Evolutionary Computation in Economics and Finance: Models of Decision-Making Systems

Cheong, Andrew W. Sinha, Supriyo

30 April 2008

Abstract

We simplify complex systems such as an economy into better-understood and organized decision-making systems, such as a stock market. We use evolutionary computation to model this system, evolving agent behaviour that rivals that of our own. We offer extensions to the model that make it more realistic, and further developments that could help us understand an economy bottom-up, by adding one variable at a time and observing its effects.

1 Introduction

Economics is "the science which studies *human behaviour* as a relationship between ends and scarce means which have alternative uses."[1] Human behaviour is a widely variant property composed of diverse tastes and preferences, rational and emotional reasoning, and information and expectations strategies. Thus, in economics, we naturally develop complicated multivariate problems with large, possibly even unbounded, solution spaces.

One field whose methods are proven to be effective for such problems is *evolutionary computation*. Evolutionary computation is a subset of computer science that applies the principles of biological evolution to solve computational problems.[2] The general term for approaches in evolutionary computation are known as *evolutionary algorithms*. These algorithms begin by generating a "population" of randomly assembled candidate solutions. Then, the algorithms apply biological mechanisms such as natural selection, reproduction, mutation, and recombination in order to evolve better solutions over subsequent "generations". (In section 2, we provide step-by-step explanations of evolutionary algorithms, accompanied by detailed and illustrated examples.)

In this paper, we wish to show that evolutionary algorithms are not only a set of effective optimization techniques, but also an ideal framework for studying economics as a system of *organized complexity*, as opposed to one of simplicity or disorganized complexity.[3] A system of *simplicity* involves only a few variables, the history of which is manageable, and the future of which is predictable. For example, consider the motion of three balls on a billiards table; with our knowledge of physics, the path of each ball and the resultants of each collision are calculable. In contrast, a system of *disorganized complexity* is a system of many variables, perhaps numbering thousands or even billions. For such systems, it is the very *multitude* of variables that allows us to conduct any useful analysis. Methods in statistical mechanics can take advantage of such large numbers to reveal meaningful aggregate information about the system. For example, on a billiards table with billions of moving balls, it is unmanageable to track the history or predict the future of each individual ball; however, it *is* feasible to determine with great precision, for instance, the *average* rate of collisions. Meanwhile, a system of *organized complexity* may still involve many variables, but shows the essential feature of *organization* among its variables. The necessity of a such a middle-ground between simplicity and disorganized complexity is best argued by Warren Weaver in his influential 1948 publication, "Science and Complexity":

How can currency be wisely and effectively stabilized? To what extent is

> it safe to depend on the free interplay of such economic forces as supply and demand? To what extent must systems of economic control be employed to prevent the wide swings from prosperity to depression? [These are complex problems that] involve analyzing systems which are organic wholes, with their parts in close interrelation. [...] These problems—and a wide range of similar problems in the biological, medical, psychological, economic, and political sciences—are just too complicated to yield to the old nineteenth-century techniques which were so dramatically successful on two-, three-, or four-variable problems of simplicity. These new problems, moreover, cannot be handled with the statistical techniques so effective in describing average behaviour in problems of disorganized complexity.[3]

Studying economics solely as a simple system does not enable us to cross the boundary between theory and application. For example, in microeconomic analyses, we often consider variables such as producers and consumers, supply and demand, prices and quantities, and costs and benefits. However, these variables are not as simple in reality as we deem in theory; imperfect competition, irrational behaviour, and information asymmetry are among the culprits. Without consideration for such intricacies, our formulation of economics might be applicable in some cases, while irrelevant in others. On the other hand, studying economics solely as a disorganized complex system does not provide us with the necessary dexterity for engineering and fine-tuning a national economy. For example, in macroeconomic analyses, we often use econometric techniques to determine correlated and causal relationships between various economic indicators and other time-series, cross-sectional, or panel observations. However, statistics alone cannot explain *why* or *how* certain monetary or fiscal policies produce desired effects, while others fail to do so. We must fill in these gaps with inferences and deductions based on our human experience and intuition.

In the past century, much has been contributed toward the formulation of economics—*human behaviour*, even—as a system of *organized* complexity. For example, we have come to understand the "Tragedy of the Commons" as an *organized* process based on human behavioural motives and social regulations (or the lack thereof); we know that by levying taxes or imposing limits, we can eliminate the over-exploitation of common goods and maximize net social benefits.[4] Meanwhile, we have connected various relationships between interest rates, real money balances, and expenditure behaviour in order to derive an *organized* theory of aggregate demand (namely, the IS/LM model in Keynesian macroeconomics).[5] We hope to one day possess a comprehensive understanding of economics as a system of organized complexity.

For now, we wish to demonstrate the usefulness of evolutionary algorithms in deciphering the underlying mechanics of an economy, especially as a system of organized human behaviour. Unfortunately, the techniques involved are numerous, and the scope is much too large to be contained in this paper. Thus, we propose to narrow our focus from "economic systems" to "decision-making systems". We will show how evolutionary algorithms can help us discover organization in seemingly disorganized complex systems such as decision-making systems. We then offer topics for further discussion, suggesting that similar analysis and development of evolutionary algorithms can aid the study of numerous other aspects of the economy.

2 Background

2.1 Genetic Algorithms

First, let us familiarize ourselves with evolutionary algorithms. Recall the basic mechanisms of evolution. There exists a population, varied in genetic makeup. For example, consider a population of snails, whose shell thicknesses vary. The most "fit" are the snails with thicker shells, since they are less subject to predation. These snails are naturally selected to reproduce. Assuming that shell thickness is a genetically inherited trait, parent genes cross-over and mutate to produce offspring genes. It follows that the new generation should have, on average, greater shell thicknesses than the previous generation. Over time, repetition of this process leads to individuals who are more and more fit to their environment.

Evolutionary algorithms borrow these ideas of *genetic variation*, *natural selection*, and *cross-over and mutation* in order to "evolve" the best solutions over time. The most common form of an evolutionary algorithm is a *genetic algorithm.* In a genetic algorithm, we initially generate a "population" of randomly assembled candidate solutions, each of which is somehow encoded as a "gene". Based on the fitness criteria (as specified by the problem), candidate solutions are selected either for survival or for elimination. Those who survive go through recombination procedures (i.e. cross-over and mutation), which leave us with a new generation of "offspring" solutions that are better-"fit", on average, than the previous generation. Over time, repetition of this process leads to "individuals" who are better solutions to the given problem.

