Automated Discovery of Symbolic Approximation Formulae using Genetic Programming

Mohamed M. Khatib
Faculty of Informatics Engineering
Aleppo University, Syria

ABSTRACT
This paper describes the use of genetic programming to automate the discovery of symbolic approximation formulae. Results are presented involving discovery of numeric approximation formulae to common functions, which are compared to Padé approximations obtained through a symbolic mathematics package. Based on these results, we consider genetic programming to be a powerful and effective technique for the automated discovery of symbolic approximation formulae.

Keywords
Genetic Programming, Padé approximations, Symbolic Regression

1. INTRODUCTION
Genetic Algorithms (GA) are probabilistic search algorithms characterized by the fact that a number N of potential solutions (called individuals) of the optimization problem simultaneously samples the search space (Holland, 1975). This population is modified according to the natural evolutionary process: after initialization, selection and recombination are executed in a loop until some termination criterion is reached. Genetic Programming (GP) uses the same principle as GA and the representation of the individuals is done by trees. Especially in control systems the ability of GP to generate symbolic solutions makes it an interesting optimization tool for many problems, e.g., symbolic regression is such a problem where functional dependencies are searched.

This task is similar to the system identification problem, where the behavior of an unknown system is to be learned and predicted based on examples given as input-output data. Time-series prediction, pattern recognition and the classification problem also belong to this problem class (Iba, 2019). The approximation function g(x) may be of predefined shape, for example a polynomial of a certain degree. In this case the regression simply consists of adjusting coefficients. Hence one is faced with a numerical optimization problem and all methods for numerical optimization may be applied, e.g., gradient methods, hill climbing methods, tabu search, genetic algorithms, evolution strategies. If the appearance of the function g(x) is unknown in advance these methods are not applicable. In this case GP is a perfect candidate for solving the problem. One specifies basic functions and terminals from which the approximation formula g can be composed and GP automatically evolves the shape and size of the formula.

This paper presents a GP-based symbolic regression system applied to automate the discovery of approximation formulae to common functions. Each approximation is an expression that can be evaluated numerically at a number of points in a given range so that the relative accuracy of each approximation can be compared. The next section gives an overview to related work; Sec. 3 explains what a symbolic regression is, Sec. 4 describes the regression model, Sec 5 presents the symbolic approximation process and Sec 6 gives the results. Lastly, conclusions are given in Sec 7.

2. RELATED WORK
Approximation problem has been investigated by (Koza, 1992; Andre & Koza, 1996; Chellapilla, 1997; Luke and Spector, 1997; Nordin, 1997; Ryan, Collins, & O'Neill, 2018). Notably, the economics exchange equation (M=PQ/V) (Koza, 1990b) and Kepler's third law (Koza, 1990a) have been rediscovered from empirical data through GP symbolic regression. Approximation of specific functions has been performed by (Keane, Koza, & Rice 1993), who use genetic programming to find an approximation to the impulse response function for a linear time invariant system, and by (Blickle & Thiele, 1995), who derives three analytic approximation formulae for functions concerning performance of various selection schemes in genetic programming.

3. SYMBOLIC REGRESSION WITH GENETIC PROGRAMMING
The basic approach of using GP for symbolic regression has already been outlined by John Koza (Koza, 1999). The objective of solving a symbolic regression problem is finding a function that closely matches some unknown function on a certain interval. More formally, given an unknown function f(x) we want to find a function g(x) such that, where X is a set of values drawn from the interval, we are interested in. Note that we normally do not know f(x) precisely. We only know the set of sample points. The advantages of using GP for symbolic regression can be summarized as follows:

- Size and shape of the approximation function need not be known in advance.
- A small risk of over-fitting the data.
- Arbitrary complex functions can be supported in the function set. Thereby allowing application specific knowledge to be included in the search process.

4. THE REGRESSION MODEL
In the regression model the individuals in the genetic population are compositions of primitive functions and terminals. The set of primitive functions used is. The set of terminals used is, where RPC symbolizes a random constant. Note that we chose the terminal and function sets so that any function can operate upon the result of any function or terminal. In this case they are all float, additionally we define divide by zero to be unity. Also note, all the operators have two operands, so the programs evolved have the form of a binary tree.

