Evolving mathematical expressions using Genetic Algorithms

Crina Groşan

Department of Computer Science
Babeş-Bolyai University, Kogălniceanu 1
Cluj-Napoca, 3400, Romania.
cgrosan@cs.ubbcluj.ro

Abstract. A new evolutionary technique for evolving mathematical expressions called Mathematical Expressions Evolver (MEXE) is proposed in this paper. Each MEXE chromosome is a string of operands and operators. Each subexpressions encoded in a chromosome is considered a potential solution of the problem. The best of these expressions is chosen to represent the chromosome. In this way variable length expressions can be stored in a fixed length chromosome. Several numerical experiments for solving symbolic regression problems are performed. Results shows that MEXE performs very well on the considered examples.

1 Introduction

Evolving mathematical expressions is a well-established field in Genetic Programming. Many evolutionary techniques which can be applied for evolving mathematical expressions (formulas) have been proposed in the recent past. Genetic Programming (GP) [5, 6], Multi Expressions Programming (MEP) [9], Linear Genetic Programming (LGP) [3, 8], Gramatical Evolution (GE) [10], Gene Expression Programming [4], Formula Prediction using Genetic Algorithms (FPEG) [1] are some of them.

A new evolutionary algorithm for detecting mathematical expressions is proposed in this paper. This algorithm is called Mathematical Expressions Evolver (MEXE). A MEXE solution is a string of operators and operands. Each subexpressions encoded in a chromosome is considered a potential solution of the problem. The best of these expressions is chosen to represent the chromosome. In this way more expressions are encoded in the same chromosome.

The paper is organized as follows: In Section 2 MEXE algorithm is described. Some numerical experiments for detecting quartic and sextic functions are performed in Section 3. A set of conclusions are presented in Section 4 of the paper.

2 MEXE algorithm

The proposed algorithm that can be applied for detecting mathematical expressions is presented in this section. Used genetic operators are crossover and mutation. Solution representation and genetic operators are described in what follows.

2.1 Solution representation

A set of operands and a set of operators are considered. Each individual (chromosome) is represented as a string of operands and operators. The odd positions of the chromosome consist of operands and the even positions consist of operators. The chromosome length is constant during the search process.

For instance, if the operands set is {x, y} and the operators set is {+, -, *, /} then and example of chromosome with length 9 is:

$$C = x + y * y - x/y$$

Remark

In the case that we use only binary operators the chromosome length has to be an odd number because the last position of the chromosome cannot be an operator.

2.2 Genetic operators

The genetic operators used by this algorithm are mutation and crossover. Both of them are illustrated bellow.

The mutation operator can affect each gene of the chromosome with a given mutation probability. For each gene a real number between 0 and 1 is generated. If this value is lower than mutation probability than the value of this gene is changed with another random generated value (if the value of the gene is an operator than another operator from the given set of operators is generated; if the value of a gene is an operand then another operand from the given set of operands is generated).

Let us suppose we have the chromosome:

$$C = x + y * y - x \ / \ y$$

and the use of mutation operator affect the genes 3 and 6. Then a new obtained chromosome can be:

$$C = x + \mathrm{x} \ * \ y + x \ / \ y$$

The value of the gene corresponding to 3^{rd} position has been changed from y to x and the value of the gene corresponding to 6^{th} position ("-") has been changed to the value "+".

The crossover operators specific to binary encoding (one cut point, two cut points, uniform, etc.) may be applied.

2.3 Fitness assignment

Each of the subexpressions encoded in a MEXE chromosome is considered as a potential problem solution. For instance the chromosome

$$C = x + y \ * \ y - x \ / \ y$$

encodes the expressions:

$E_1 = x$;
$E_2 = x + y$;
$E_3 = x + y * y$;
$E_4 = x + y * y - x$;
$E_5 = x + y * y - x \ / \ y$;

The value of these expressions may be computed by reading the chromosome from left to right. Partial results are by the means of dynamic programming [2].

The chromosome fitness is usually defined as the fitness of the best expression detected in that chromosome.

For instance, if we want to solve symbolic regression problems, the fitness of each sub-expression E_i may be computed using the formula:

$$f(E_i) = \sum_{k=1}^{n} |o_{k,i} - w_k|,$$

where $o_{k,i}$ is the result obtained by the expression E_i for the fitness case k and w_k is the targeted result for the fitness case k. In this case the fitness needs to be minimized.

The fitness of an individual is equal to the lowest fitness of the expressions encoded in the chromosome:

$$f(C) = \min_i f(E_i).$$

There is neither practical nor theoretical evidence that one of these expressions is better than the others. Moreover, Wolpert and McReady [12] proved that we cannot use the search algorithm's behavior so far for a particular test function to predict its future behavior on that function.

2.4 Algorithm description

Standard MEXE algorithm uses steady-state evolutionary model [11] as its underlying mechanism.

The MEXE algorithm starts by creating a random population of individuals. The following steps are repeated until a given number of generations is reached: Two parents are selected using a standard selection procedure. The parents are recombined in order to obtain two offspring. The offspring are considered for mutation. The best offspring O replaces the worst individual W in the current population if O is better than W.

The variation operators ensure that the chromosome length is a constant of the search process. The algorithm returns as its answer the best expression evolved along a fixed number of generations.

3 Numerical experiments

Test problems used in our experiment are presented bellow. The relationship between the success rate and the chromosomes length, population size and the number of iterations is also performed in the next sections. For these experiments the operators set is {+, -, *, /}.

In these experiments the problems for detection of quartic function $f(x) = x^4 + x^3 + x^2 + x$ and the sextic polynomial functions $f(x) = x^6 - 2x^4 + x^2$ are considered.