Given a specific problem to solve, a genetic algorithm may be constructed in three steps.

i. Define the space of candidate solutions and a way to encode each as a "gene"

ii. Define a way to determine the fitness of each candidate solution

iii. Define the genetic operations that create new candidate solutions

Example: The Transportation Problem

Consider the transportation problem. Units of a single product are to be shipped from m sources (e.g. warehouses) to n destinations (e.g. retailers). Each source i has a supply of a_i units. Each destination j has a supply of b_j units. The cost of shipping from i to j is $c_{i,j}$. Given this information, how many units, $x_{i,j}$, must we ship from each source i to each destination j, in order to minimize total shipping costs? In figure 1, we provide a simple example.

There already exist several linear programming methods, for example the *transportation tableau algorithm*, that are able to solve the transportation problem.[6] However, for demonstrative purposes, we construct a genetic algorithm for solving the transportation problem.

Define the space of candidate solutions and a way to encode each as a "gene". Recall that the solution we wish to find is a set of non-negative integers $x_{i,j}$ that describe how many units to ship from each source i to each destination j. A naïve genetic algorithm might consider *any* such set of non-negative integers to be

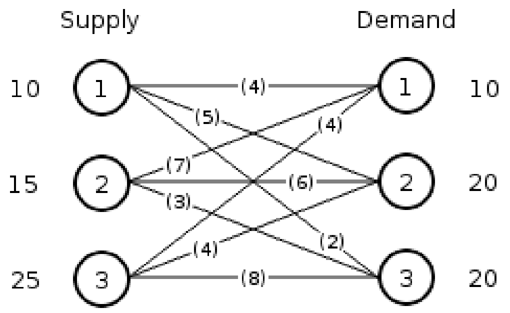

Figure 1: A simple transportation problem with $m = 3$ sources and $n = 3$ destinations. Supplies are listed on the left, demands are listed on the right, and shipping costs are written in parentheses over their respective routes. For example, source 2 has a supply of $a_2 = 15$, destination 1 has a demand of $b_1 = 10$, and the cost of shipping one unit from source 2 to destination 1 is $c_{2,1} = 7$. Note that in this simplified transportation problem, we force the total supply to equal the total demand.

a candidate solution. However, this strategy entails a solution space that is terribly large; it does not take advantage of any *constraints* that are available to us. For example, the solution must be *feasible*. That is, each source cannot ship more than its available supply, and each destination cannot receive less than its demand. One such feasible solution is illustrated in figure 2. By limiting candidate solutions to only those whose set of shipments, $x_{i,j}$, respect the feasibility condition, we significantly reduce the solution space.*

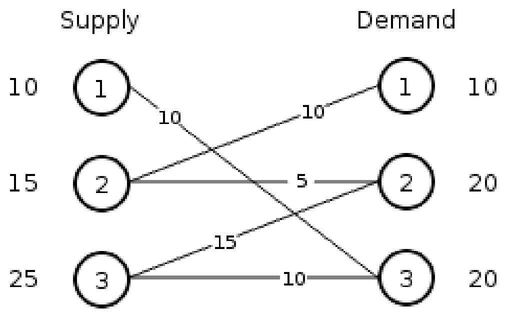

Figure 2: A feasible solution to the transportation problem. (Note that in contrast to figure 1, the non-parenthesized values over the shipping routes represent units to be shipped, $x_{i,j}$, not shipping costs.) In a feasible solution, each source cannot ship more than its available supply, and each destination cannot receive less than its demand. For example, $x_{2,1} + x_{2,2} + x_{2,3} = 10 + 5 + 0 = a_2$. The naïve algorithm does not respect these conditions.

We can even further constrain the solution space, given some advanced insight regarding the transportation problem: *a cost-minimizing solution always ships the*

*The feasibility condition, in mathematical notation, is $\sum_j x_{i,j} = a_i, \forall i$ and $\sum_i x_{i,j} = b_j, \forall j$.

maximum number of units from each source to each destination.[†] This observation has one very interesting implication. If each source must ship the maximum number of units to each demand, then we do not need to *specify* how many units, $x_{i,j}$, to ship from each source to each destination; we need only to specify *an order of sources and destinations* to ship. The *amount* to ship, then, follows naturally from the maximum-shipping rule. To illustrate, we return to our sample problem. Suppose we ordered the sources by $i = 2, 3, 1$ and the destinations by $j = 3, 1, 2$. As shown in figure 3, by selecting an ordering of sources and destinations, and applying the maximum-shipping rule, we have also determined a *unique* feasible solution. One of these orderings is *guaranteed* to be a cost-minimizing solution. We have thus limited our solution space to only the permutations of sources and destinations, or $(m! \times n!)$ possible solutions.

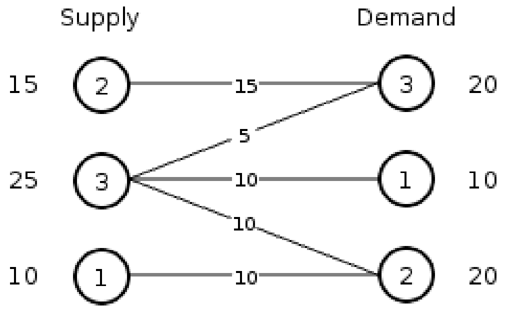

Figure 3: The transportation problem, reordered. The maximum number of units we may ship from source 2 to destination 3 is $x_{2,3} = 15$ units, since source 2 supplies only 15 units. Then, the maximum number of units we may ship from source 3 to destination 3 is $x_{3,3} = 5$ units, since destination 3 only demands 20 units. Next, the maximum number of units we may ship from source 3 to destination 1 is $x_{3,1} = 10$ units, since destination 1 only demands 10 units. This repeated process, the "maximum-shipping rule", maps each ordering of sources and destinations to a unique feasible solution. Using figure 1 as a reference, and the formula given in equation (1), we can calculate the total shipping cost of this particular solution to be 215.

Now that we have defined the solution space, we must define a way to encode each candidate solution as a "gene". Let us simply concatenate the orderings of sources and destinations. For example, for the sources ordered by $i = 2, 3, 1$ and destinations ordered by $j = 3, 1, 2$, we define the representative gene to be `231312`.

