On the Relevance of Genetic Programming to Evolutionary Economics*

Shu-Heng Chen

AI-ECON Research Center
Department of Economics
National Chengchi University
Taipei, Taiwan 11623
chchen@nccu.edu.tw

Abstract. In this paper, we review the development of artificial adaptive economic agents in evolutionary economics. The review starts from a 1986 paper by Robert Lucas, a Nobel Prize laureate in economics. From there, we shall see how the idea of economic adaptive agents was enriched and implemented by Holland's two books on genetic algorithms (Holland 1975) and on classifier systems (Holland, et al. 1986). We then examine the impact of Holland's artificial adaptive agents on two different groups of economists. One was led by Thomas Sargent, representing New Classical Economics, and the other by Brian Arthur, standing for Santa Fe Institute Economics. A moot point brought here is that the spirit of the genetic algorithm (GA) (John Holland's legacy) is lost in mainstream economics, but is reserved in SFI economics. We then shift to Koza's genetic programming, and show how John Holland's legacy was further expanded in evolutionary economics.

Keywords: Artificial adaptive economic agents, Building blocks, Genetic programming, LISP S-expression, Evolving populations of decision rules

1 Motivation

While genetic programming has been applied to economic modeling for more than half a decade, its relevance to the nature of economics has not been fully acknowledged. In the most sympathetic situations, it is regarded as nothing but *alchemy*. In unsympathetic situation, it is notorious for its *black-box* operation. Sometimes, the process and results are so complicated that economists can hardly consider it relevant and interesting. This chapter is not an attempt to make economists embrace genetic programming, but from a scientific viewpoint, we would like to avoid prejudice due to a lack of solid understanding. This chapter is intended to deliver a simple but strong message: *genetic programming is not just another fancy technique exploited by the unorthodox, but could be a faithful language to express the essence of economics*. In particular, it provides evolutionary economists with a way to *substantiate* some features which distinguish them from the mainstream economists.

To achieve this goal, the paper will address a list of questions as follows.

- Why is genetic programming relevant to economics?
- Are genetic operators relevant to economics?

* This paper is published in K. Aruka (ed.), *Evolutionary Controversy in Economics towards a New Method in Preference of Trans Discipline*, Springer-Verlag, Tokyo.

- Technically speaking, what makes economists like or dislike genetic programming?

2 An Evolving Population of Decision Rules

Let's start from the most fundamental issue: *why is genetic programming relevant?* Lucas (1986) provided a notion of an economic agent. "In general terms, we view or model an individual as *a collection of decision rules* (rules that dictate the action to be taken in given situations) and *a set of preferences* used to evaluate the outcomes arising from particular situation-action combinations." (Lucas, 1986; p.217. Italics Added.) Immediately after the *static description* of the economic agent, Lucas continued to add an *adaptive* (*evolutionary*) version of it. "These decision rules are continuously under review and revision: new decision rules are tried and tested against experience, and rules that produce desirable outcomes supplant those that do not. (Ibid; p.217). So, according to Lucas, the essence of an economic agent is *a collection of decision rules which are adapting (evolving) based on a set of preferences*. In brief, it is an idea of an *evolving population*.

Suppose that an evolving population is the essence of the economic agent, then it seems important to know whether we economists know any operational procedure to substantiate this essence. Back in 1986, the answer was absolutely *no*. That certainly does not mean that we did not know anything about evolving one *decision rule*. On the contrary, since the late 1970s, the literature known as the bounded rationality in macroeconomics has introduced a number of techniques to evolve a single decision rule (a single equation or a single system of equations): recursive regression, Kalman filtering, and Bayesian updating, to name a few. Sargent (1993) made an extensive survey of this subject. However, these techniques shed little light on how to build a Lucasian agent, especially since what we wanted to evolve was not a single decision rule but a population of decision rules.

In fact, it may sound a little surprising that economists in those days rarely considered an individual as a population of decision rules, not to mention attending to the details of its evolution. Therefore, all the basic issues pertaining to models of the evolving population received little, if any, attention. For example, how does the agent *initialize* a population of decision rules? Once the agent has a population of decision rules, which one should they follow? Furthermore, in what ways should this population of decision rules "be continuously under review and revision"? Should we review and revise them one by one because they are independent, or modify them together because they may correlate with each other? Moreover, if there are some "new decision rules to be tried", how do we generate (or find) these new rules? What are the relations between these new rules and the old ones? Finally, it is also not clear how " rules that produce desirable outcomes should supplant those that do not."

There is one way to explain why economists are not interested in, and hence not good at, dealing with a population of decision rules: economists used to derive the decision rule for the agent *deductively*, and the deductive approach usually leads to

only one solution (decision rule), which is the *optimal* one. There was simply no need for a population of decision rules.

3 Genetic Algorithms and Classifier Systems

We do not know exactly when or how the idea of the *evolving population of decision rules* began to attract economists, but John Holland's contribution to *genetic algorithms* definitely exerted a great influence. Genetic algorithms simulate the biological evolution of a society of computer programs, each of which is represented by a chromosome or, normally, a string of binary ones and zeros. Each of these computer programs can be matched to a solution to a problem. This structure provides us with an operational procedure of the Lucasian agent. First, a collection of decision rules are now represented by a society of computer programs (a society of strings of binary ones and zeros). Second, the review and revision process is implemented as a process of natural selection. Table 1 shows how close the relation is between the Lcuasian agent and the genetic algorithm.

