Submitted November 19, 1989 to 2nd Conference Economics and Artificial Intelligence, July 2-6, 1990, Paris

DISCOVERING AN ECONOMETRIC MODEL BY . GENETIC BREEDING OF A POPULATION OF MATHEMATICAL FUNCTIONS

John R. Koza
Computer Science Department
Stanford University
Stanford, California 94305
USA
Koza@Polya.Stanford.Edu
415-94l-0336

Keywords: 2.2 Macro- and Micro-Economic Models

DISCOVERING AN ECONOMETRIC MODEL BY . GENETIC BREEDING MATHEMATICAL FUNCTIONS

John R. Koza
Computer Science Department
Stanford University
Stanford, California 94305 USA
Koza@Polya.Stanford.Edu
415-94l-0336

Abstract: This paper reports on a new "genetic computing" paradigm for solving problems. In this new artificial intelligence paradigm, populations of computer programs are genetically bred using genetic operations patterned after sexual reproduction and based on Darwinian principles of reproduction and survival of the fittest. The paradigm is illustrated by rediscovering the well-known multiplicative (non-linear) "exchange equation" M=PQ/V.

Abstract: This paper reports on a new "genetic computing" paradigm for solving problems. In this new artificial intelligence paradigm, populations of computer programs are genetically bred using genetic operations patterned after sexual reproduction and based on Darwinian principles of reproduction and survival of the fittest. The paradigm is illustrated by rediscovering the well-known multiplicative (non-linear) "exchange equation" M=PQ/V.

Keywords: 2.2 Macro- and Micro-Economic Models

1.0 THE "SYMBOLIC REGRESSION" PROBLEM

An important problem in many areas of science, including economics, is finding the empirical relationship underlying the observed numeric values of various variables measuring a system. Langley and Zytkow (1989) discuss efforts to apply artificial intelligence techniques to this problem.

In many conventional methods for modeling, one begins by selecting the size and shape of the mathematical model. Often this choice is a linear model. After making this selection, one then tries to find the values of certain constants and coefficients required by the particular model so as to attain the best fit between the observed data and the model. But, in many cases, the most important issue is the size and shape of the mathematical model itself. Thus, the real problem involves finding the entire functional form for the model (i.e. both the functional form and the numeric coefficients and constants) that best fits observed empirical data. The new "genetic computing" (hierarchical genetic algorithm) paradigm described in this paper searches a space of functional forms to solve this problem of “symbolic regression” to find a functional form fitting the observed data.

2.0 BACKGROUND ON GENETIC ALGORITHMS

Genetic algorithms are mathematical algorithms that transform populations of individual mathematical objects (typically fixed-length binary character strings) into new populations using operations patterned after natural genetic operations such as sexual recombination (crossover) and fitness proportionate reproduction (Darwinian survival of the fittest). Genetic algorithms begin with an initial population of individuals (typically randomly generated) and then iteratively (1) evaluate the individuals in the population for fitness with respect to the problem environment and (2) perform genetic operations on various individuals in the population to produce a new population. Professor John Holland of the University of Michigan presented the pioneering formulation of genetic algorithms for fixed-length character strings in <u>Adaptation in Natural and Artificial Systems</u> (Holland 1975). Holland established, among other things, that genetic algorithms operating on strings are a mathematically near optimal approach to adaptation when the adaptive process is viewed as a set of simultaneous multi-armed slot machine problems requiring an optimal allocation of future trials given currently available information. Recent work on genetic algorithms can be surveyed in Goldberg (1989).

3.0 THE “GENETIC COMPUTING” PARADIGM

In the recently developed new “genetic computing” (hierarchical genetic algorithm) paradigm, the individuals in the genetic population are hierarchical compositions of functions (i.e. S-expressions in the LISP programming language) rather than the fixed-length character strings.