Define a way to determine the fitness of each candidate solution. In the transportation problem, the lesser the total cost of shipping, the better-fit the individual. To compute the total cost of an individual, we convert the orderings of sources and destinations into values for $x_{i,j}$ using the maximum-shipping rule. Then, the total cost C is given by

[†] The proof is as follows. Suppose source i must ship a_i units. Let j^* denote the cost-minimizing destination. If j^* is the cost-minimizing destination, then j^* remains the cost-minimizing destination regardless the number of units shipped to it. Thus, source i will ship as many units to j^* as possible.

$$C = \sum_i \sum_j c_{i,j} x_{i,j}. \tag{1}$$

Define the genetic operations that create new candidate solutions. Given two parents, let us perform *cross-over* operations by swapping the source and destination orderings; for example, crossing parents $\texttt{231312}$ and $\overline{\texttt{132321}}$ would result in offspring $\texttt{231}\overline{\texttt{321}}$ and $\overline{\texttt{132}}\texttt{312}$. Meanwhile, let us perform *mutation* operations by swapping adjacent bits in either half of the gene; for example, $\texttt{2313}\widehat{\texttt{12}}$ might mutate to become $\texttt{2313}\widehat{\texttt{21}}$.

Having constructed the genetic algorithm, we are now prepared to *run* it. To begin, the algorithm produces an initial population of randomly assembled candidate solutions, or individuals. That is, random permutations of $\texttt{123}$ are combined to produce an initial gene pool. Then, the fitness of each individual is computed. Based on these fitnesses, we select a "mating", or parent, population. We briefly describe two common selection methods.

i. In *roulette selection*, individuals are selected into the mating population with a probability proportional to their fitness.

ii. In *tournament selection*, k individuals are randomly selected, and the fittest among the k individuals is selected into the mating population.

Both methods are repeated until the desired number of parents is selected. Then, the parents are crossed and mutated until the desired number offspring is obtained. (For a fixed-population genetic algorithm, we wish to generate as many offspring as there are parents.) This process is repeated until a *stopping condition* is met. For example, we might run the algorithm until 90% of the population possess identical genes (indicating that this particular gene was strongly favored during the selection process.)

In one instance, running our genetic algorithm with a fixed-population of 10 and a stopping condition of 90% identicality, we end up with 9 of 10 offspring in the 5th generation having identical genes of $\texttt{213312}$. Indeed, this is the cost-minimizing solution, as depicted in figure 4.

Although we began with a population of arbitrary candidate solutions, the population evolved toward an optimal solution.

2.2 Genetic Programming

In a genetic algorithm, each "individual" is a particular solution. However, in a genetic *program*, each "individual" is a program, or function. As one might imagine, this generalized technique is far more powerful than genetic algorithms. Conceptually, however, it is not much different from a genetic algorithm.

Example: The Phillips Curve

Suppose we hypothesize a relationship, whether linear or non-linear, between inflation π, expected inflation π_e, and unemployment u. We assume that these variables are

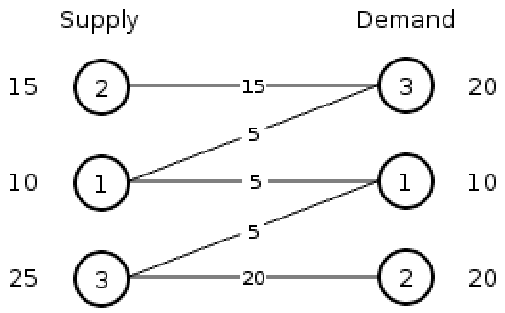

Figure 4: The cost-minimizing solution to the transportation problem, obtained from the genetic algorithm. The total shipping cost of this optimal solution, computed using figure 1 and equation (1), is 175.

related by certain operators: namely, addition (+), subtraction (–), multiplication (×), and division (÷). If sufficient data for π, π_e, and u are available, then we may employ genetic programming techniques to derive the relationship $\pi = f(\pi_e, u)$.

Analogous to constructing a genetic algorithm, first we **define the space of candidate solutions and a way to encode each as a "gene".** With no constraints at hand, we must leave the solution space—any function $f(\pi_e, u)$—as it is. As for encoding, however, one common representation scheme for functions is a *parse tree.* A parse tree is a graph of *operators* and *terminals.* For example, figure 5 is a parse tree for the function, $Ae^R + B - C^2$, with operators +, –, ×, eˆ, and ˆ, and terminals A, B, C, R, and 2.

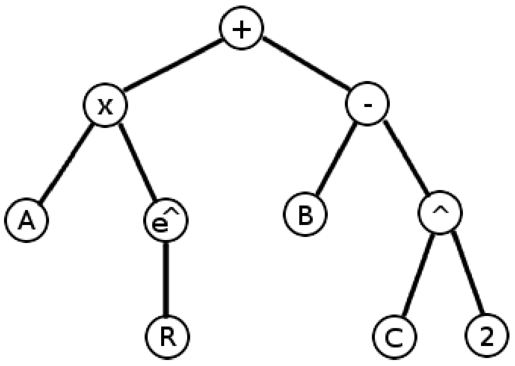

Figure 5: A parse tree encoding the formula, $Ae^R + B - C^2$. All of the intermediate nodes, i.e. +, –, ×, eˆ, and ˆ, are operators. All of the "leaf" nodes, i.e. A, B, C, R, and 2, are terminals.

Each individual in our genetic program will be encoded as a parse tree that represents the function $\hat{f}(\pi_e, u)$. The parse tree must contain the operators +, –, ×, and ÷, and the terminals π_e and u. We also include the terminal *rand*, which is a random number between 0 and 1, in order to allow for parameter variation. We hope to evolve a function that consistently estimates π.

Next, we **define a way to determine the fitness of each candidate solution.** For each individual (i.e. function) i, we "plug in" historical data for π_e and u to obtain an estimated rate of inflation $\hat{\pi} = \hat{f}(\pi_e, u)$. The lower the error residual $r_i = |\pi - \hat{\pi}|$, the better fit the individual.

Finally, we **define the genetic operations that create new candidate solutions.** Given two parents (two parse trees), we perform *cross-over* operations by swapping a subtree from each parent. We perform *mutation* operations by randomly replacing a node with a different function or terminal.

