Binary Representation in Gene Expression Programming: Towards a Better Scalability

Jose G. Moreno-Torres
Illinois Genetic Algorithms Laboratory (IlliGAL)
University of Illinois at Urbana-Champaign
104 S. Mathews Ave, Urbana, IL 61801
josegmt@illinois.edu

Xavier Llorà
National Center of Supercomputing Applications
University of Illinois at Urbana-Champaign
1205 W. Clark St, Urbana, IL 61801
xllora@illinois.edu

David E. Goldberg
Illinois Genetic Algorithms Laboratory (IlliGAL)
University of Illinois at Urbana-Champaign
104 S. Mathews Ave, Urbana, IL 61801
deg@illinois.edu

ABSTRACT

One of the main problems that arises when using gene expression programming (GEP) conditions in learning classifier systems is the increasing number of symbols present as the problem size grows. When doing model-building LCS, this issue limits the scalability of such a technique, due to the cost required. This paper proposes a binary representation of GEP chromosomes to paliate the computation requirements needed. A theoretical reasoning behind the proposed representation is provided, along with empirical validation.

Categories and Subject Descriptors:
I.5.2 [**Pattern Recognition**]: Design Methodology - *classifier design and evaluation*

General Terms: Algorithms

Keywords: machine learning, genetic algorithms, gene expression programming, classifier systems

1. INTRODUCTION

There have been interesting results when attempting to use GEP-based[1] conditions in learning classifier systems [5]. Despite its flexibility, when used in model-building LCS, there is a scalability limitation [3]. This limitation comes from the arity of the individuals growing linearly with the problem size. Such a growth usually leads to the population sizes requiring an exponential growth. This paper presents a possible solution to this problem by using a different representation of the GEP chromosome. We begin presenting an alternative representation, followed by a theoretical analysis of the advantages. Then, we construct a set of experiments to test the proposed representation and to confirm its merits. Finally, we analyze the results from our experiments, and discuss their meaning and relevance.

2. BINARY REPRESENTATION AND GEP

The proposed representation provides:

1. An arity $\chi = 2$.
2. A logarithmic increase of the population size encoding requirements respect the number of symbols.
3. A Hamming distance bound between any two functions' representation: $d(f_1, f_2) <= log_2(nFunctions)$

This representation uses its first bit to distinguish between functions and variables/constants. If the bit string starts with a '1', this indicates a variable/constant, and the following bits encode what variable or constant it is (constants follow inmediately after the last variable in the problem, as is shown in the example). Otherwise, it is a function, and again the following bits determine which one it is. Non-coding bits are ignored. The number of bits to encode a symbol can be calculated as

$$bPS = 1 + \max(\lceil \log_2(nFunctions) \rceil, \lceil \log_2(nConstants + nVariables) \rceil). \quad (1)$$

For instance, let us assume a problem with $nVariables = 30$ variables, and let the maximum number of constants $nConstants = 10$, and to have $nFunctions = 6$ different possible functions. Substituting in (1),

$$bPS = 1 + \max(3, 6) = 7. \quad (2)$$

The representations of some different possible symbols are:

- 3rd function $= 0011 * **$ ($*$ represents a non-coding bit).
- 23rd variable $= 1010111$.
- 4th constant $= 1100010$. (The 4th constant is the equivalent to the $nVariables + 4$ variable, so it is the 34th variable in our example).

3. THEORETICAL ANALYSIS

Goldberg[2] gives a theoretical bound for the population size needed for BB signal:

$$s = O(m \log(m) \cdot \chi^k), \quad (3)$$

where s is the population size, m is the number of building blocks, χ is the arity of the alphabet in our representation and k is the size of the building block. Given a χ-ary representation scheme, a building block of size two gives us $m = 1$, $\chi = N$ (where N = dimensionality of the problem, in the worst case $nVariables + nConstants + nFunctions$), $k = 2$. From these values, we can substitute in (3) and we obtain

$$s = O(N^2), \quad (4)$$

so the population size grows quadratically with the problem size.

