978-1-60558-130-9/08/07.

2. PROTEIN-PROTEIN FUNCTIONAL ASSOCIATION PREDICTION PROBLEM
In this research, we do not predict physical but functional interaction between pairs of proteins. The application domain is the proteome of *E.coli* which 4,339 known proteins.

The database sources where the 89,401 positive instances are retrieved from are BIND, DIP, IntAct, EcoCyc, KEGG, iHoP and Butland's set. Each one contains information about evidence (such as physical interactions, complexes, regulation processes, metabolic pathways and text mining) which indicates the possibility of an interaction between pairs of proteins.

3. EXPERIMENTAL SET UP

3.1 Data Representation and Solution Coding
The data are represented in attribute-value pairs. We define 9 features: 5 scores from 5 prediction computational methods [3] based on different evidence, and 4 biological characteristics.

The instances are divided in two classes. The positive class includes pairs of proteins which appear in some of the databases previously mentioned. The negative class includes the rest of pairs composed by *E.coli* proteins, but not included in the positive set. After applying different filters, train and test sets have 10,000 instances each one (half positive class half negative class).

We define here the elements which are part of the GP trees. There are 10 terminals: the 9 attributes explained above and 1 ERC [1]. The operators used are the arithmetical ones, the conditional one *[if (a>=b) then x else y]*, and finally one new specific operator, tailored for this domain: if_? *[if (k is unknown) then x else y]*.

3.2 Evolutionary Process
The evolved individual *f*, is applied to two proteins *(p1, p2)*, and a threshold is used to give a positive or a negative class. Hence, *if (f >= threshold) then (p1, p2) functionally interact; else (p1, p2) do not interact*. In this work, the threshold is 0.5.

The fitness function is *(TP+TN)/(TP+TN+FP+FN)*, according to T, True; F, False; P, Positives; and N, Negatives.

An appropriate base configuration for the evolutionary process parameters (a detailed description in the *lil-gp 1.1* GP tool manual [5]) has been found from a few preliminary experiments: 1,000 for population size, 50 for no. generations, 17 for maximum depth, 200 for maximum no. nodes, crossover, reproduction and mutation like genetic operators (with 0.5, 0.1 and 0.4 probability, respectively), tournament with size=7 for individuals selection method, and arithmetical and conditional operators.

4. RESULTS

4.1 Comparison with other ML Techniques

Table 1 summarizes results from other ML techniques (using *Weka* [4]). All the parameters follow default *Weka* options.

Table 1. GP and Machine Learning: accuracy comparison.

Algorithm	% Train	% Test	% Test with unknown values	Test Sensitivity TP/TP+FN	Test Specificity TN/TN+FP
GP	62.34 / 62.92	60.83 / 61.44	60.67 / 61.22	58.87 / 63.54	62.62 / 59.34
ADTree	61.28	60.02	60.35	64.56	55.48
AODE	62.48	61.32	58.99	48.60	74.04
KStar	98.86	61.60	58.92	60.24	62.96
MLP	58.85	58.22	60.00	20.40	96.06
PART	64.06	61.96	58.33	60.84	63.08
Simple Logistic	60.29	60.70	57.61	56.34	65.06
SMO	59.17	59.96	57.62	56.98	62.94

Table 1 shows that the accuracy in both train and test results (first and second columns) is nearly the same in all classifiers (except Kstar), with values around 60-61% in test (in GP, 60.83% on average and 61.44 for the best run). Besides, almost all algorithms get similar correct predictions in both classes (see the two last columns). In conclusion, GP gets accuracy about as high as most of traditional ML algorithms that we have tested.

4.2 Changing Significant Parameters: if_? Operator and the Tarpeian Method

4.2.1 Missing Values Handling Comparison

A missing value is a feature without a known value in some of the instances. Missing values are a relevant problem in this domain, because of the huge number of them. The *if_unknown* operator is designed in this work specifically to try to solve this problem.