In one instance, running our genetic program with a fixed-population of 200 and a stopping condition of 80% identicality, we end up with 176 of 200 offspring in the 20th generation having identical genes, as drawn in figure 6.

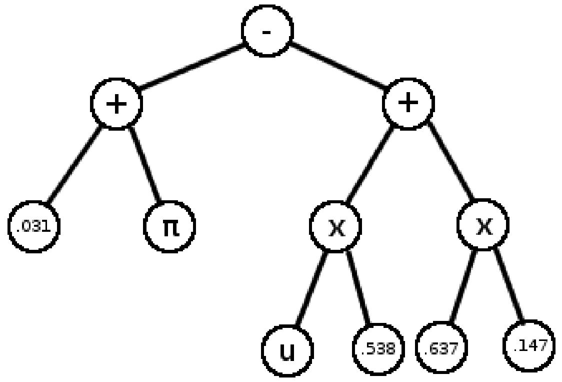

Figure 6: The resulting parse tree in one run of the genetic program.

The formula represented by figure 6 simplifies to

$$\hat{\pi} = \hat{f}(\pi^e, u) = \pi^e - 0.538u + 0.125. \tag{2}$$

The actual relation is known as the Phillips Curve, shown in equation (3), and also accounts for the natural rate of unemployment u_n and supply shock ν. Our genetic program absorbed these variables as the best-fitting constants.

$$\pi = f(\pi^e, u) = \pi^e - \beta(u - u_n) + \nu. \tag{3}$$

Although we began with a population of arbitrary candidate functions, the population evolved toward an optimal estimating function.

2.3 Decision-Making Systems

The economy, whether we speak of a small group of producers and consumers, or all entities of an entire nation, is essentially a collection of individuals who make decisions based on various environmental and situational variables, as well as other individuals' decisions. Recall that our objective is to model the underlying mechanics of an economy—more specifically, decision-making systems—as a system of organized complexity. We now propose a further simplified objective; we wish to model the underlying mechanics of a specific decision-making system: the stock market. Many evolutionary techniques and ideas for modeling a stock market are easily extensible to modeling other financial markets, economic markets (of both perfect and imperfect competition), and ultimately, general decision-making systems. Again, the purpose is to simplify our model to one of organized complexity. Later, we will discuss steps for generalizing models.

Let us provide an extremely compact overview of the stock market. The stock market is a market in which shares of a public company's stock are traded. In a real stock market, shares usually pay recurring *dividends* proportional to the stock price. Each trader in the stock market aims to maximize profits, whether made from dividends or from "buying low and selling high". In his or her decision-making process, the trader must take into account both public and private information. *Public information* is information to which all traders have equal (i.e. non-rival and non-exclusive) access; for example, stock prices. *Private information*, however, is available only to those traders who incur the cost (time, effort, and/or money) of obtaining the information; for example, audits of a company, popular opinion regarding a company, or surveys of other traders. When buying or selling shares, traders can make two types of orders: execution orders and limit orders. When placing an *execution order*, a trader buys or sells shares at the most recent available price.[‡] Orders of this type are guaranteed to be successful (that is, if the trader is buying then there is guaranteed to be a seller; and if the trader is selling then there is guaranteed to be a buyer). Meanwhile, when placing a *limit order*, a trader buys or sells shares *only if* the price falls or rises to a specified level. Limit orders are *not* guaranteed to be successful; however, these orders give a certain amount of strategic advantage to the trader. So far, we seem to regard stock prices as an exogenous entity. However, this is not so. *Limit orders drive stock prices.* In general, when more traders place limit orders to buy or sell at prices higher than the current price, the stock price is driven up. When more traders place limit orders to buy or sell at prices lower than the current prices, the stock prices is driven down. Thus, we see not only that stock prices affect traders' decisions, but also that traders' decisions affect stock prices! We omit the detailed machinery of how prices are driven up or down.

3 Model & Motivation

Recal that our most recently refined objective is to model the underlying mechanics of a stock market as an system of organized complexity. Once we do so, we will extend the ideas of a stock market to more complex financial markets, and ultimately, any decision-making system. We begin with a model of the stock market that is simplified enough that, to us, it appears a completely *organized*, almost *simple* system.

[‡]To be precise, *clients* place an order for *brokers* to manage. Throughout this paper, we refer to both clients and brokers interchangeably as "traders" in order to simplify discussion. Also, we assume that the market is *liquid*.

i. Instead of a stock market containing multitudes of companies, we model a stock market containing just a *single* company. Thus, there exist only two assets: one risky (the stock) and one riskless (in our model, non-interest bearing cash). Additionally, shares of our stock do not pay dividends. Thus, the only way for traders to profit is by "buying low and selling high".

ii. Private information will not available to any traders. Therefore, traders can base decisions only on public information, which consists of, in our case, stock prices. (Mathematically speaking, this ensures that a trader's decision will never be a function of any other trader. This simplification eliminates much complexity and chaos in our model.)

iii. Finally, we do not allow traders in our model to place limit orders, since these orders are far more complicated to model than execution orders. However, if we eliminate limit orders, what drives the movement of prices? We address this issue with some clever engineering, as follows. We *assume* that the traders in our model are only a *subset* of a significantly larger population of traders (who *can* place limit orders). Thus, the traders in our model become price-*takers*, but not price-*setters*. Meanwhile, in order to model the movement of prices caused by the traders *not* in our model, we use a specialized random walk.

In this simplified but organized model of the stock market, we wish to analyze the behaviour of traders. How does a trader use given information to make the transactions that maximize his or her profits? Once we understand this aspect of the model in sufficient detail, we can modify the stock market model in order to include more and more intricate details (e.g. private information and limit orders). In each step of the way, we are able to analyze the independent effects of a new variable. This allows us to study the interaction of variables in as much rigor as we deem necessary, and eventually model the financial market as a system of *organized* complexity.

4 Methods

We model the stock market with a simulation of 2,000 time steps. The realistic equivalent of one time step is arbitrary, as long as volatility is adjusted accordingly. Before any simulations begin, we predetermine the prices for each of the 2,000 time steps using an "Exp-3, Beta-64" one-dimensional random walk, and a volatility of \$0.003.[§] The prices to be used throughout simulations in this paper are plotted in figure 7.