Populations of LISP S-expressions (computer programs) can be genetically bred to solve a variety of different problems in several different areas of artificial intelligence and symbolic processing (Koza 1989, Koza 1990, Koza and Keane 1990), including (1) optimal control (e.g. finding a non-linear function that is a time-optimal control strategy for balancing a broom), (2) sequence induction (e.g. inducing a recursive computational procedure for the Fibonacci sequence), (3) planning problems (e.g. a robotic action sequence for stacking blocks in order), (4) automatic programming (e.g. discovering a computational procedure for solving pairs of linear equations, solving quadratic equations for complex roots, and discovering trigonometric identities), (5) machine learning of functions (e.g. learning the exclusive-or function, the parity function, and a Boolean multiplexer function), (6) pattern recognition (e.g. translation-invariant recognition of a simple one-dimensional shape in a linear retina), and (7) various computational problems (e.g. finding an infinite series for the exponential function and finding primes).

Problems such as the above can be viewed as a search for a compositions of functions in the space of possible compositions of functions. In particular, the search space is the hyperspace of LISP S-expressions (i.e. composition of functions, computer programs, control strategies, robotic action plans) composed of various standard mathematical functions, various logical functions, various standard programming operations, and various domain specific functions and atoms appropriate for the given problem.

In fact, problems such as the above can be expeditiously solved only if the flexibility found in computer programs is available. This flexibility includes the ability to perform iterations and recursions to achieve the desired result, to perform alternative computations conditioned on the outcome of intermediate calculations, to perform computations on variables of many different types, and to define and subsequently use computed values and sub-programs.

As will be seen, the composition of functions required to solve the problems described above will, in each case, emerge from a simulated evolutionary process using genetic operations patterned after sexual reproduction and based on Darwinian principles of reproduction and survival of the fittest.

In each case, this process will start with an initial population of randomly generated LISP S-expressions composed of functions and atoms appropriate to the problem domain (and at least sufficient to solve the problem).

The raw fitness of any such LISP S-expression in a given problem environment can be naturally measured by the sum of the distances (taken for all the environmental cases in the test suite) between the point in the solution space created by the S-expression and the correct point in the solution space. The closer this sum of differences is to zero, the better the S-expression. Predictably, randomly generated initial S-expressions will have exceedingly poor fitness. Nonetheless, some individuals in the population will be somewhat more fit than others. And, in the valley of the blind, the one-eyed man is king.

Then, a process of sexual reproduction among two parental S-expressions will be used to create offspring S-expressions. At least one of two parental S-expressions is selected in proportion to fitness using Darwinian principles of reproduction and survival of the fittest. The offspring S-expressions resulting from the process of sexual reproduction are composed of sub-expressions ("building blocks") from their parents. In particular, the crossover (recombination) operation creates new offspring S-expressions by exchanging sub-trees (i.e. sub-lists) between the two parents at randomly selected points. Then, the new population of offspring (i.e. a new generation) replaces the current population (i.e. parents).

This process using the operations of Darwinian fitness proportionate reproduction and genetic crossover (recombination) tends to produce populations which, over a period of generations, exhibit increasing average fitness in dealing with their environment and which can robustly (i.e. rapidly and effectively) adapt to changes in their environment.

The dynamic variability of the size and shape of the computer programs that are incrementally developed along the way towards a solution is also an essential aspect of the paradigm. In each case, it would be difficult and unnatural to try to specify the size and shape of the eventual solution in advance. Moreover, the specification in advance of the size and shape of the solution to the problem narrows the window by which the system views the world and might well preclude finding the solution to the problem. The results of the genetic computing paradigm process are inherently hierarchical.

4.0 AN EXAMPLE --- THE “EXCHANGE EQUATION”

The problem of discovering such empirical relationships can be illustrated by the well-known econometric “exchange equation” M=PQ/V, which relates the money supply M, price level P, gross national product Q, and the velocity of money V of an economy. Suppose that our goal is to find the relationship between quarterly values of the money supply M2 and the three other elements of the equation.

In particular, suppose we are given the 112 quarterly values (from 1961:1 to 1988:4) of four econometric time series. The first time series is “GNP82” (i.e. the annual rate for the United States gross national product in billions of 1982 dollars). The second time series is “GD” (i.e.the gross national product deflator normalized to 1.0 for 1982). The third series is “FYGM3” (i.e. the monthly interest rate yields of 3-month Treasury bills, averaged for each quarter). The fourth series is “M2” (i.e. the monthly values of the seasonally adjusted money stock M2 in billions of dollars, averaged for each quarter). The time series used here were obtained from the CITIBASE data base of machine-readable econometric time series (Citybank 1989) and were entered into an Apple Micro-Explorer computer (i.e. a Macintosh II computer with a Texas Instruments LISP board) and then ported into an Explorer LISP machine using an Ethernet.