Table 1. The Lucasian Agent and the Genetic Algorithm

Lucasian Agents	Genetic Algorithms
Decision rule	A string of binary ones and zeros
A collection of decision rules	A collection of binary strings
Decision rule review	Fitness evaluation
Good rules supplanting bad rules	Selection
Generation of new rules	Crossover and mutation
No equivalent	Schema

While Holland's genetic algorithms have had a great impact on computer science, mathematics, and engineering since the early 1980s, their implications for social sciences were not acknowledged until the late 1980s. In 1987, Robert Axelrod, a political scientist at the University of Michigan, published the first application of the GA to the social sciences. A year later, the first Ph.D dissertation that applied the GA to social sciences was completed by John Miller from, not surprisingly, the University of Michigan. The issue addressed by Axelrod and Miller is the well-known *repeated prisoner's dilemma*. In addition to these two early publications, perhaps the most notable event that brought GAs into economics was the invited speech by Holland at an economic conference at the Santa Fe Institute in the autumn of 1987. Among the audience were some of the most prestigious contemporary economists, including Kenneth Arrow, Thomas Sargent, Hollis Chenery, Jose Scheinkman, and Brian Arthur. In his lecture entitled "The global economy as an adaptive process", Holland introduced to the economics circle the essence of genetic algorithms: *building blocks*.

A building block refers to the specific pattern of a chromosome, i.e., an essential characteristic of a decision rule. There is a formal word for this in the genetic algorithm; it is called a *schema*. In the genetic algorithm, a schema is regarded as the

basic unit of learning, evolution, and adaptation. Each decision rule can be defined as a combination of some schemata. The review and revision process of decision rules is nothing more than a search for the right combination of those, possibly infinite, schemata. To rephrase Lucas's description in Holland's words, *economic agents are constantly revising and rearranging their building blocks as they gain experience.* Not only do genetic algorithms make the Lucasian economic agent implementable, but they also enrich its details.

After a gradual spread and accumulation of knowledge about GA among economists, modeling economic agents with an evolving population of decision rules finally began to increase in the 1990s. To the best of my knowledge, the first referred journal article is Marimon et al. (1990). This paper is a follow-up research of research by Kiyotaki and Wright (1989). In a simple barter economy, Kiyotaki and Wright found that low storage costs *are not* the only reason why individuals use money. The other one is that money makes it easier to find a suitable partner. Replacing the rational agents in the Kiyotaki-Wright environment with *artificially intelligent* agents, Marimon et al., however, found that goods with low storage costs play the dominating role as a medium of exchange.

The population of decision rules used to model each agent is a *classifier system*, another contribution made by Holland in the late 1970s. A classifier system is similar to the Newell-Simon type expert system, which is a population of *if-then* or *condition-action* rules. However, the classical expert system is not adaptive. What Holland did with the classifier system was to apply the idea of competition in the market economy to a society of if-then rules. To implement market-like competition, a formal algorithm known as the *bucket-brigade algorithm,* credits rules generating good outcomes and debits rules generating bad outcomes. This accounting system is further used to resolve conflicts among rules. The shortcoming of the classifier system is that it cannot automatically generate or delete rules. Nonetheless, by adding the genetic algorithm on top of the bucket brigade and the rule-based system, one can come up with something similar to the Lucasian agent, who not only learns from experience, but can be *spontaneous* and *creative.*

While Holland's version of the adaptive agent is much richer and more implementable than the Lucasian economic agent, and the work was already completed before the publication of Holland's second book (Holland et al., 1986), its formal introduction to economists came 5 years after the publication of Lucas (1986). In 1991, Holland and Miller published a sketch of the *artificial adaptive agent* in the highly influential journal *American Economic Review.* The first technique to implement the Lucasian economic agent was finally "registered" in the economic science, and genetic algorithms and classifier systems were formally added to the toolkit of economic analysis. Is 5 years too long? *Maybe not,* given that "Economic analysis has largely avoided questions about the way in which economic agents make choices when confronted by a perpetually novel and evolving world" (Holland and Miller, 1991; p.365).

What is next? If the Lucasian economic agent is a desirable incarnation of the economic agent in economic theory, and if Holland's artificial adaptive agent is in-

deed an effective implementation of it, then the follow-up research can proceed in three directions: first, the novel applications of this new technology, second, the theoretical justifications of the new technology, and finally, the technical improvement of it. That is exactly what we experienced during the 1990s.

For the first line of research, Jasmina Arifovic, a student of Sargent's, finished the first PhD dissertation that applied GAs to macroeconomics in 1991. It was not until 1994, however, that she published her work as a journal article. Arifovic (1994) replaced the rational representative firm in the cobweb model with Holland's adaptive firms, and demonstrated how the adaptation of firms, driven by market forces (natural selection), collectively make the market price converge to the rational-expectations equilibrium price. Since then, a series of her papers has been published in various journals with a range of new application areas, including inflation (Arifovic, 1995), exchange rates (Arifovic, 1996) and coordination games (Arifovic and Eaton, 1995).