By limiting the set of functions to the arithmetic function set, it is possible to evolve rational polynomial approximations to functions, where a rational polynomial is defined as the ratio of two polynomial expressions. Since approximations evolved with the specified function set use only arithmetic operators, they can easily be converted to rational polynomial form by hand, or by using a symbolic mathematics package such as Maple. Approximations evolved in this manner can be compared to approximations obtained through other techniques such as padé approximations (see section 6).

4.1 Initial Programs (Populations)

The first step in actually performing a GP run is to initialize the population. There are two different methods for initializing tree structures in GP, namely full and grow (Wolfgang, 1998). The full method generates full trees of the given depth. The grow method generates trees of random structure of depth at most equal to the maximum depth. Diversity is valuable in GP populations. The above methods could result in a uniform set of structures in initial population because the routine is the same for all individuals. In our regression model we used ``ramped half-and-half" method (Wolfgang, 1998) that combines the previous two methods to get maximum diversity in the initial population as follows:

For *i* from 2 to max-initial-depth:

1. Generate $\left(\frac{50}{\max - initial - depth - 1}\right)$ % of the population using the ``full" method with maximum depth *i*.
2. Generate $\left(\frac{50}{\max - initial - depth - 1}\right)$ % of the population using the ``grow" method with maximum depth *i*.

4.2 The Fitness Function

In regression model the fitness of a program or tree reflect the quality of approximation of the experimental data by a current expression represented by a tree. The fitness of program in the regression model is measured as follows. Let $\phi(x)$ be the program in hand and the working data stored in arrays $x[]$ and $y[]$ representing input and output of test points which lie upon the curve we are attempting to match. The fitness of $\phi(x)$ with respect to the data is the ratio:

$$F(\phi) = \frac{1}{1 + \sum_{i=1}^{n} |(\phi(x[i]) - y[i])|} \qquad (1)$$

4.3 Genetic Operators

Trees (programs) in our model evolve through the action of two basic genetic operators: crossover and mutation. *Crossover* (see Figure 1) is implemented as follows:

- Select two trees randomly from the whole population,
- Within each of these tress, randomly select one node,
- Swap the sub-trees under the selected nodes, thus generating two offsprings belonging to the new population

After crossover, a small number of random trees are changed using sub-tree mutation (see Figure 2):

It is clear from the expression of the fitness that when $\sum_{i=1}^{n} |(\phi(x[i]) - y[i])|$ is closer to 0, we get a fitter program (i.e., $F(\phi)$ =1).