We want to run a genetic program of 50 individuals. We note in advance that each individual represents *one* hypothetical trader, while each trader employs only *one* trading strategy. Thus, the terms "individual", "trader", and "trading strategy" are interchangeable (in a sense, one-to-one relations). Furthermore, a trading strategy is simply a function that outputs the profit-maximizing number of shares to hold in a portfolio. Recall that in genetic programs, functions are encoded as parse-trees.

[§] We designed the "random-random walk" especially to simulate stock prices. An exponential random number with decay parameter 3 is used to select the length of a certain trend. The trend itself is determined by a symmetric Beta random number with shape parameter 64.

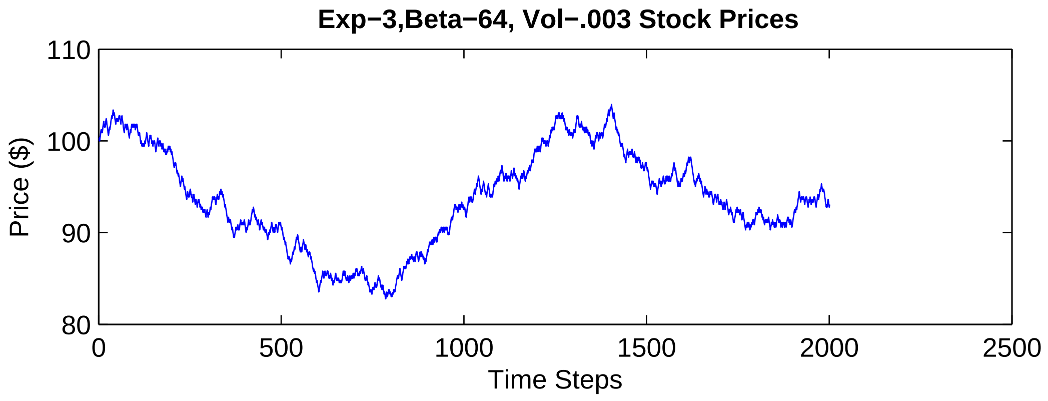

Figure 7: The set of 2000 prices used for trading simulations. Note three significant trends, denoted by A, B, and C.

Thus, one might even append the terms "function" and "parse-tree" to the list of interchangeable terms.

We want to run the genetic program for 50 generations. In each generation, we perform a full trading simulation of 2,000 time steps with the population of 50 individuals. For each time step, each trader calculates a portfolio strategy based on his or her genes. If the portfolio strategy recommends holding more shares than currently owned, the trader buys shares to match the number, and vice versa. After 2,000 time steps, traders with good portfolio strategies will have made higher profit (or lower loss) compared to most others in the population. Conversely, traders with bad portfolio strategies will have made lower profit (or greater loss) compared to most others in the population. In our genetic program, we let profits directly determine fitnesses; and as in all evolutionary algorithms, fitnesses determine the selection of parents for the next generation. Parse-trees are crossed over and mutated in order to produce offspring strategies, which are then tested in the next generation.

We use GPLab, a toolbox written for MATLAB, as a framework for our genetic program.[7] Many modifications were made in order to include the trading simulation phase, trader personalities (e.g. risk aversion and weighting of historical information), and compensate for certain non-deterministic fitness calculations. Potential operators are addition, subtraction, multiplication, division, and differentiation. Potential terminals are historical prices and volatility (only public information). We will thus allow the genetic program to evolve its own trading strategy.

However, we want to know how our evolved strategy performs *relative* real-world strategies. Thus we derive as follows, purely from the theoretical foundations of utility, one possible "rational expectations" strategy.

Letting π denote profit, ψ denote the number of shares held in a portfolio, P_0 denote the current stock price, and P_1 denote the stock price at one time-step in the future (a random variable), we have

$$\pi = \psi(P_1 - P_0). \tag{4}$$

Assuming a quadratic utility function, the utility $U(\pi)$ brought by profit π is

$$U(\pi) = a\pi^2 + b\pi + c. \tag{5}$$

Apply the principle of certainty equivalence, and denoting risk-adjusted profits by $U(\Pi)$, we find

$$\begin{aligned} U(\Pi) &= E[U(\pi)] && (6)\\ &= E[a\pi^2 + b\pi + c]. && (7) \end{aligned}$$

Substituting (4) into (7), we get

$$\begin{aligned} U(\Pi) &= E[a\psi^2(P_1 - P_0)^2 + b\psi(P_1 - P_0) + c] && (8)\\ &= a\psi^2 E[P_1^2] + a\psi^2 P_0^2 - 2a\psi^2 P_0 E[P_1] + b\psi E[P_1] - b\psi E[P_1] - b\psi P_0 + c. && (9) \end{aligned}$$

We know the utility function $U()$, so (9) becomes

$$a\Pi^2 + b\Pi + c = a\psi^2 E[P_1^2] + a\psi^2 P_0^2 - 2a\psi^2 P_0 E[P_1] + b\psi E[P_1] - b\psi E[P_1] - b\psi P_0 + c. \tag{10}$$

Solving for Π with the quadratic equation and simplifying, we get

$$\Pi = -\frac{b}{2a} - \frac{1}{2a}\sqrt{b^2 + 4ab\psi(E[P_1] - P_0) + 4a^2\psi^2(E[P_1^2] - 2P_0E[P_1] + P_0^2)}. \tag{11}$$

To find the number of shares ψ^* that maximizes Π, we need to calculate $\frac{\partial \Pi}{\partial \psi}$:

$$\frac{\partial \Pi}{\partial \psi} = -\frac{b(E[P_1] - P_0) + 2a\psi(E[P_1^2] - 2P_0E[P_1] + P_0^2)}{\sqrt{b^2 + 4ab\psi(E[P_1] - P_0) + 4a^2\psi^2(E[P_1^2] - 2P_0E[P_1] + P_0^2)}}. \tag{12}$$

Finally, we set $\frac{\partial \Pi}{\partial \psi} = 0$ and simplify to obtain:

$$\psi^* = \frac{b}{2a}\frac{E[P_1] - P_0}{Var[P_1] + (E[P_1] - P_0)^2}. \tag{13}$$

The expression $\frac{b}{2a}$ represents the risk aversion parameter. This equation is not quite that of De Fountnouvelle.[8]

We first run a "control" genetic program that forces all individuals to use the rational expectations strategy. Since there is no diversity in the gene pool during the entire run, the only source of evolution comes from traders' personalities. At the moment, these include risk aversion and the "lookback" parameter, which is the number of time steps an individual uses for finite-difference approximations.[¶] Second, we run the full genetic program with selected operators and terminals, and see whether any comparable strategies (relative to rational expecatations) evolve.