4 The SFI Economics

Although Holland introduced this powerful toolkit to economists, he did not conduct any economic research with this toolkit himself except for a joint work with Brian Arthur. Holland and Arthur met in September 1987 at a physics and economics Workshop hosted by the Santa Fe Institute. They had a great conversation on the nature of economics. The *chess* analogy proposed by Arthur led Holland to believe that the real problem with economics is "*how do we make a science out of imperfectly smart agents exploring their way into an essentially infinite space of possibilities?*" (Waldrop 1992, p.151). On the other hand, Arthur was impressed by Holland's approach to complex adaptive systems. Holland's ideas of adaptation, emergence, and perpetual novelty, along with other notions, offered illuminating revelations to Arthur, insights he could never have had gained if he had confined himself to theorizing on equilibria.

This new vision of economics turned out to be the approach of the Santa Fe Institute when it established its economics program in 1988. The essence of the SFI economics was well documented by Arthur (1992). Instead of explaining genetic algorithms and classifier systems, which Holland and Miller (1991) had already done, this paper put a great emphasis on motivation. Arthur eloquently argued why the deductive approach should give way to the inductive approach when we are dealing with a model of heterogeneous agents. His paper thus built the microfoundation of economics upon agents' cognitive processes such as pattern recognition, concept formation, and hypothesis formulation and refutation. Arthur then showed how the dynamics of these cognitive processes can be amenable to analysis with Holland's toolkit.

Maybe the best project to exemplify the SFI approach to economics is the *artificial stock market*. This research project started in 1988. Despite progress made in 1989, journal articles documenting this research were not available until 1994. Palmer et al. (1994) first built their stock market from a standard asset pricing

model (Grossman and Stiglitz 1980). They then replaced the rational representative agent in the model with Holland's artificial adaptive agents, and then simulated the market. The three steps they took, i.e.,

- choosing a benchmark,
- replacing the rational representative agent with artificial adaptive agents, and
- simulating the economy,

are the same as those in procedure followed by Marimon et al. (1990) and Arifovic (1994), but unlike these two papers, the SFI has quite a different motive behind using the idea of artificial adaptive agents.

First of all, for new-classical Economists, using artificial adaptive agents is mainly a way of understanding how the economy will converge to the rational-expectations equilibrium (REE) even when agents are not perfectly rational. Second, in a case where there are multiple equilibria, this device can be further used to select one of them. Third, when the situations get so complicated that the REE is not analytically available, this device can be used to compute the REE. Sargent (1993), in the first economics textbook to introduce genetic algorithms and classifier systems, gave a clear account of all these three advantages.

The SFI economists, in contrast, find these advantages rather minor. As Arthur (1992) stated:

> Yet it is hard to find much justification for why economics should concern itself deeply with learning and adaptive behavior. The new literature convinces us that learning models are useful in spelling out processes of adjustment to the standard equilibria of economic theory; and it raises fascinating questions about what it means to recursively incorporate new information into decision behavior. But it also leaves us with a vague feeling that learning is somehow ancillary to economics–an add-on adjustment dynamics to the core theory, not fully necessary perhaps to theorizing in the field though interesting in its own right. (Ibid., p.1)

There is little doubt that adaptive behavior can lead to the REE, but to the SFI economists, this can happen only if the problem is simple enough. For Arthur, the relevance of genetic algorithms to economics is much more than just strengthening the rational expectations equilibrium. He would like to see how one can use this tool to simulate the evolution of a real economy, such as the emergence of barter trading, money, a central bank, labor unions, and even Communists. However, he understood that one should start with a more modest problem than building a whole artificial economy, and this led to *the artificial stock market*.

Given this different motive, it is also interesting to see how the new classical and the SFI economists programmed agents in their models, and, given their coding or programming, how complex their agents can evolve to be. Arifovic (1994) simply coded a decision made by firms, namely, the quantity supplied, which is just a *number*. She did not code the *decision rules* of firms. Thus, generation after generation, firms were just crunching numbers. Nothing more complex or sophisticated can be

expected from these firms no matter how long last. Marimon et al. (1990) was a little different. They coded the decision rules of traders as exchange and consumption classifiers. The condition-action space is rather simple: there are only 72 possible classifiers in total; one can simply enumerate all of them. With such a limited variation, one can hardly expect any interesting complex behavior to evolve from this style of application.

Palmer et al. (1994) also used the standard trinary string to code different types of trading rules frequently used by financial market traders. Each bit of a string was randomly drawn from the trinary alphabet $\{0, 1, *\}$. Each bit corresponds to the *condition part* of a single trading rule. For example, the condition part of a *double moving average rule* could be "The 20-period moving average of price is above the 100-period moving average." The appropriate bit is 1 if the condition is true, and 0 if it is false. They typically used strings of 70-80 symbols, i.e., the same as the number of trading rules. This defines a search space with 3^{70} to 3^{80} possible non-redundant classifiers. However, each artificial trader has "only" 60 classifiers in her own their classifier system. Consider a case with 100 computerized traders: there are at most 6000 different rules being evaluated in one single trading run. Compared with the size of the search space, the number of rules is infinitesimal. This rather large search space is certainly beyond what Arthur (1992) called the *problem complex boundary*, a boundary beyond which arriving at the deductive solution and calculating it are unlikely or impossible for human agents, and this is where the SFI stock market comes into play. It provides the right place to use genetic algorithms and a great opportunity to watch evolution. As depicted by Arthur (1992):

> We find no evidence that market behavior ever settles down; the population of predictors continually coevolves. One way to test this is to take agents out of the system and inject them in again later on. If market behavior is stationary they should be able to do as well in the future as they are doing today. But we find that when we "freeze" a successful agent's predictors early on and inject the agent into the system much later, the formerly successful agent is now a dinosaur. His predictions are unadapted and perform poorly. The system has changed. From our vantage point looking in, the market–the "only game in town" on our computer–looks much the same. But internally it coevolves and changes and transforms. It never settles. (p.24)

