Fundamental Issues in the Use of Genetic Programming in Agent-Based Computational Economics

Shu-Heng Chen
AI-ECON Research Center
Department of Economics
National Chengchi University
Taipei, Taiwan 11623
E-mail: chchen@nccu.edu.tw

Abstract. This paper provides a review of some fundamental issues of the applications of genetic programming to agent-based computational economics. The issues under review covers four aspects of genetic programming, namely, *primitives*, *semantics*, *genetic operators*, and *architecture*. The paper surveys the technical issues encountered in each of these four aspects, and some proposed solutions to them.

1 Motivation

Genetic programming has been applied to agent-based computational economics for more than half a decade. This line of research is currently challenged by several non-trivial technical issues. This paper will give a full account of them, including their significance and solvability.

Genetic programming is designed to grow (evolve) a population of evolving hierarchies of building blocks (subroutines), the basic units of learning and information, from an immense space of them. There are three key words in this brief definition, namely, *building blocks*, *hierarchies*, and *evolving population of hierarchies*. A building block is a class of decision rules defined by some specific characteristics which can perform certain kinds of functions. In genetic programming, building blocks are initially randomly generated by a set of *primitives*, known as the function set and terminal set. Here comes the first issue: *the choice of primitives*.

Once a set of primitives is given, *hierarchies* are derived by some *production rules* (*grammar*). Given the grammar, any hierarchy which is *syntactically valid* is a legitimate species. Its appearance and popularity will be crucially dependent on its *fitness*, which is basically driven by three genetic operators, namely, *reproduction*, *crossover*, and *mutation*. Issues encountered at this stage are two-fold: *the semantic restrictions of derived hierarchies* and *the use of genetic operators*.

Finally, those hierarchies are not static, but dynamically adapted to the environment which is either exogenously given or endogenously change with the agents. The dynamics generated by GP is a sequence of sets of programs (parse trees, subroutines, ideas, strategies). This sequence can be interpreted as the evolution of an artificial society as a whole. In other words,

a society of adaptive agents has a one-to-one and onto relation to a population of programs. Alternatively, this sequence can also be interpreted as the adaptation of a single agent. In this case, a society of agents consists of many populations of programs. The first interpretation is often referred to as *single-population GP* (**SGP**), whereas the second is dubbed *multi-population GP* (**MGP**). Which interpretation is appropriate? We will address this issue at the end of the paper.

2 Selection of the Function Set and Terminal Set

This issue is crucial because the *algorithmic complexity* (*minimal description length*) of any decision rule depends on the choice of the function set and terminal set. Algorithmic complexity is relevant because the chance of discovering a specific decision rule in a finite number of generations is a decreasing function of its algorithmic complexity.

Example 1: [12]
We shall exemplify the assertion above based on [12]. [12] 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 *algorithmic size*, i.e., the *length* of the LISP symbolic expressions. To see this, we rewrite each of the equations above into the corresponding LISP S-expressions.

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

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

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

The *length* of a LISP S–expression is determined by counting from the leftmost to the rightmost position the number of elements (atoms) in the string that makes up the S–expression. From Equations (4) to (6), the length of the LISP S-expression is 7, 11, and 16 respectively. Therefore, in terms of algorithmic complexity, Equation (1) is the simplest, while Equation (3) is the most complex. [12] 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 fifty observations was generated for Equations (1)-(3) respectively. Call them Time Series 1, 2, and 3. These time

series served as the training data for GP. Four experiments were implemented for each series. For each experiment, they let GP run for 1,000 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 respectively, 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 in the only successful case the law of motion was discovered at the 151th generation. These experiments demonstrated the effect of the length of the LISP S-expression (algorithmic complexity of the program) on discovery.

The function set originally employed by [12] 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 Equation 2 is simply

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

and the program length of it is only 8. 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 Equation 2 is even shorter–7, to be exact, which is the same as that of Equation 1. It is, therefore, likely to discover the hidden law of Series 2 as fast as to discover that of Series 1. Thus, a mathematical function can have different program lengths, depending on the user-supplied function set and terminal set.[2]

Formally speaking, let $\mathcal{T}$ be the terminal set and $\mathcal{F}$ be the function set, and denote their cardinality by $\mid \mathcal{T} \mid$ and $\mid \mathcal{F} \mid$.

$$Prob(f^* \in G_n) = P[K(f^* \mid \mathcal{F} \bigcup \mathcal{T})], \tag{9}$$

and

$$\frac{\Delta Prob(f^* \in G_n)}{\Delta K} \mid_{\mathcal{F} \bigcup \mathcal{T}} \leq 0, \tag{10}$$

where f^* is a targeted decision rule, say, the optimal decision rule, G_n is the nth generation of population, and $K(f^* \mid \mathcal{F} \bigcup \mathcal{T})$ refers to the algorithmic complexity of the decision rule of f^* given the functional set $\mathcal{F}$ and the terminal set $\mathcal{T}$.