With a binary representation, the building block size depends on the distance between the symbols we are learning: $k = 2(d+1)$, where d is the Hamming distance. Substituting in our population size formula (3), we get

$$s = O(\log(N)\log(\log(N)) * 2^{2(d+1)}). \quad (5)$$

Since we have an upper bound for the Hamming distance ($d = \log_2(N)$), we can solve (5) to get

$$\begin{aligned} s &= O(\log(N)\log(\log(N)) * 2^{2\log(N)}) \\ &= O(\log(N)\log(\log(N)) * N^2). \quad (6) \end{aligned}$$

Since we chose to separate functions and terminals, we can bound the maximum Hamming distance when working with functions to $d <= \log_2(nFunctions)$; and since $nFunctions$ is constant, we can solve (5) to get a new theoretical prediction for the binary representation:

$$s = O(\log(N)\log(\log(N))). \quad (7)$$

4. EXPERIMENTS AND RESULTS

The aim of the paper was to study how the population size required for model building grows as a function of the problem size. The tests were run using ECGA's model building algorithm, probing the population size needed to extract the correct building blocks as the problem size increases. In order to consider a population size to be successful, it has to extract either all the correct building blocks or all of them but one in 19 out of 20 runs. To perform the testing, an artificial population of the desired size was built to apply the model building to.

We performed two independent experiments. In the first one, we seek confirmation for the hypothesis stating that the Hamming distance is the real measure of difficulty when trying to model a building block composed by two symbols, doing it with variables: $x \quad or \quad y$. For this experiment, in the binary case, we did a worst-case study, by fixing x and y to be as far away from each other as possible, in terms of Hamming distance (ie, a population size of 128 actually means $hDistance = \log_2(128) = 7$).

The second experiment was designed to probe the effectiveness of the designed representation when dealing with functions only. In this experiment, we ignored the terminals that would need to be present in order to have a legal expression, and focused only on the BB formed by the functions. What this means is that, for an expression like $(x_i \quad or \quad x_j) \quad and \quad not \quad x_k$, this experiment would only focus on the functions $(or)and(not)$, ignoring the variables.

According to the theoretical analysis presented previously, equivalent results for both tests when using a χ-ary representation are to be expected, following equation (4); while the binary representation should follow (6) for the first experiment, and (7) for the second one, showing a significant improvement over the χ-ary representation in the second experiment.

In the first experiment, for the binary case, the growth follows (6), and in the χ-ary one it follows (4), so there is a slight advantage for the χ-ary representation. Again, it should be noted that in the binary case we conducted a worst-case analysis, while the χ-ary one has the same results independently of the symbols involved.

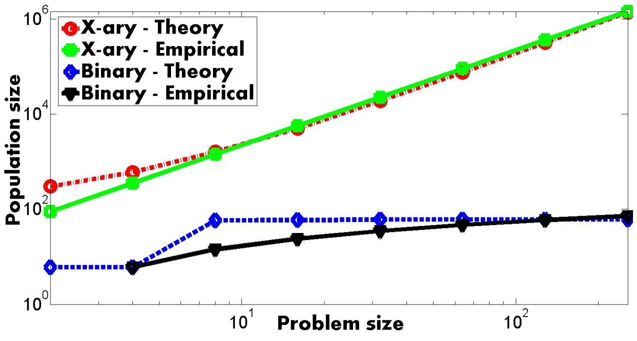

Figure 1: Population size as a function of problem size for binary vs χ-ary representations when learning $(or)and(not)$

Figure 1 shows the results obtained in the second experiment. Results accurately match the theoretical model proposed. This a key result of this study, since it shows a real improvement in the population size needed for model building when dealing with relationships between functions.

5. DISCUSSION AND CONCLUSIONS

We have designed a new representation for LCS using GEP that, in theory, can outperform the commonly used χ-ary representation when working with functions. However, the binary representation has a slightly worse performance than the χ-ary one when dealing with relationships between variables/constants.

The performance improvement is the first step towards solving the scalability limitations to effectively using GEP conditions model-building LCS. The empirical data matches remarkably closely the one predicted by the equations derived from theory, which gives us confidence that we can develop solutions based on this theory.

Acknowledgments

The authors would like to acknowledge the technical support provided by Thyago Duque. They would also like to thank Obra Social “la Caixa” for its financial support.

6. REFERENCES

[1] C. Ferreira. *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence.* Springer-Verlag, Germany, 2006.

[2] D. E. Goldberg. *The Design of Innovation: Lessons from and for competent genetic algorithms.* Kluwer Academic publishers, Boston, MA, 2002.

[3] X. Llorà, K. Sastry, C. F. Lima, F. G. Lobo and D. E. Goldberg. Linkage Learning, Rule Representation, and the X-Ary Extended Compact Classifier System. In *IWLCS*, pages 189–205, 2007.

[4] J. G. Moreno-Torres, X. Llorà, and D. E. Goldberg. Binary representation in gene expression programming: Towards a better scalability. Technical report, Illinois Genetic Algorithms Lab (IlliGAL) at UIUC, 2009.

[5] S. W. Wilson. Classifier conditions using gene expression programming. In *IWLCS*, pages 206–217, 2007.