Two different approaches for missing values handling are validated in this section. The former fills them in with a specific numerical flag (base configuration). The latter one preserves the missing values in the data, where GP adds the new operator (*if_?*).

The second and third columns in Table 1 show the test accuracy corresponding to the first and the second approach, respectively. In the first approach, PART is slightly better than GP. However, if unknown values are preserved in the data set, GP outperforms the other ML algorithms (see the highest value in the third column).

4.2.2 Different Configurations Comparison

Figure 1 shows how tree size and execution time change for six different experiment configurations. All the configurations displayed come from averaging 30 GP runs. (a)*Base* is the initial configuration, whose parameters were mentioned previously. (b)*Base without limit* means the *base* configuration but without restricting the maximum tree size. (c)*If_?* refers to *base* configuration including this new operator. Finally, (e)*Tarpeian* configuration comprises this control bloat method and the *without limit* characteristic. (d)*if_? & without limit* and (f)*if_? & Tarpeian* are configurations which includes the elements of both of them.

In Figure 1, the Y-axis quantifies size (in no. nodes) and time (in seconds) on average. The scale is the same for both measures.

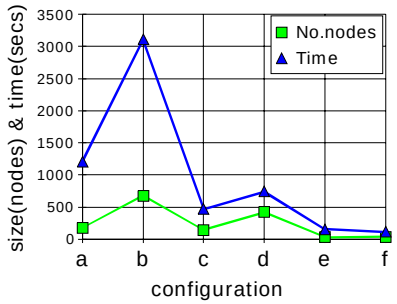

Figure 1. Influence of if_? and Tarpeian: tree size and time.

With reference to tree size and time (see Figure 1), the values for 'c' to 'f' configurations, which include the *if_?* operator or/and the Tarpeian method, are considerably lower than others. In addition, the solution size is quite shorter than in PART.

To sum up, the *if_?* operator and the Tarpeian method reduce tree size and time, dropping scarcely test accuracy. Nevertheless, the obtained trees have an easier interpretation and a very much faster evolution process. Therefore, it seems convenient to include in the solution both the *if_?* operator and the Tarpeian method.

5. CONCLUSIONS

In this paper, we have applied Genetic Programming (GP) to the protein-protein functional association prediction problem. Our initial work shows that GP manages to obtain accuracy results similar to other ML methods (around 61%). Besides, the predictor integrates information from different sources.

We have taken advantage of the flexibility offered by GP to define primitives. For example, *if_?* operator, that takes into account the large number of unknown values in our data. GP handles missing values slightly better than the rest of ML algorithms tested.

We have tried to reduce bloat by means of the *if_?* operator and the Tarpeian method. Two negative effects of bloat are controlled in this domain. First, the tree size has been reduced. Second, the execution time goes down, due to do not wasting evaluating excessive big trees, improving the efficiency of the GP system. Both effects are achieved with almost no decrease in accuracy.

6. ACKNOWLEDGMENTS

Data used in these experiments has been obtained in support of the Structural Computational Biology Group in Spanish National Cancer Research Centre (CNIO). This work has been supported by CICYT (2004-07) TRA2004-07441-C03-02/IA project.

7. REFERENCES

[1] Koza J. *Genetic Programming II.* MIT Press, 1994.

[2] Poli R. A Simple but Theoretically-Motivated Method to Control Bloat in Genetic Programming. In *Proceedings of EuroGP'03.* Springer Berlin, (Apr 2003), 43-76.

[3] Valencia A. and Pazos F. Computational methods for the prediction of protein interactions. *Curr. Opin. Struct. Biol., 12,* 3 (Jun 2002), 368-373.

[4] Witten I. H. and Frank E. *Data Mining: Practical machine learning tools and techniques*. San Francisco, 2005.

[5] Zongker D. and Punch B. *lil-gp.* Michigan State University, Michigan, 1998. http://garage.cse.msu.edu/software/lil-gp/.