[¶]We use finite-differences to approximate price derivatives since our price history is a discrete function.

5 Results

5.1 Rational Expectations

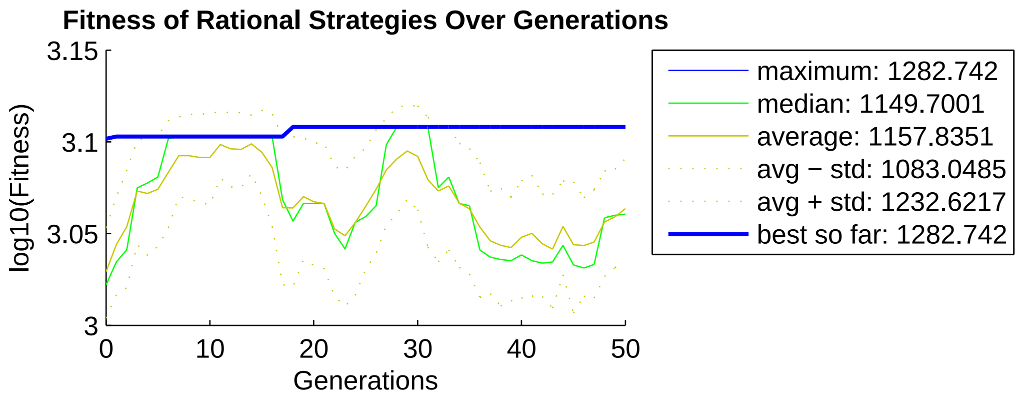

Figure 8: The rational expectations strategy over 50 generations. This strategy is identical for all individuals in the population. The varying property is the individual's preference on the number of time steps in the past to consider into the decision-making process.

In the "control" run, traders' lookbacks evolved toward certain optimal windows of time-steps. After 50 generations, the best-fit trader had $1,282.74 (traders start with $1,000), and had a lookback of 157 time steps, as shown in figure 8. The median profit was $1,149.70, and the average profit was $1,157.835. Since the average was greater than the median, there were more individuals of greater fitness. The "crashes" in median and average wealth can be attributed to our inclusion of *adaptive operator probabilities*. As populations converge toward an optima, mutation rates are significantly increased in order to promote diversity. This is to prevent uniform convergence toward local optima, and instead allow the genetic program to search for a true global optimum.

In figure 9, we show the portfolio strategies of the best-fit trader of the last generation during trends A, B, and C. In trend A, from time step 40 to 603, stock prices fall dramatically. Thus, we should expect more selling than buying. Accordingly, we note many negative portfolio strategies (technically, the lending of shares) in the first diagram. In trend B, from time step 803 to 1259, there was a steep incline in stock prices. Thus, we should expect more buying than selling. As expected, we note many positive portfolio strategies, most of which are far greater than the initial 10 shares that traders began with. Finally, in trend C, from time step 1404 to 1771, there was a moderate drop in stock prices. Thus, we should expect more selling than buying. This is the result we see in the last diagram.

5.2 Evolved Strategies

In the genetic program attempting to evolve strategies from a blank slate, the best-fit trader at generation 50 performed exactly as well as the rational expectations trader!