Maybe the real issue is not whether GA are used to strengthen the idea of REE, or to simulate artificial life, but *how we program adaptive agents*. This is crucial because different programming schemes may lead to different results. As Frank Hahn pointed out, while there is only one way to be perfectly rational, there are an infinite number of ways to be partially rational (Waldrop 1992 pp. 250-251). This unlimited "degree of freedom" of programming adaptive agents was also noticed by Sargent (1993): "This area is wilderness because the researcher faces so many choices after he decides to forgo the discipline provided by equilibrium theorizing." (p.2) Arthur would consider letting the agents start off *perfectly stupid*, and get *smarter and smarter* as they learn from experience. Now, comes the core of the issue: *how to*

program agents so that they can be initialized as perfectly stupid individuals, but can potentially get very smart. To answer this question, let us go back to the origin of genetic algorithms.

5 Genetic Programming

It is interesting to note that the binary strings initiated by Holland were originally motivated by an analogy to machine codes. After decoding, they can be computer programs written in a specific language, say, LISP or FORTRAN. Therefore, when a GA is used to evolve a population of binary strings, it behaves as if it is used to evolve a population of computer programs. If a decision rule is explicit enough not to cause any confusion in implementation, then one should be able to write it in a computer program. It is the *population of computer programs* (or their machine codes) which provides the most general representation of the *population of decision rules*. However, the equivalence between computer programs and machine codes *breaks down* when what is coded is the parameters of decision rules rather than decision rules (programs) themselves, as we often see in economic applications with GAs. The original meaning of evolving binary strings as evolving computer programs is lost.

The gradual loss of the original function of GAs has finally been noticed by John Koza. In 1987, when John Koza was on a London stopover after attending an AI conference in Italy, a friend handed him the proceedings of a conference on genetic algorithms. Impressed with the variety of applications, he nonetheless noticed a glaring omission: there was a lack of ways to *generate programs*. He then began to work on a system to breed *computer programs* genetically. He chose the language LISP as the medium for the programs created by genetic programming (GP) because the syntax of LISP allows computer programs to be manipulated easily like the bitstrings in GAs, so that the same genetic operations used on bitstrings in GAs can also be applied to GP. His breakthrough, however, " was deciding to identify the *units of crossover* not as single characters, or even as lines in a computer program, but as symbolic expressions (S-expressions) written in the LISP syntax." (Levy 1992 pp 176-177; italics added.)

5.1 LISP S-expression

LISP S–expressions consist of either *atoms* or *lists*. Atoms are either members of a *terminal set*, that comprise the data (e.g., constants and variables) to be used in the computer programs, or they are members of a *function set* that consists of a number of prespecified functions or operators that are capable of processing any data value from the terminal set *and* any data value that results from the application of any function or operator in the function set. Lists are collections of atoms or lists, grouped within parentheses. In the LISP language, everything is expressed in terms of operators operating on some operands. The operator appears as the left-most element in the parentheses and is followed by its operands and a closing

(right) parenthesis. For example, the S-expression (+ X 3) consists of three atoms: from the left-most to right-most they are the function "+", the variable X and the constant 3. As another example, (× X (− Y 3)) consists of two atoms and a list. The two atoms are the function "×" and the variable "X," which is then followed by the list (− Y 3).

LISP was invented in the late 1950s by John McCarthy at MIT as a formalism for reasoning about the use of certain kinds of logical expressions, called recursion equations. LISP possesses unique features that make it an excellent medium for complex compositions of functions of various types, handling hierarchies, recursion, logical functions, self-modifying computer programs, self-executing computer programs, iterations, and structures whose size and shapes are dynamically determined. The most significant of these features is the fact that LISP descriptions of processes (routines) can themselves be represented and manipulated as LISP data (subroutines). As Koza (1992a) demonstrated, LISP's flexibility in handling procedures as data makes it one of the most convenient language in existence for exploring the idea of evolving computer programs genetically, however, Koza and others have noted that the use of LISP is not necessary for genetic programming; what is important for genetic programming is the implementation of a LISP–like environment, where individual expressions can be manipulated like data, and are immediately executable.

5.2 Symbolic Regression

The distinguishing feature of GP is manifested by its first type of application in economics, known as *symbolic regression*. In symbolic regression, GP is used to discover the underlying data-generation process of a series of observations. While this type of application is well known to econometricians, the perspective from GP is novel. As Koza (1992b) stated,

> An important problem in economics is finding the mathematical relationship between the empirically observed variables measuring a system. In many conventional modeling techniques, one necessarily begins by selecting the size and shape of the model. After making this choice, one usually then tries to find the values of certain coefficients required by the particular model so as to achieve 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 model itself.* (p.57; italics added.)

Econometricians offer no general solution to the determination of size and shape (the functional form), but for Koza, finding the functional form of the model can be viewed as *searching a space of possible computer programs* for the particular computer program which produces the desired output for given inputs.