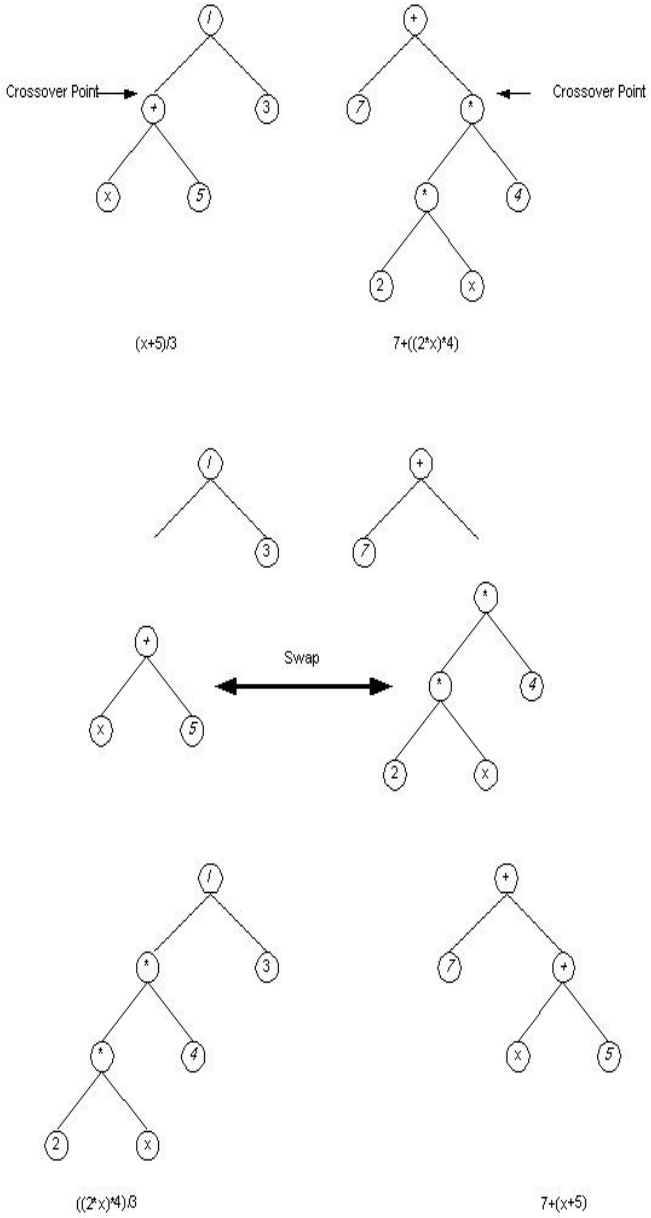

Fig 1: Crossover Process.

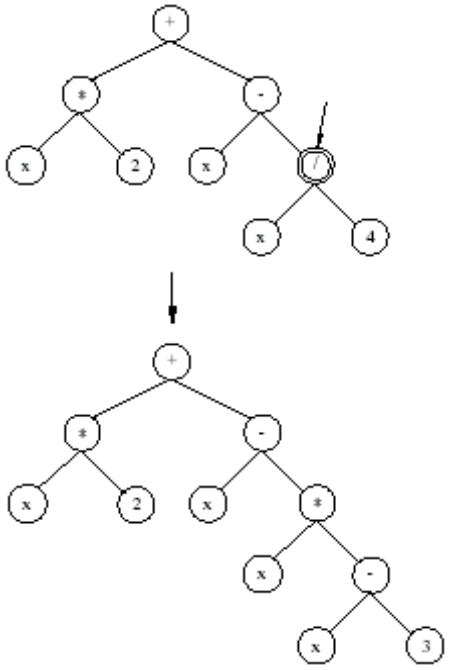

Figure 2: The Mutation Process.

5. THE SYMBOLIC APPROXIMATION PROCESS

The following algorithm is used for symbolic regression:

1. Start with a randomly generated population of trees (programs) composed of elements from the functional set and the terminal set. The root node of every tree must be restricted to a function. If a terminal is chosen, that node is an endpoint of the tree. To limit the complexity of the initial trees, an input parameter defines the maximum depth of the tree.
2. Iteratively perform the following steps on the population until the termination criterion has been satisfied:

a. Calculate the fitness of each tree (program) according to equation (1).

b. Sort the population according to the fitness.

c. Create a new population:

i. In the reproduction stage kill a small percentage of the programs with the worst fitness.

ii. Fill up the population with the surviving trees according to binary tournament selection (Thomas, 1996).

iii. *S*elect a pair of trees from the current population. The same tree can be selected more than once to become a parent. Recombine sub-trees using the *crossover* operation.

iv. with probability P_C taken as 0.9. Two new offsprings are inserted into the new population. Crossover takes place starting from the second node, not the root, to avoid duplication of trees.

v. With probability P_m taken as 0.3, *mutate* a randomly selected tree at a randomly selected point with a randomly generated sub-tree.

3. Check the termination criterion, if not satisfied, perform the next iteration.

The flowchart of the symbolic regression algorithm is shown in Figure 3.