The actual long-term historic postwar value of the M2 velocity of money in the United States is 1.6527 (Hallman _et_. _al_. 1989) so that the "correct" solution is the multiplicative (non-linear) relationship

$$M2 = \frac{GD * GNP82}{1.6527}$$

However, we are not told _a priori_ whether the functional relationship between the given observed data (the three independent variables) and the target function (the dependent variable M2) is linear, multiplicative, polynomial, exponential, logarithmic, or otherwise. The set of available functions for this problem is F = {+, -, *, %, EXP, RLOG}. The set of available atoms for this problem is A = {GNP82, GD, FYGM3}. They provide access to the values of the 28-year time series for particular quarters. We are not told that the addition, subtraction, exponential, and logarithmic functions and the time series for the 3-month Treasury bill yields (FYGM3) are irrelevant to the problem.

Note that the restricted logarithm function RLOG used here is the logarithm of the absolute value and returns 0 for an argument of 0. Note also that the restricted division function % returns a value of 0 if division by 0 is attempted.

A population size of 300 LISP S-expressions was used. In generating the initial random population (generation 0), various random real-valued constants were inserted at random as atoms amongst the initial random LISP S-expressions. The initial random population was, predictably, highly unfit. In one fairly typical run, none of the 300 individual S-expressions in the initial random population came within 3% of any of the 112 environmental data points in the time series for M2. The sum of errors between that best S-expression and the actual time series was very large (88448). Similarly, the best individuals in generations 1 through 4 came close to the actual time series in only a small number of cases (i.e. 7, 2, 3, and 5 of the 112 cases, respectively) and also had large sum of errors (72342, 70298, 26537, 23627). However, by generation 6, the best individual came close to the actual time series in 41 of the 112 environmental cases and had an sum of errors of 6528.

In generation 9, the following S-expression for M2 emerged:
(* GD (% GNP82 (% (% -.587 0.681) (RLOG -0.587)))).
Note that this S-expression for M2 is equivalent to
(% (* GD GNP82) 1.618),
or, more familiarly,

$$M2 = \frac{GD * GNP82}{1.618}$$

The S-expression discovered in the 9th generation comes within 3% of the actual values of M2 for 82 of the 112 points in the 28-year time series. The sum of the absolute errors between the S-expression discovered and the 112 points of the 28-year time series is 3765.2. The S-expression discovered here compares favorably to the "correct" "exchange equation" M=PQ/V (with a value of V of 1.6527) which had a sum of errors of 3920.7 and which came within 3% of the actual time series data for 73 of 112 points in the 28-year time period studied.

REFERENCES

Citibank, N. A. CITIBASE: Citibank Economic Database (Machine Readable Magnetic Data File), 1946 - Present. New York: Citibank N.A. 1989.

Goldberg, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley l989.

Hallman, Jeffrey J., Porter, Richard D., Small, David H. M2 per Unit of Potential GNP as an Anchor for the Price Level. Washington,DC: Board of Governors of the Federal Reserve System. Staff Study 157, April 1989.

Holland, John H. Adaptation in Natural and Artificial Systems, Ann Arbor, MI: University of Michigan Press 1975.

Koza, John R. Hierarchical genetic algorithms operating on populations of computer programs. Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI). San Mateo, CA: Morgan Kaufman 1989.

Koza, John R. Genetic computing using hierarchical genetic algorithms. Machine Learning. In press. 1990.

Koza, John R. and Keane, Martin. Cart Centering and broom balancing by genetically breeding populations of control strategy programs. Proceedings of International Joint Conference on Neural Networks - January 1990.

Langley, P. and Zytkow, J. Data-driven approaches to empirical discovery.Artificial Intelligence.40(1989)283-312.