Since a larger function set and terminal set will help abbreviate the representation of a decision rule, the algorithmic complexity of a function can only be non-positively related to $\mid \mathcal{F} \mid$ and $\mid \mathcal{T} \mid$, i.e.,

$$\frac{\Delta K}{\Delta \mid \mathcal{F} \mid} \leq 0, \ \ and \ \ \frac{\Delta K}{\Delta \mid \mathcal{T} \mid} \leq 0 \tag{11}$$

Back to Equation (10). It seems that a larger terminal set and function set can enhance search efficiency. In fact, there are empirical evidence suggests that this is indeed the case

[1] See [23] for details of the ephemeral random constant.

[2] While the assertion and example given above are based on deterministic functions, we believe that it also holds for the stochastic cases. For example, see [22].

([21]). Still, the influence of search space and population size also has to be taken into consideration. Let $\mathcal{S}$ be the search space, which is a collection of all potential species. The size of it, $\mid \mathcal{S} \mid$, shall grow exponentially with $\mid \mathcal{F} \mid$ and $\mid \mathcal{T} \mid$. Therefore, if population size does not grow exponentially with $\mid \mathcal{F} \mid$ and $\mid \mathcal{T} \mid$, then

$$\lim_{|\Delta\mathcal{F}|,|\Delta\mathcal{T}|\to\infty} s = 0, \tag{12}$$

where

$$s = \frac{\mid G \mid}{\mid S \mid}. \tag{13}$$

To take into account the effect of s, let us rewrite equation (9) into a conditional density,

$$Prob(f^* \in G_n \mid s) = P[K(f^* \mid \mathcal{F} \bigcup \mathcal{T})]. \tag{14}$$

Equation (14) says that the probability of finding f^* in a finite number of generation n is *conditional on* population size ratio s. Since in practice the population size cannot grow in proportion to the size of function and terminal set, reducing algorithmic complexity of a decision rule by enlarging terminal and function set may help gain little efficiency. Therefore, constrained by the population size ratio, the size of $\mathcal{F}$ and $\mathcal{T}$ have to be economized.

What is popular among GP users is to incorporate some known decision rules (*benchmark rules*) into $\mathcal{F}$ and $\mathcal{T}$. For example, in their study of option pricing formula, [15] included the Black-Scholes model, as an existing subroutine, in their function set. The formulas generated by genetic programming for the equity options were hence adaptations of the Black-Scholes model. Examples are

$$C(S, X, \tau, r, \sigma) = C_{B-s} + Constant \ * \ \tau \tag{15}$$

where, C_{B-S} is the Black-Scholes formula and τ is the option's time to maturity.

Another solution to this problem is to make the terminal set or function set adaptive. For the former, [29] showed that how to distinguish relevant terminal sets from irrelevant ones via their adaptation mechanism.[3] For the latter, *automatic define functions* (**ADFs**) provide a promising approach.

3 Semantic Restrictions

Given a function set and terminal set, genetic programming can automatically generate a series of decision rules. While all these decision rules are syntactically correct, they may not be semantically valid or sensical at all if there are no restrictions on *grammar* (*production rules*). Let us illustrate it with a few examples.

Example 1: [16]

[16] used genetic programming as a means of inferring the strategies that were played by subjects in economic decision-making experiments. The game that interested them is the

[3]But, the empirical evidence demonstrated in the paper shows that even regular GP is able to distinguish those "good" terminals from those "bad" ones. Nonetheless, it did this in a less efficient way.

well-known *two-player, repeated ultimatum game*. Given their $\mathcal{F}$ and $\mathcal{T}$, it is possible to generate a decision rule like
(IF((>(A0)(7))))(0)(1)),
which means that if the offer from the proposer (player A) to the responder (player B) is greater than 3, then B will accept it; otherwise she will reject it. This is a typical decision rule which one can expect from the ultimatum game, and hence is sensible. However, their $\mathcal{F}$ and $\mathcal{T}$ can also produce a decision rule like
(IF((<(T)(10)))(B1)(1)),
which means if T is less than 10, the responder will repeat the action taken in the previous period; otherwise she will always unconditionally accept the offer. This decision rule implies that before period 10, the responder will either accept or reject the offer, and after period 10, she will always accept it. In other words, she would not base her decision on the actual offer no matter how attractive it can be. Such decision rules barely make sense.