The training set for quartic and sextic functions contains 20 instances randomly generated between zero and one.

3.1 Experiment

In this experiment the relationship between the chromosomes length and the success rate is analyzed. The results obtained in 100 runs are considered.

For the quartic function the chromosome length varies between 20 and 100.

The parameters used by MEXE algorithm for the quartic function are: mutation probability: 0.3; crossover probability:0.9; population size: 50; number of iterations: 50.

The parameters used by MEXE algorithm for the sextic polynomial function are: Mutation probability: 0.3; crossover probability: 0.9; population size: 100; number of iterations: 50.

Success rate for quartic function is depicted in Figure 1.

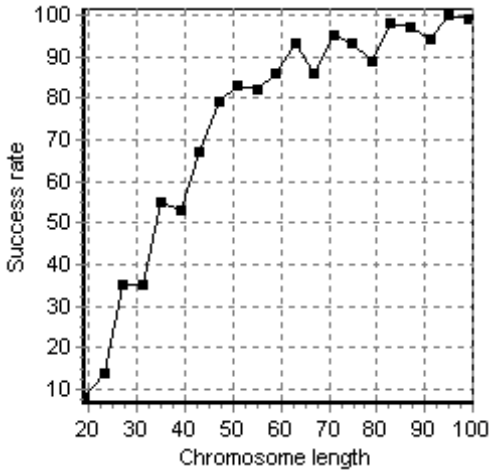

Fig. 1. The relationship between the chromosome length and the success rate for the quartic function. Results are averaged over 100 runs.

Figure 1 shows that MEXE success rate is 100% when the chromosome length is 95. We performed several experiments with Genetic Programming [5]. Using similar parameter setting we obtained a success rate of 90% for GP.

For the sextic polynomial function the chromosome length varies between 33 and 150. Success rate for sextic polynomial function is depicted in Figure 2.

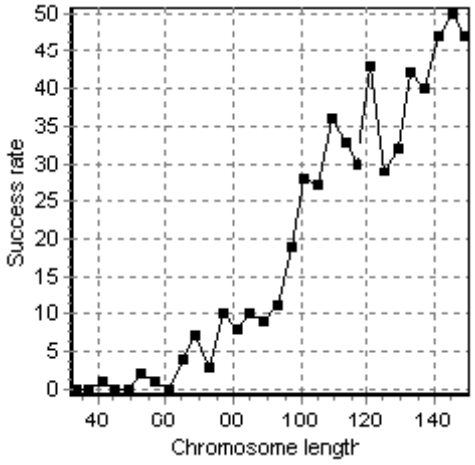

Fig. 2. The relationship between chromosome length and success rate for the sextic polynomial function. Results are averaged over 100 runs.

We performed several experiments with Genetic Programming for solving the sextic polynomial function. Using similar parameter setting we obtained a success rate of 34 % for GP. By contrast, the success rate of MEXE algorithm 50 %.

4 Conclusions and further work

A new evolutionary technique for detecting mathematical expressions has been proposed in this paper. The proposed algorithm encodes multiple expressions in the same chromosome. In this way fixed-length chromosome are used to encode variable-length expressions. Several numerical experiments for detecting quartic and sextic polynomial functions have been performed in this paper. Results show that MEXE performs better than Genetic Programming. Other numerical experiment can be performed in order to analyze the relationship between success rate and population size, number of iterations, etc.

MEXE algorithm will be extended to deal with a larger set of operators (including unary operators). In this way more complicated data set can be analyzed.

References

1. Aldawoodi N., Perez, R.: Formula prediction using Genetic Algorithms. In Proceedings of Genetic and Evolutionary Computation Conference (GECCO), Late breaking papers, 1-6, Chicago, IL, 2003.
2. Bellman, R.: Dynamic Programming, Princeton, Princeton University Press, New Jersey, 1957.
3. Brameier, M. and Banzhaf, W.: Evolving Teams of Predictors with Linear Genetic Programming, Genetic Programming and Evolvable Machines, 2:381-407, 2001.
4. Ferreira, C.: Gene Expression Programming: a New Adaptive Algorithm for Solving Problems, Complex Systems, Vol. 13, 87-129, 2001.
5. Koza, J. R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, 1992.
6. Koza, J. R.: Genetic Programming II: Automatic Discovery of Reusable Subprograms, Cambridge, MA: MIT Press, 1994.
7. Miller, J., Thomson, P.: Cartesian Genetic Programming, In R. Poli et al (editor) Third European Conference on Genetic Programming, Vol. 1802, Lecture notes in Computer Science, Springer, 2002.
8. Nordin, P.: A Compiling Genetic Programming System that Directly Manipulates the Machine Code, K. E. Kinnear, Jr. (editor), Advances in Genetic Programming, pp. 311-331, MIT Press, 1994.
9. Oltean, M., Grosan, C.: Evolving Evolutionary Algorithms using Multi Expression Programming, The 7th European Conference on Artificial Life, September 14-17, 2003, Dortmund, W. Banzhaf et al. (editor), LNAI 2801, pp. 651-658, Springer Berlin, 2003.
10. O'Neill, M., Ryan, C.: Grammatical Evolution, IEEE Transaction on Evolutionary Computation, Vol 5, No 4, 2001.
11. Syswerda, G.: Uniform Crossover in Genetic Algorithms, Schaffer, J.D., (editor), Proceedings of the 3^{rd} International Conference on Genetic Algorithms, pp. 2-9, Morgan Kaufmann Publishers, San Mateo, CA, 1989.
12. Wolpert D.H., McReady W.G.: No Free Lunch Theorems for Optimisation, IEEE Transaction on Evolutionary Computation, Vol. 1, 67-82, 1997.