Koza employed GP to rediscover some basic physical laws from experimental data, for example, Kepler's third law and Ohm's law (Koza 1992a). He then also applied it to eliciting a very fundamental economic law, namely, the *quantity theory of money* or the *exchange equation* (Koza 1992b). Genetic programming was

thus formally demonstrated as a *knowledge discovery* tool. This was probably the closest step ever made toward the original motivation of John Holland's invention: "Instead of trying to write your programs to perform a task you don't quite know how to do, *evolve them.*" Indeed, Koza did not evolve the parameters of an arbitrary chosen equation; instead, he evolved the whole equation from scratch. This style of application provides an evolutionary determination of bounded rationality.

Koza (1992b) motivated a series of economic applications of genetic programming in the mid-1990s. Chen and Yeh (1996a) applied genetic programming to rediscovering *the efficient market hypothesis* in a financial time series. Chen and Yeh (1997a) then moved one step forward to propose an alternative formulation of the efficient market hypothesis in the spirit of the *Kolmogorov complexity* of algorithms for pattern extraction from asset price data. Chen and Yeh (1997a) and Szpiro (1997a) employed GP to discover the underlying chaotic laws of motion of time series data. Neely et al. (1997) and Allen and Karjalainen (1999) also adopted a GP approach to discover profitable technical trading rules for the foreign exchange market and the stock market, respectively. Another area in which GP was actively applied is *option pricing*. Chen et al. (1999) used GP for hedging derivative securities. Keber (1999) showed that genetically determined formulas outperformed most frequently quoted analytical approximations in calculating the implied volatility based on the Black-Scholes model. Chidambaran et al. (2000a) and Keber (2000) derived approximations for calculating option prices and showed that GP-models outperformed various other models presented in the literature.

Needless to say, in the future, one can expect many more applications of GP to the automatic discovery of economic and financial knowledge (automatic generation of economic and financial knowledge in terms of their computer-programmed representations). However, its significant contribution to economics should not be mistaken for a perfect solution to knowledge discovery, data mining, or, more generally, *function optimization*. In a nutshell, genetic programming should be used to grow *evolving hierarchies* of building blocks (subroutines), the basic units of learning and information, from an immense space of subroutines. All evolution can do is look for improvements, not perfection. John Holland believed that these evolving hierarchies are generic in adaptation, and can play a key role in understanding human learning and adaptive processes. Hence, the value of GP to economists can best be seen as a model of human learning. From this standpoint, one may ask how GP, as a model of human learning, can enrich our understanding of that particular subject. This is how we approach the issue. Considering GP as a connection between adaptive agents and their surroundings, from the agent's viewpoint, we shall first show that GP can give us a "subjective" measure of the complexity of the agent's surroundings, and being a model builder, we will then argue that the "subjective" measure can be "objectively" determined by the culture of the society where the agent is situated.

6 Genetic Programming and Human Learning

6.1 Program Complexity of an Agent's Surroundings: A Subjective Measure

We shall exemplify the first assertion based on Chen and Yeh (1997a). They employed GP to discover the underlying law of motion for some simple chaotic time series. They considered the following three chaotic laws of motion.

$$x_{t+1} = 4x_t(1 - x_t), \quad x_t \in [0, 1] \ \ \forall t \tag{1}$$

$$x_{t+1} = 4x_t^3 - 3x_t, \quad x_t \in [-1, 1] \ \ \forall t \tag{2}$$

$$x_{t+1} = 8x_t^4 - 8x_t^2 + 1, \quad x_t \in [-1, 1] \ \ \forall t \tag{3}$$

These three laws of motion are different in their *algorithmic size*, i.e., the *length* of their symbolic expression. To see this, we rewrite each of the equations above into the corresponding LISP S-expression.

$$(\ * \ (\ 4 \ * \ (\ x_t \ (\ - \ 1 \ x_t \ ) \ ) \ ) \ ) \tag{4}$$

$$(\ - \ (\ * \ 4 \ (\ * \ x_t \ (\ * \ x_t \ \ x_t \ ) \ ) \ ) \ (\ * \ 3 \ \ x_t \ ) \ ) \tag{5}$$

$$\begin{array}{l} (\ + \ (\ - \ (\ * \ 8 \ (\ * \ x_t \ (\ * \ x_t \quad (\ * \ x_t \ \ x_t \ ) \ ) \ ) \ ) \\ (\ * \ 8 \ (\ * \ x_t \ \ x_t \ ) \ ) \ ) \ 1 \ ) \end{array} \tag{6}$$

The *length* of a LISP S–expression is determined by counting the number of elements (atoms) in the string that makes up the S–expression from the left-most to the right-most position. From Eqs. 4 to 6, the lengths of the LISP S-expressions are 7, 11, and 16 respectively. Therefore, in terms of algorithmic complexity, Eq. 1 is the simplest, while Eq. 3 is the most complex. Chen and Yeh then examined how this difference might affect the performance of GP.

By setting the initial value x_0=0.213, a time series composed of 50 observations is generated for Eqs. (1)-(3) respectively. Call them time series 1, 2, and 3. These time series serve as the training data for GP. Four experiments were implemented for each series. For each experiment, they let GP run for 1000 generations. For series 1 and 2, GP was able to discover the underlying law of motion in all four experiments. However, the number of generations required for this discovery was different. For series 1, it took 7, 12, 14, and 19 generations, whereas for Series 2, it took 29, 37, 37, and 70 generations. As for series 3, GP failed to discover the law of motion in three out of the four simulations, and for the only success the law of motion was discovered at the 151th generation. These experiments show the effect of the length of the LISP S-expression (the algorithmic complexity of the program) on discovery. A simple hypothesis was then proposed in their paper: *The chance of discovering*

simple (short) programs in a small number of generations is relatively higher than that of discovering long (complex) programs.