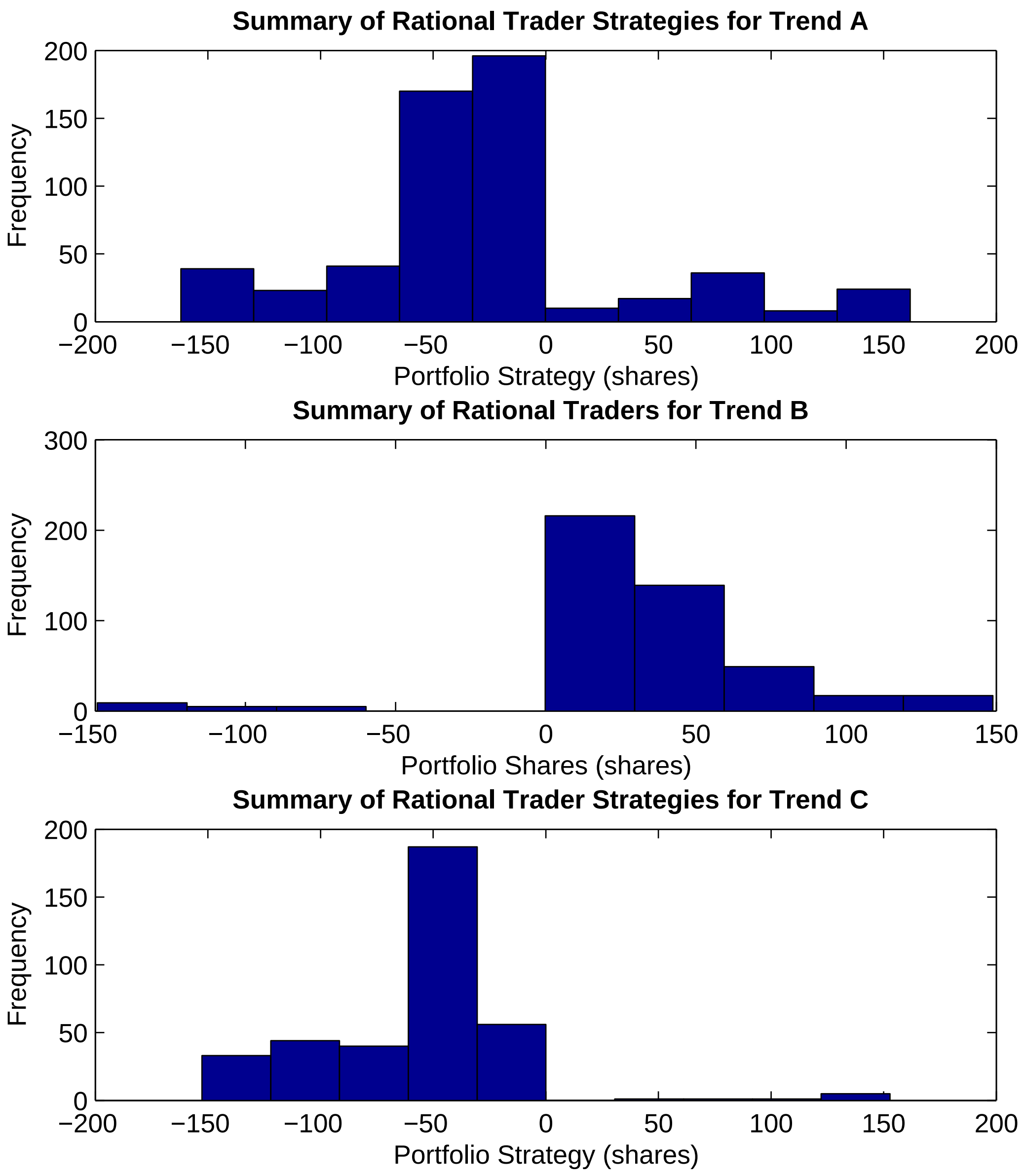

Figure 9: The frequency of portolio strategies during trends A, B, and C for the best-fit trader of generation 50.

This is shown in figure 10. Unlike the control run, the median wealth did not converge toward the best wealth. This is attributable to the significantly greater diversity in this second genetic program. We suspect, however, that we would detect convergence if we run the genetic program with a larger population and for more generations. We are currently limited by computational power.

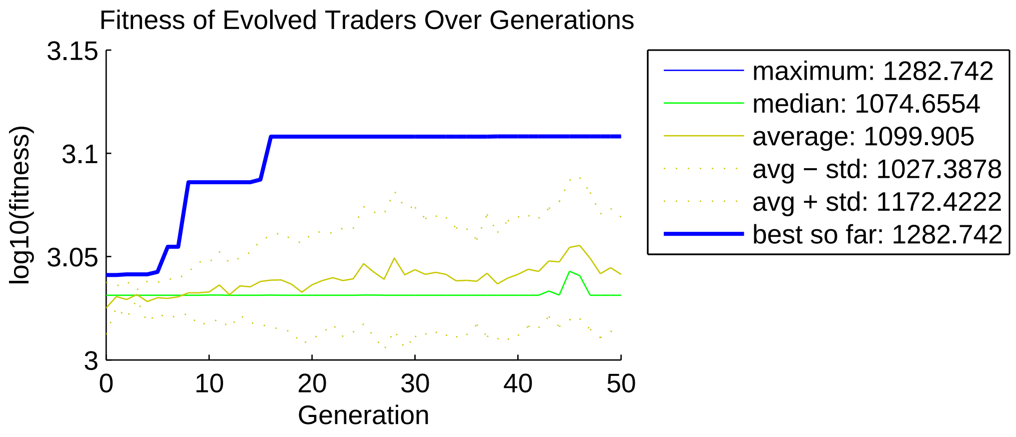

Figure 10: The evolution of the evolved strategy over 50 generations. Here both strategies and individual preferences evolve.

Figure 11 is similar to that of the rational expectations strategy. During trends A, B, and C, the best-fit trader tended to sell, buy, and sell shares, respectively. The portfolio strategies, however, were not as clear cut, as made visible by the "fatter" distribution of frequencies.

6 Discussion

For the reasons enumerated in section 3, our model of trading behavior is not complete. We modeled traders who did not take into account any private information. Realistically, those who are willing to pay the cost of private information should get better estimates of future prices, and thus a strategic advantage in the market. (However, those who spend too much on private information may not make up for information costs.) Private information can be incorporated in one of two ways. On one hand, we could continue to let our traders be only a small subset of a large population of traders; we simply add a certain amount of predictive accuracy into information bearers' strategies. On the other hand, we can extend the model such that traders pay literal costs (representative of hiring analysts, conducting surveys, etc.) to get information regarding the strategies of other traders. Under this model, however, we must abandon certain assumptions about the model, leading to unfavorable consequences such as non-deterministic fitnesses.

Although excluded from this paper, we also implemented a simulation where traders buy and sell shares *from each other*. Thus, competition and scarcity was introduced