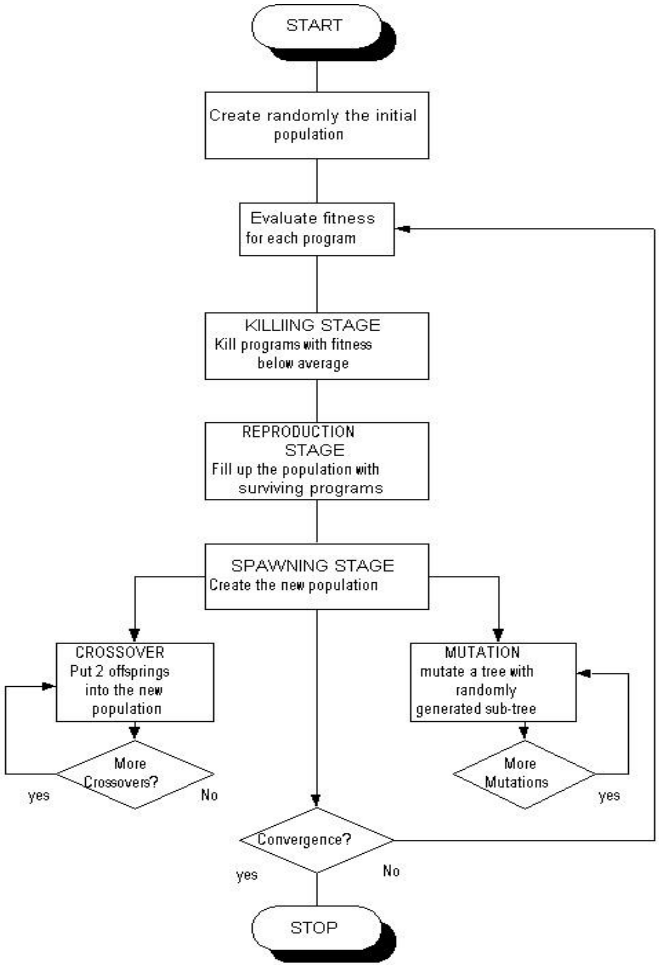

Fig 3: Flowchart of the Symbolic Regression Algorithm.

6. EXPERIMENTS

In the following we will apply the GP-symbolic regression algorithm for deriving approximation formulae of common mathematical functions. In this paper, we only present the results approximations to the following functions: the sin(x) function, the cos(x) function, the negative exponential exp (-x), the positive exponential exp (x), and the tangent tan(x) function. The functions are approximated over the interval [-1, 1]. The training data consisted of 50 points, uniformly spaced over the interval of approximation. The parameter settings of the GP-symbolic regression algorithm are summarized in Table 1. The evolved approximations are displayed in Table 2.

Table 1. The Global Parameters of the GP-Symbolic Regression Algorithm.

Selection method	Tournament Selection 2
Fitness	equation 1
Stop Criterion	Maximum generations or perfect fit
Crossover rate	0.9

Mutation Rate	0.3
Population Size	500-15000
Maximum generations	50-150
Tree depth	10
Error Rate	0.1-0.001

Table 2. Evolved Approximations.

Functions	**Evolved Approximations**	**Fitness**
sin(x)	$P_1(x)=\frac{6x-x^3}{6}$	0.999781
cos(x)	$P_2(x)=\frac{-7x^2+18}{2x^2+18}$	0.999851
Exp(x)	$P_3(x)=\frac{-x^3-9x^2-27x-36}{9x-36}$	0.999644
Exp(-x)	$P_4(x)=\frac{x^2-3x+4}{x+4}$	0.988904
Tan(x)	$P_5(x)=\frac{15x+5x^3+3x^5}{15}$	0.999554

We evaluated our evolved approximations with padé approximations (Baker, 1975). A padé approximation to f(x) on [a, b] is the quotient of two polynomials $P_n(x)$ and $Q_m(x)$ of degrees n and m, respectively. We use the notation $R_{n,m}(x)$ to denote this quotient: $R_{n,m}(x)=\frac{P_n(x)}{Q_m(x)}$. We have calculated all padé approximations of the approximated functions over the interval [-1, 1] using the Maple symbolic mathematics package as shown in Table 3. Tables 4 through 8 display the numerical calculated values for each genetically evolved approximation and corresponding padé approximation.

Table 3. Padé Approximations of the Approximated Functions.