While this hypothesis sounds intuitive, a formal test of it is not that straightforward. One of the technical reasons is that the complexity of a program is not independent of *the terminal set* or *the function set* initially given by the user. To see this, let us go back to the chaos experiment discussed above. The function set originally employed by Chen and Yeh (1997a) is $\{+, -, \times, \%\}$, and the terminal set is $\{x_t, \mathcal{R}\}$, where $\mathcal{R}$ is the ephemeral random constant.[1] If we add the function "cubic" to the function set, then the minimal description of Eq. 2 is

$$(\ -\ (\ *\ 4\ (\ cubic\ \ x_t\ )\ )\ (\ *\ 3\ \ x_t\ )\ ) \tag{7}$$

and the program length is only eight. Alternatively, if we add x_t^3 to the terminal set, then the minimal description becomes

$$(\ -\ (\ *\ 4\ \ x_t^3\ )\ (\ *\ 3\ \ x_t\ )\ ) \tag{8}$$

In this case, the program length of Eq. 2 is even shorter and is seven, which is the same as that of Eq. 1. It is therefore likely to discover the hidden law of series 2 as fast as to discover that of series 1. So depending on the user-supplied function set and terminal set, a mathematical function can have different program lengths. Therefore, a refinement of the hypothesis proposed above is given as follows. Let us define the program complexity of a mathematical function *with respect to a given function set and terminal set* as the length of its *minimal description. Then the chance of discovering the mathematical function with the given function set and terminal set in a finite number of generations is an inverse function of its program complexity.*

6.2 Complex or Simple: An Objective Measure Based on Culture

While the omission of relevant functions and terminals can make the description of a hidden law (a mathematical function) extraordinarily complex and hence difficult to learn, this does not suggest that one should start with a sizeable function set and terminal set. Indeed, the inclusion of irrelevant functions or terminals can enlarge the search space, which may make a target more difficult to reach. Therefore, while large function and terminal sets are more likely to reduce the program complexity of a mathematical function and make it easier to find, the larger search space induced may have the net effect of making the chance of discovering it uncertain. Perhaps there is no need for users to worry about whether their choice of function sets and terminal sets is "*optimal*". As we have already mentioned, GP is better regarded as a simulator which simulates human thinking and discovery processes. If humans can make mistakes and learn very slowly in some situations, there is no point in asking GP to be a panacea. As a simulator, what GP can do is to spell out the factors affecting discovery, such as the *program complexity* discussed above. In addition,

[1] See Koza (1992a) for details of the ephemeral random constant.

Szpiro (2000) gave a thorough discussion of a list of factors some of which were found to be of importance in his two earlier studies (Szpiro 1997a,b). These factors are the size of the function set, the terminal set, the initial population, the mutation rate, and the hold of the best-so-far program.

Let us review the first two factors on this list, namely, the function set and the terminal set. A function in the function set can be regarded as a *subroutine*, and a terminal in the terminal set can be treated as a *concept*. Those subroutines and concepts together represent the *culture* of a society. An advanced society may have accumulated a large number of well-structured subroutines with highly constructive concepts. To model an artificial adaptive agent in this society, one need not start with a primitive function and terminal set. For example, to model an artificial adaptive agent in modern financial markets, one can assume the availability of advanced pricing formulas, and start from there. In their study of option pricing formulas, Chidambaran et al. (2000b) included the Black-Scholes model, as an existing subroutine, into their function set. The formulas generated by GP for the equity options were then adaptations of the Black-Scholes model.

Since sizeable function set and terminal sets induces a huge search space for the agent, it is desirable to have a large number of programs in the initial population. The combined effect of a large population size and large function and terminal sets is just a corollary of an advanced civilization: to tackle a problem, citizens of an advanced civilization tend to come up with a lot more ideas (alternatives) than citizens in a less developed society. Nonetheless, after a few generations the population of programs generally becomes very homogeneous, and hence the effective population size is largely reduced. In terms of human behavior, this is not unexpected. If an agent is put down in a stable situation, they will evolve to rely on a few fixed rules, usually called *routines*, without much diversity.

However, a problem technically known as *premature convergence* or *trapping into a local optimum* can be another consequence of the reduction in diversity. While this is also not inconsistent with human behavior, considering that humans are frequently deceived by their eyes and intuitions, it is important to introduce novelties into human cognition. In GP, novelties are introduced into an agent's cognition via the crossover operator and the mutation operator. In most applications in economics and finance, the crossover rate and mutation rate are set as *constants*. While genetic programmers from an engineering background has evidenced the significance of not setting these as constants and Szpiro (2000) also made a similar suggestion, how those flexible designs can meaningfully be interpreted as a real human reasoning process is an issue yet to be addressed.

6.3 Features of Ideas-Generation Processes

While by varying the size of the initial population, the terminal set, and the function set one is able to capture artificial agents initially endowed with different civilizations, the capacity of GP to simulate human behavior has to be judged by the *process of ideas* being generated. Note that we are interested not only in the *final outcome*

(the discovery), but also in the *process* converging to the outcome. The quotation below may help explain what this criterion means.

> Not only did the genetic programming system manage to "rediscover" Kepler's famous third law,..., but also, as the system climbed up the fitness scale, one of its interim solutions corresponds to an earlier *conjecture* by Kepler, published ten years before the great mathematician finally perfected the equation! (Levy 1992 p.179; italics added.)