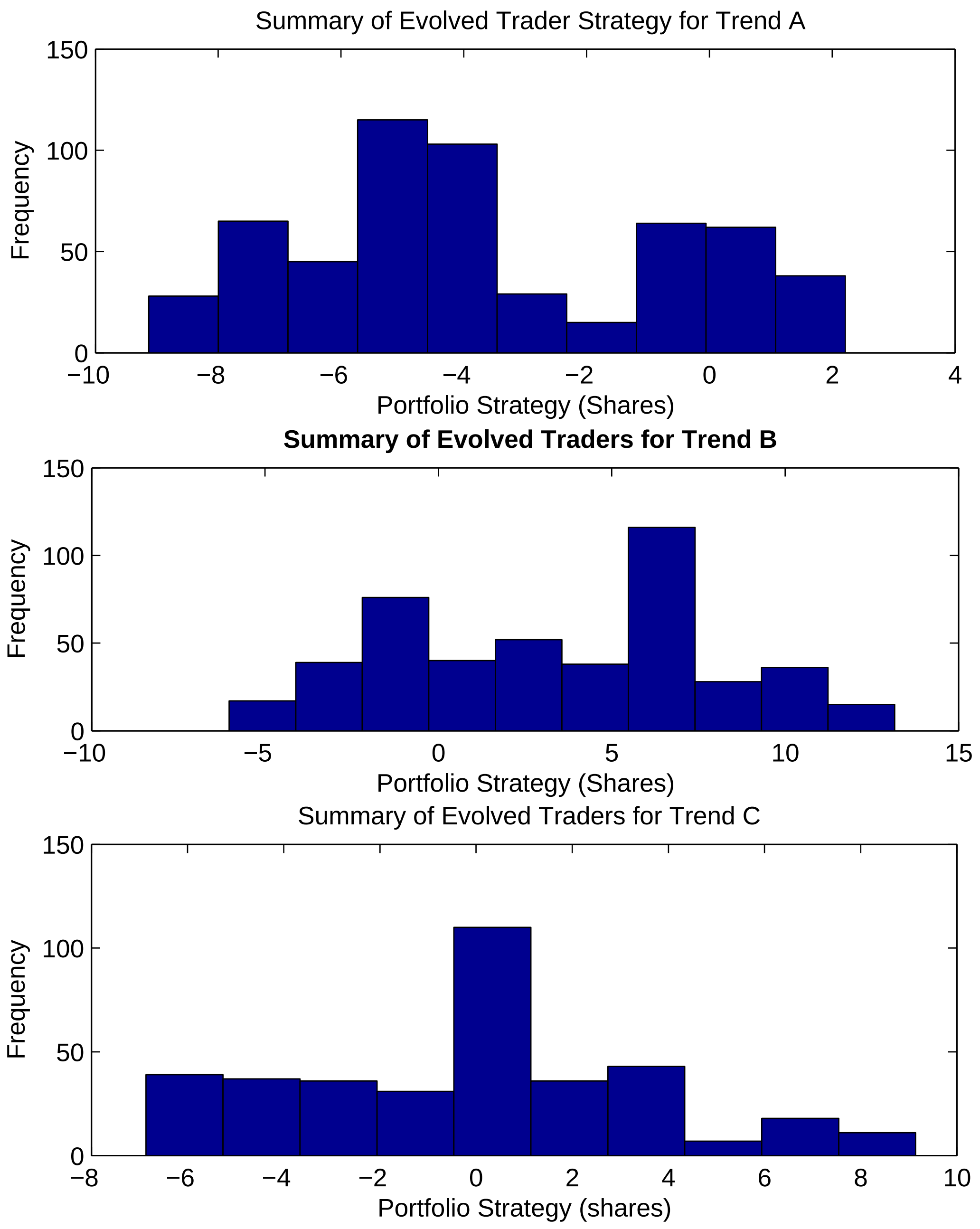

Figure 11: The frequency of portolio strategies during trends A, B, and C for the best-fit trader of generation 50.

into the genetic program. Even with such a small change, the results very different from the ones presented in this paper. Most of the traders' behavior were not explainable, indicating that we should start with simpler changes.

Eventually, we must drop the "black-box" model of the stock market, where prices are explained by a random walk. We would like the traders themselves to not only *take*, but also *set* stock prices. Thus, we began to derive a rational expectations strategy that not only considers the number of shares to hold, but whether a limit order (instead of an execution order) should be placed, and if so, at what price.

Let δp denote how much more or less than the current price a trader is willing to pay. When $\delta p = 0$, the trader is willing to pay exactly the current price; this case represents the placement of an execution order. When $|\delta p| > 0$, the trader is willing to transact only if the price changes by δp; this case represents the placement of a limit order. We can model the *probability of success* of a limit order as a function $\zeta(\delta p)$, with our knowledge that the more the limit price deviates, the less the probability of success. (To further simplify, we can assume the symmetry, $\zeta(|\delta p|)$.) Thus, profit π is

$$\pi = (\text{Profit if order succeeds})\zeta(|\delta p|) + (\text{Profit if order fails})(1 - \zeta(|\delta p|)) \tag{14}$$

$$= [\psi^*(P_1 - P_0 + \delta p) - \psi_0 \delta p]\zeta(|\delta p|) + [\psi_0(P_1 - P_0)](1 - \zeta(|\delta p|)), \tag{15}$$

where ψ^* is the optimal portfolio strategy and ψ_0 is the current number of shares held. We maximize this profit-function with respect to δp and $\psi*$, giving us a simple two-dimensional maximization problem, and eventually the optimal strategy.

Finally, we must add additonal assets such as interest-bearing bonds and dividend-paying stocks into the simulation. Portfolio diversification is an important part of hedging risk on stock markets. In our simulation, we allowed traders to "diversify" only by converting shares into cash. (Since there was no inflation or the opportunity cost of interest, cash was considered "riskless".)

All of these new variables make the model more realistic, but put a strain on computational abilities. We suggest the exploration of Monte Carlo methods to reduce the computational work required per generation.

Clearly, modeling a stock market is still a work in progress. However, note the similarities between stock markets and other financial, even decision-making systems. The market for any product, whether it be cars or health care, can be modeled similarly. "Traders" can be classified as "consumers" or "producers", each group acting to maximize individual benefits. Each agent will have diverse preferences for goods and services. We can manipulate runs to simulate cases in which health care is in high demand, as opposed to low demand, and observe how strategies evolve differently. When the genetic program is complete, we can observe how steady-states are different. To extend this idea to decision-making systems, consider The Prisoner's Dilemma. A strategy only needs to output whether a prisoner should "confess" or "keep silent". Numerous studies have shown that populations evolve toward a "tit-for-tat" strategy, where prisoners "punish" those who have confessed, while "rewarding" those who have kept silent.[9]

7 Conclusion

To begin to understand an economy as a system of organized complexity, we used evolutionary computation to take a bottom-up approach. That is, we simplified the system from an economy to a decision-making process to a stock market. We made assumptions about the market and the traders that enabled us to implement an organized system. We successfully evolved a trading strategy that rivaled the rational expectations strategy that we developed on our own. Then, we proposed extensions to the model that could help us learn more about such complex systems. Although our work could not be completed in its entirety, we set the ground work for incorporating more and more intricacies into the model, hopefully reaching toward realistic yet understandable representations of a truly complex system. We are interested in seeing how further developments could evolve strategies that are far superior to our own models. There is much to be learned of the dynamics of complex systems.

Notes

[1]Robbins, Lionel (1945). An Essay on the Nature and Significance of Economic Science. London: Macmillan and Co., Limited.

[2]Mitchell, M. & Taylor, C. (1999). Evolutionary Computing: An Overview. *Annual Review of Ecological Systems, 30*, 593-616.

[3]Weaver, Warren (1948). Science and Complexity. *American Scientist, 36*, 536.

[4]Hardin, Garrett (1968). The Tragedy of the Commons. *Science, 162*, 1243-1248.

[5]Mankiw, Gregory N. (2007). Macroeconomics. New York, NY: Worth Publishers.

[6]Ecker, J. & Kupferschmid, M. (1988). Introduction to Operations Research. New York: John Wiley and Sons, Inc.

[7]Silva, Sara(2007). GPLAB, version 3, [Computer Program] Coimbra, Portugal: University of Coimbra.

[8]De Fountnouvelle, Patrick (2000). Information Dynamics in Financial Markets. *Macroeconomic Dynamics, 4*, 139-169.

[9]Levy, Steven (1992). Artificial Life. Pantheon.