Function	**Padé Approximation**
Sin(x)	$R_{3,2}(x)=\frac{x-\frac{7}{60}x^3}{1+\frac{1}{20}x^2}$
Cos(x)	$R_{4,4}(x)=\frac{15120-6900x^2+313x^4}{15120+660x^2+13x^4}$
Exp(x)	$R_{2,3}(x)=\frac{1+\frac{2}{5}x+\frac{1}{20}x^2}{1-\frac{3}{5}x+\frac{3}{20}x^2-\frac{1}{60}x^3}$
Exp(-x)	$R_{3,2}(x)=\frac{1-\frac{3}{5}x+\frac{3}{20}x^2-\frac{1}{60}x^3}{1+\frac{2}{5}x+\frac{1}{20}x^2}$
Tan(x)	$R_{1,4}(x)=\frac{-\frac{2}{21}x^3+x}{1-\frac{3}{7}x^2+\frac{1}{105}x^4}$

Table 4. Numerical Evaluation of $P_1(x)$ and $R_{3,2}(x)$.

X	**$P_1(x)$**	**$R_{3,2}(x)$**	**$\lvert P_1(x)-R_{3,2}(x)\rvert$**
-0.332000	-0.319802	-0.325934	0.006132
0.000000	0.000000	0.000000	0.000000
-0.106000	-0.105603	-0.105802	0.000199
0.096000	0.095705	0.095853	0.000148
0.058000	0.057935	0.057967	0.000033
-0.086000	-0.085788	-0.085894	0.000106
0.082000	0.081816	0.081908	0.000092
0.076000	0.075854	0.075927	0.000073
-0.158000	-0.156685	-0.157343	0.000658
-0.066000	-0.065904	-0.065952	0.000048

Table 5. Numerical Evaluation of $P_2(x)$ and $R_{4,4}(x)$.

X	**$P_2(x)$**	**$R_{4,4}(x)$**	$\lvert P_2(x)-R_{4,4}(x)\rvert$
-0.044000	0.999032	0.999032	0.000000
-0.072000	0.997409	0.997409	0.000000
-0.128000	0.991823	0.991819	0.000004
-0.158000	0.987553	0.987544	0.000009
-0.106000	0.994389	0.994387	0.000002
-0.214000	0.977218	0.977189	0.000029
0.096000	0.995397	0.995396	0.000001
0.082000	0.996641	0.996640	0.000001
-0.140000	0.990221	0.990216	0.000005
0.148000	0.989075	0.989068	0.000007

Table 6. Numerical Evaluation of $P_3(x)$ and $R_{2,3}(x)$.

X	$P_3(x)$	$R_{2,3}(x)$	$\lvert P_3(x)-R_{2,3}(x)\rvert$
-0.066000	0.936135	0.936131	0.000004
0.000000	1.000000	1.000000	0.000000
0.044000	0.956955	0.956954	0.000001
-0.284000	0.753060	0.752767	0.000294
-0.018000	0.982161	0.982161	0.000000
0.076000	1.078956	1.078963	0.000006
0.058000	1.059712	1.059715	0.000003
-0.086000	0.917603	0.917594	0.000009
-0.426000	0.654064	0.653117	0.000947
0.024000	1.024290	1.024290	0.000000

7. CONCLUSIONS

This paper has described a GP- based symbolic regression system to the approximation of functions and a number of experiments that were carried out using it. We have presented several successful experiments involving the application of genetic programming to the automated discovery of symbolic approximation formulae. In particular, we have presented positive results in applying genetic programming to the discovery of approximations to common functions, and have shown that evolved approximations can compare favorably with padé approximations. Based on these results, we consider genetic programming to be a powerful and effective technique for the automated discovery of approximation formulae.

Table 7. Numerical Evaluation of $P_4(x)$ and $R_{3,2}(x)$.

X	$P_4(x)$	$R_{3,2}(x)$	$\lvert P_4(x)-R_{3,2}(x)\rvert$
-0.066000	1.068215	1.068227	0.000012
-0.044000	1.044979	1.044982	0.000004
0.000000	1.000000	1.000000	0.000000
-0.044000	1.044979	1.044982	0.000004
-0.072000	1.074640	1.074655	0.000016
-0.018000	1.018163	1.018163	0.000000
0.058000	0.943658	0.943650	0.000008
0.024000	0.976286	0.976286	0.000001
0.082000	0.921294	0.921272	0.000023
0.074000	0.928688	0.928672	0.000017

Table 8. Numerical Evaluation of $P_5(x)$ and $R_{3,2}(x)$.