Example 2: [7]

[7] discussed the semantic issue of using regular GP in technical analysis. Without semantic restrictions, regular GP can generate a decision rule using a comparison between a price and a volatility term, a logical ANDing of a Boolean variable and a price term, or an IF function specifying just a moving-average price term in the condition. None of these expressions make sense.

Example 3: [11]

Using genetic programming, [11] proposed *agent-based computational modeling* of *double auction* (**DA**) *markets*, i.e., a DA market is modeled as an *evolving market* of *autonomous interacting traders* (*automated software agents*). Given his $\mathcal{F}$ and $\mathcal{T}$, regular GP can generate a bargaining strategy like
(* (sin (sin (sin (sin (sin (RLog (If-Then-Else (sin (+ Time2 NT)) Time1 (sin (sin (+ Time2 NT)))))))))) (If-Then-Else Pass (sin (sin (sin (Max PAvg HT)))) Pavg))).
The frequent use of the function sin makes this decision rule barely comprehensible and one can hardly get any useful insights from it.

Since genetic programming is frequently expected to simulate human reasoning processes in agent-based computational economics, generating semantically invalid solutions is somewhat disappointing. There are a few solutions to this problem. The first solution is not to do anything and let the problem be solved by natural forces, i.e., the *survival of the fittest principle*. However, there is no guarantee that natural forces can make these nonsensical decision rules extinct. The reason is simple. Since GP is designed to implement the survival-of-the-fittest principle, GP would remain *loyal* only to the fitnees function provided to it. Unless users' preference are well articulated in the fitness function, there is no magic that GP can figure it out what they are. For example, if fitness is only a function of profits or utilities, then a fair evaluation of GP should be only based on the examination of the profits or utilities of the GP-derived decisions, rather than anything not mentioned in the fitness function, such as the *simplicity* or *semantic validity*.[4]

[4]Therefore, the common criticism to GP, "*the regular GP representation and operators often lead to overly complex solutions*," is indeed not a fair criticism to GP, and certainly cannot be an acceptable argument to reject

Based on what has just been said, the second solution to semantically invalid decision rules is to incorporate into the fitness function *the preference for simplicity or comprehensiblity*. [20] is an example of illustrating the idea of penalizing complex behavior via the well-known minimum description length (MDL) principle, and chose MDL as the fitness function. By using the MDL principle, one may trim down the complexity of evolved behavior, and, hopefully, may make them easier to understand. However, simplicity and semantic validity are not the same thing, and simpler solutions can still be semantic invalid as the example of [16] shown above. So, how should we modify our regular fitness function to reduce the probability of having semantically invalid solutions? The answer is that *we simply do not know*. There is no fitness function which can represent the preference for semantic validity, like what MDL does for the preference for simplicity. It is hard to have such kind of fitness function because measuring semantic validity is a very prohibitive task.

Instead of expressing our preference into the fitness function, alternatively, one can can also put direct restriction to the production rule (the grammar). The third solution proposed by [25] is known as *strongly typed genetic programming* (**STGP**). "Strongly typed GP is an enhancement to regular GP where (i) all the functions know what data types they take as arguments and what data type they return, and (ii) the routines for tree generation, mutation, and crossover all ensure that data types are consistent." ([26], p.334) In economics and finance, STGP is now the major approach to dealing with the semantically invalid solutions. Examples are abounded ([7]; [27]; [8]; [16]).

For example, [7] focused on a special class of non-sensible solutions, i.e., *symmetric trading models* in the foreign exchange market. Regardless of the sign of state variable, symmetric trading models always give the same recommendations to buy or sell, and hence is useless for generating effective trading signals. It is easier to characterize this special class of non-sensible solutions with STGP, and that is what had been done in [7].

Maybe the most profound application of STGP to economic and finance is a series of studies done by Hitoshi Iba and his colleagues. One of their specific application domains, which is also relevant to agent-based computation economics, is the above-mentioned *symbolic regression*. In symbolic regression, one key solution to the choice of primitives is to take a *function approximation approach*. Different series expansion has been considered over the past few years, including *power series*, *Fourier series*, and some *combined series*. In this approach, the choice of primitives is simply a choice of a *basis* (Fourier polynomials, trigonometric polynomial), and the mathematical properties of this choice, such as convergence and density, are well known in functional analysis.