Here, not only did GP discover Kepler's third law in its mathematically perfected version, but along the way it also discovered the *conjecture*, i.e., the version before the perfection. Of course, there is no point in expecting GP always to be so precise in replicating a discovery process, since the discovery process itself may be highly random, and hence not unique. Instead, as a first approximation, it would be crucial to see whether the ideas-generation process driven by GP is similar in spirit to the human reasoning process. For this purpose, it would be useful to summarize certain general features of human reasoning.

Generally speaking, the human reasoning process can be characterized as a process *from the simple to the complex*, *from the major to the minor* (*from the heavy to the light*), and *from the linear to the nonlinear*. If the words "simple" and "complex" can be formally defined in terms of *program complexity*, then the ideas-generation process driven by GP is indeed a process from the simple to the complex. It is mainly due to the trees-generation mechanism, which implicitly follows a Possion-like process; therefore, the search process is initially biased toward simple programs. Unless simple programs fail to work, complex programs would find it difficult to become popular in the initial stage. The feature of "*simple-to-complex*" can be found quite extensively in the videotape accompanying the book by Koza (1992a) (Koza and Rice 1992).

Secondly, if the meanings of "major" and "minor" are consistent with the fitness, i.e., the major part corresponds to a much larger fitness than the minor part, then the ideas-generation process driven by GP should present the second feature, *from the major to the minor*. In a practical application, it may get stuck in the major part and be unable to move further, but it is unlikely to be locked in the minor part.[2] While the second feature has not been well acknowledged or studied by genetic programmers, it has motivated a few performance boosters for GP, such as *the two-stage symbolic regression*, *cross breeding*, etc. (Szpiro 2000).

Maybe the most difficult part for GP to satisfy is the third property, "from linear to nonlinear." As mentioned above, GP tends to start with simple functions, and then gradually moves toward complex functions if necessary. Nonetheless, GP is unable to distinguish a simple linear function from a simple nonlinear function. For example, the function e^x is as simple as $\beta_1 x$, even though the latter is linear and the former is not. As a result, the chances of having e^x or $\beta_1 x$ are basically the same in

[2] For example, using the Rossler attractor with a regular fitness function, Szpiro (2000) showed that GP could "discard" 90% of observations simply because they are *minor*. By doing that, GP could keep track of the remaining 10% even though they were sometimes described as outliers.

the initial generation of ideas. However, if this is the case, then given the density of linear functions, the probability of generating a linear function in the initial stage is rather small. In other words, just as it is biased toward simple functions, GP is also biased toward *nonlinear* functions.

The bias toward nonlinearity has created a lot of difficulties in the modeling of human learning and adaptation. For example, regular GP often leads to overly complex nonlinear functions. Such nonlinear functions, comprised of complex combinations of functions and indicators, are often difficult to understand and interpret. In many applications, deciphering the winning programs is an almost impossible task. As a result, it might be futile to use these programs (rules) to understand the processes by which humans behave. The usefulness of the GP method for modeling human learning and adaptation thus seems quite restricted a priori.

One way to fix this problem is to make the terminal set and the function set adaptive also. One may start with a primitive function and terminal set at the beginning of learning, and may enlarge it later when obliged to do so. In this way, we are able to choose a convincing sequence of function sets and terminal sets so that the sequence of rules generated can mimic human learning more closely. For example, if one considers that humans generally start their search from a space of linear functions, not just simple functions, then one can first include only "+" and "×" into their function set. By further restricting the generation rules appropriately, only functions in the form of Eq. (9) will appear, and other complex and nonlinear functions will not show up at this stage.

$$a_0 + \sum_{i}^{n} a_i x_i \tag{9}$$

7 Concluding Remarks

This paper considers an important contribution initiated by John Holland and further extended by John Koza. Based on a biological foundation, an artificial adaptive agent is introduced to economic modeling. This agent is characterized by its capacity for autonomous discovery (autonomous programming ability). This idea has been extensively applied to economics and finance, and we have reviewed a few of these cases. Owing to limitations of space, what has not been covered in this paper is the study of the behavior of a collection of GP-based agents in a market-like environment. Studies in this area can be found in Andrews and Prager (1994), Chen and Yeh (1996b, 1997b, 1999, 2000a,b), and Lensberg (1999). This subject will be covered in the near future.

References

1. Allen F. Karjalainen R (1999) Using genetic algorithms to find technical trading rules. Journal of Financial Economics 51(2): 245–71.
2. Andrews M, Prager R (1994) Genetic Programming for the Acquisition of Double Auction Market Strategies. In Kinnear K, (Ed.) Advances in Genetic Programming, Vol.1, MIT Press. pp 355-368.