X	$P_5(x)$	$R_{3,2}(x)$	$\lvert P_5(x)-R_{1,4}(x)\rvert$
-0.128000	-0.128706	-0.128704	0.000002
-0.158000	-0.159334	-0.159328	0.000006
-0.072000	-0.072125	-0.072125	0.000000
-0.106000	-0.106400	-0.106399	0.000001
0.310000	0.320503	0.320328	0.000175
-0.326000	-0.338285	-0.338062	0.000223
-0.086000	-0.086213	-0.086213	0.000000
-0.426000	-0.454576	-0.453789	0.000787
-0.234000	-0.238411	-0.238367	0.000045
-0.214000	-0.217357	-0.217328	0.000029

8. REFERENCES

[1] Andre, D. & Koza, J. R. (1996) *Parallel Genetic Programming: A scalable implementation using the transputer network architecture*. In P. J. Angeline and K. E. Kinnear, Jr. (eds.), *Advances in Genetic Programming 2*, 317-338. Cambridge, MA: MIT Press.

[2] Andre, D. & Koza, J. R. (1996) *Parallel Genetic Programming: A scalable implementation using the transputer network architecture*. In P. J. Angeline and K. E. Kinnear, Jr. (eds.), *Advances in Genetic Programming 2*, 317-338. Cambridge, MA: MIT Press.

[3] Baker, G. A (1975) *Essentials of Padé Approximants*. New York: Academic Press.

[4] Blickle, T. & Thiele, L. (1995) *A Comparison of Selection Schemes Used in Genetic Algorithms*. TIK-Report 11, TIK Institut fur Technische Informatik und Kommunikationsnetze, Computer Engineering and Networks Laboratory, ETH, Swiss Federal Institute of Technology.

[5] Chellapilla, K. (1997) *Evolving Computer Programs without Subtree Crossover*. IEEE Transactions on Evolutionary Computation **1**(3):209-216.

[6] Faires, D. & Burden, R. (1998) *Numerical Methods*, Brooks/Cole Publishing Company, USA.

[7] Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*. Ann Arbor, Michigan: The University of Michigan Press.

[8] Iba, H., Kurita, T., de Garis, H., & Sato, T. (2019) *System identification using structured genetic algorithms*. In Stefanie Forrest, editor, Proceedings of the Fifth International Conference on Genetic Algorithms, pages 279-286, San Mateo, CA, Morgan Kaufmann Publishers.

[9] Keane, M. A., Koza, J. R. & Rice, J. P. (1993) *Finding an impulse response function using genetic programming*. In *Proceedings of the 1993 American Control Conference*, 3:2345-2350.

[10] Koza, J. R. (1990a) *Genetic Programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Stanford University

Computer Science Department technical report STAN-CS-90-1314.

[11] Koza, J. R. (1990b). *A Genetic Approach to Econometric Modeling*. Presented at Sixth World Congress of the Econometric Society, Barcelona, Spain.

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

[13] Koza, J.R., Bennett III, F.H. Andre, D. and Keane, M.A. (1999), "*Genetic Programming III: Darwinian Invention and Problem Solving*", Morgan Kaufmann.

[14] Luke, S. & Spector, L. (1997) *A Comparison of Crossover and Mutation in Genetic Programming.* In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo (eds.), Genetic Programming 1997: Proceedings of the Second Annual Conference, 240-248. San Mateo, CA: Morgan Kaufmann.

[15] Nordin, P. (1997) *Evolutionary Program Induction of Binary Machine Code and its Applications*. PhD thesis, der Universitat Dortmund am Fachereich Informatik.

[16] Ryan, C., Collins, J. & O'Neill, M. (2018). *Grammatical Evolution: evolving programs for an arbitrary language*. In W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty (eds.), Proceedings of the First European Workshop on Genetic Programming, 1391:83-95. New York: Springer-Verlag.

[17] Thomas, B. (1996) *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, Inc.

[18] Wolfgang, B., Peter, N., Robert, E. K., & Frank, D. F. (1998) *Genetic Programming- an introduction: on the automatic evolution of computer programs and its applications*. Morgan Kaufmann Publishers, San Francisco.