Currently, there are at least two methods of applying the idea of function approximation to symbolic regression. The first one is more straightforward and puts no restriction to the tree structure. So, it behaves like regular GP except that the primitives are selected by the chosen polynomials and the polynomial coefficients are solved by projection. Versions like this can be found in [30] and [31]. The other approach delicately uses a hierarchical multiple regression analysis method, known as *Group Method of Data Handling* (**GMDH**), to add more structure to GP trees. An algorithm called *STructured Representation On Genetic Algorithms for NOn-linear Function Fitting* (**STROGANOFF**) was pioneered by Hiotoshi Iba and his colleague on a series of studies. In [27] STGP was used to generate *Kolmogorov-*

GP.

Gabor Polynomial,

$$f(x_1, x_2, ..., x_d) = \omega_0 + \sum_{j \in S, \forall S \in P(1,2,...,d)} \omega_j \Pi_{j \in S} x_j. \quad (16)$$

By their STGP, all GP-derived solutions will automatically be Kolmogorov-Gabor polynomial, and hence all semantically invalid solution will be avoided.

But **STGP** has its limitation too. The limitation comes from the fact that there exists no universal algorithmic description of nonsensical decision rules. To some extent, even an objective line separating sensible rules from nonsensical ones is difficult to draw. Therefore, it is hard to know the kinds of restrictions to be added *a priori*. Weak restrictions may help little, but strong restrictions may generally diminish adaptation flexibility, which is the essence of agent-based computational economics. Therefore, at its best, **STGP** can only serve to *generate a particular class of decision rules*, rather than as *a general solution* to avoiding semantically invalid rules.

Absence of well-defined fitness is by no means unique to computational economics. Other GP application areas such as criminal suspect search, graphic art, music, architecture design aid, and speech processing also have such problems. When a well-defined fitness function is not available, human evaluation seems to be necessary, and this leads to our next point of discussion: *interactive genetic programming* (**IGP**).

IGP is a kind of regular genetic programming except that the fitness function is replaced by human evaluation. This approach seems to be the only choice when *the fitness function cannot be explicitly defined and is heavily dependent upon the user's implicit preference*. While this idea has been extensively used in other areas, the application of **IGP** to economics and finance is all but virgin territory. The only example known to the author is **EDDIE** ([32]). **EDDIE** (*Evolutionary Dynamic Data Investment Evaluator*) is an *interactive tool*, designed by the University of Essex, to help analysts to search the space of decision trees and make financial decisions. Under this system, the user (expert) may initialize her interaction with the EDDIE by suggesting a terminal set and function set. The EDDIE would then generate decision rules genetically by using GP. The human user may approve or reject these rules based on their experience. This generate-and-approve/reject cycle continues until the user is satisfied. Such a process enhances the semantic validity of solutions.

4 Genetic Operators

In the literature, the EC (Evolutionary Computation) society in particular, a related issue which has attracted researchers' attention is the *design of genetic operators*. Most of the studies in this area were motivated by optimization problems. While the results obtained are suitable for engineering designs, they may be less relevant to agent-based economic modeling.[5]

4.1 The Election Operator

The first study that pointed out the significance of genetic operators to agent-based economic modeling is [1]. [1] introduced the *election operator*, and showed that without this operator

[5]However, as we shall see later, economists did frequently refers to computer scientists' studies from the *optimization* perspective without carefully checking their applicability.

strict convergence to the rational expectations equilibrium cannot be attained in her cobweb model.[6] The idea of the election operator is to "force" well-performing agents to survive for extended periods of time. By doing that, it can prevent evolution from the disturbing effects of crossover and mutation, or the dark side of innovation.[7]

The pioneering works by Arifovic popularized a standard procedure, also known as *the augmented genetic algorithms*, to evolve a population of agents, namely,

reproduction → **crossover** → **mutation** → **election**,

or, written in composite function,

Election (**Mutation** (**Crossover** (**Reproduction**))).

However, other variants also exist. [9] treated *imitation* (*reproduction*) and *innovation* (*crossover* and *mutation*) as two separate learning processes, and run a parallel procedure on both of them. Each process will produce one decision rule, and adaptive agent will decide which she should follow by the election operator. Therefore, their procedure is

Election ((**Reproduction**), (**Mutation** (**Crossover**))).

We see no particular reason why these two procedures could result in different outcomes. In particular, in both procedure, the election operator used as the last step gives the same protection against the disturbance from innovation.