3. Arifovic J (1991) Learning by genetic algorithms in economic environments. Ph.D. thesis, University of Chicago.
4. Arifovic J (1994) Genetic algorithms learning and the cobweb model. Journal of Economic Dynamics and Control 18(1): 3–28.
5. Arifovic J (1995) Genetic algorithms and inflationary economies. Journal of Monetary Economics 36(1): 219–43.
6. Arifovic J, Eaton B (1995) Coordination via genetic learning. Computational Economics 8(3): 181–203.
7. Arifovic J (1996) The behavior of the exchange rate in the genetic algorithm and experimental economies. Journal of Political Economy 104(3):510–541
8. Arthur B (1992) On learning and adaptation in the economy. Sante FI Economics Research Program Working Paper 92-07-038.
9. Chen S.-H, Yeh C.-H (1996a) Genetic programming and the efficient market hypothesis. In Koza J, Goldberg D, Fogel D, Riolo R (Eds), Genetic programming 1996: proceedings of the first annual conference. MIT Press, Cambridge, pp 45-53
10. Chen S.-H, Yeh C.-H (1996b) Genetic programming learning and the cobweb model. In Angeline P. (Ed) Advances in Genetic Programming, Vol.2. MIT Press, Cambridge, MA, pp 443-466.
11. Chen S.-H, Yeh C.-H (1997a) Toward a computable approach to the efficient market hypothesis: An application of genetic programming. Journal of Economic Dynamics and Control 21: 1043-1063.
12. Chen S.-H, Yeh C.-H (1997b) Modeling speculators with genetic programming. In Angeline P. Reynolds R. McDonnell J. Eberhart R. (Eds.) Evolutionary programming VI, Springer-Verlag, Berlin, pp. 137-147.
13. Chen S.-H, Yeh C.-H (1999) Modeling the expectations of inflation in the OLG model with genetic programming. Soft Computing 3(2):53-62.
14. Chen S.-H, Yeh C.-H (2000a) Simulating economic transition processes by genetic programming. Annals of Operation Research 97: 265-286
15. Chen S.-H, Yeh C.-H (2000b) Evolving traders and the business school with genetic programming: A new architecture of the agent-based artificial stock market. Journal of Economic Dynamics and Control 25: 363-393.
16. Chen, S.-H, Lee W.-C. Lee, Yeh C.-H. (1999) Hedging derivative securities with genetic programming. International Journal of Intelligent Systems in Accounting, Finance and Management 8(4): 237-251.
17. Chidambaran N, Lee C, Trigueros J. (2000a) Option Pricing via Genetic Programming. In: Abu–Mostafa Y, LeBaron B, Lo A, Weigend A (Eds.) Computational Finance – Proceedings of the Sixth International Conference. The MIT Press.
18. Chidambaran N, Triqueros J, Lee C.-W (2000b) Option pricing via genetic programming. In: Chen S.-H. (Ed) Evolutionary Computation in Economics and Finance (in press). Physica Verlag.
19. Grossman, S J, Siglitz J. (1980) On the impossibility of inofrmationally efficient markets. American Economic Review 70:393-408
20. Holland J. (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.
21. Holland J, Holyoak K, Nisbett R (1986) Induction: processes of inference, learning and discovery (computational models of cognition and perception), MIT Press, Cambridge.
22. Holland J. Miller J (1991) Artificial adaptive agents in economic theory. American Economic Review. 81(2):365-370
23. Keber C (1999) Genetically Derived Approximations for Determining the Implied Volatility. OR Spektrum **21**, 205–238
24. Keber C (2000) Option Valuation with the Genetic Programming Approach. In: Abu–Mostafa Y, LeBaron B, Lo A, Weigend A. (Eds.) Computational Finance – Proceedings of the Sixth International Conference. The MIT Press.
25. Kiyotaki N, Wright R (1989) On money as a medium of exchange. Journal of Political Economy 97:927-954.
26. Koza J (1992a) Genetic programming: on the programming of computers by means of natural selection. The MIT Press.
27. Koza J (1992b) A Genetic approach to econometric modelling. In Bourgine P. Walliser B (Eds.) Economics and cognitive Science. Pergamon Press, pp 57-75.
28. Koza J, Rice J (1992) Genetic programming: The movie. (Videotape Accompanying the Book Genetic Programming: On the Programming of Computers by Means of Natural Selection by John Koza). The MIT Press.
29. Levy S (1992) Artificial life: a report from the frontier where computers meet biology. Vintage, New York.

30. Lensberg T. (1999) Investment behavior under Knightian uncertainty- an evolutionary approach. Journal of Economic Dynamics and Control 23:1587-1604.
31. Lucas R (1986) Adaptive Behaviour and Economic Theory. In: Hogarth R. Reder M. (Eds) Rational choice: the contrast between economics and psychology. University of Chicago Press, pp 217-242.
32. Marimon R, McGrattan E. Sargent T (1990) Money as Medium of Exchange in an Economy with Artificially Intelligent Agents. Journal of Economic Dynamics and Control 14, pp 329-373.
33. Neely C. Weller P, Ditmar R (1997) Is technical analysis in the foreign exchange market profitable? A genetic programming approach. Journal of Financial and Quantitative Analysis 32(4): 405–427.
34. Palmer R, Arthur W, Holland J, LeBaron B, and Tayler P (1994) Artificial economic life: A simple model of a stockmarket. Physica D 75: 264-274.
35. Sargent, T (1993) Bounded rationality in macroeconomics. Oxford.
36. Szpiro G (1997a) Forecasting chaotic time series with genetic algorithms. Physical Review E 55: 2557-2568.
37. Szpiro G (1997b) A search for hidden relationships: Data mining with genetic algorithms. Computational Economics 10: 267-277.
38. Szpiro G (2000) Tinkering with genetic algorithms: Forecasting and data mining in finance and economics. In: Chen S.-H. (Ed) Evolutionary Computation in Economics and Finance (in press). Physica Verlag.
39. Waldrop M (1992) Complexity: The emerging science at the edge of order and chaos. Simon and Schuster.