While the election operator was eloquently defended and was extensively used in many applications, sometimes, we see the necessity of not using it. For example, in their study of *endogenous take-off* model, i.e., an endogenous transition from the era of *pre-industrialization* to *industrialization*, [5] did not use the election operator as a final step to protect undesirable disturbance. The *exclusion of the election operator* plays a vital role in explaining why take-off shall eventually happen, "these systems will eventually be attracted to a neighborhood of the high-income steady state *with probability 1*." (p. 199) The key is that mutation and crossover *without further* election will allows agents experiment with positive amounts of training. Since there is no depreciation rate on human capital, this experimentation with positive amount of training ensures that the stock of human capital rises over time. The rising in the stock of human capital will help the economy pass a threshold and take off. However, *before* passing the threshold, since the return (fitness) from investing in human capital is dominated by the return from investing in physical capital, the election operator, had it been used, would kill off all decision rules who call for investing positive amount of time in training. Consequently, the stock of human capital will remain to be zero and a transition from a low-income steady state to a high-income steady state is impossible. So, here, we see the necessity of not using the election operator. Therefore, no matter how eloquently it has been argued before, the use of the election operator still seems somewhat *arbitrarily*.

[6] This is also true in [2], where strict convergence to a perfect-foresight inflation equilibrium can fail without the election operator.

[7] A similar idea of the election operator was proposed by McCain ([24]), which he called it *teleological conservatism*. But, McCain's origin contribution is not acknowledged by users of the election operator. [28] is the only one who cites [24] together with [1].

4.2 Selection Schemes

In addition to the election operator, the importance of the *selection scheme* was also noticed by economists . In the GA society, there are two selection schemes being extensively used. One is the *roulette-wheel selection scheme*, and the other is the *tournament selection scheme*. The performance difference between these two schemes was well evidenced in the context of optimization problems. While one may expect that their differences can also be observed in agent-based economic models, the motivation would not be the same. Because in the context of agent-based economic model, these two selection schemes can represent two different *communication networks*. Roulette-wheel selection is feasible only in a *global network* which allows for imitation among a large number of peoples, whereas tournament selection is supported by a *local network* where imitation is permissible only in a small group of people.[8] As a result, the performance comparison of these two schemes can be useful in addressing the issues related to the network effects.

The first paper who showed that the selection scheme can matter is [6]. The paper is based on the model of a *signaling game*, developed in [17]. The Eaton-White model is a two person game. Each player has a *type*. In a match, the payoff of player j depends on the action she takes and the action her opponent takes. In a case if both players take the appropriate action, then they both receive the highest payoffs. Anyone who take an inappropriate action can hurt not only her opponent but also herself. The *appropriate action* is determined by the *type* of agent, which is not directly observable. The player can, however, send a signal to her opponent regarding her type with the hope that the opponent can gauge the type of the player based on the signal received. Of course, there is a risk: if the player's signal is misinterpreted, then the action taken must also be inappropriate. An interesting issue arose is whether players are able to perfectly decode the signals received and hence can take appropriate actions, or, in brief, coordinate to a Pareto-preferred *signaling equilibrium*.

[6] simulated this game with genetic algorithms, and they found that the result of converging to the signaling equilibrium crucially depends on the *difference in payoffs* between "appropriate action" and "neutral action". Their argument was built upon the roulette-wheel selection scheme.

> There has to be larger difference in the payoffs between 'appropriate action' and 'neutral action' chromosomes in order for the former to obtain more copies during reproduction. Only a large difference in payoffs can offset the large number of neutral chromosomes on a *roulette wheel* that generates copies of chromosomes. (p.193. Italics added)

Now, one can see why the employment of the roulette-wheel selection scheme can matter: it is not able to distinguish the well-performing chromosomes when fitness between the good and bad is close. This can further make it hard to implement the survival of the fittest principle. They did not proceed further to show how their results would changed if the tournament selection scheme is used instead. Nonetheless, in a separate paper, [3] did show how these two selection schemes can result in different results.

The first paper who made a formal remark on the superiority of the *tournament selection scheme* is [9]. The remark is based on a long discussion on selection schemes among *computer scientists*, who tended to concluded that the tournament selection scheme is preferred

[8] This is true because tournament size usually used in the tournament selection is very small.

to the roulette wheel selection. Nonetheless, what is not sure is whether this evidence derived mainly from computer scientists is relevant for agent-based economic modeling. As we mentioned above, in social sciences, these two selection schemes can be connected to two different social networks. This connection may, however, not interest computer scientists in their optimization problem.

[10] showed that, relative to *the survival of the fittest principle*, genetic operators only play a secondary role. Specifically, in a context of the *overlapping generations model*, [13] found that their simulation results on inflation rates are quite robust to the genetic operators used. However, if the survival of the fittest principle is abandoned, the behavior generated changed dramatically and was quite different from the observations based on human subjects.

4.3 Influence from Computer Scientists

What have been summarized above can be considered as economists' contributions to some thoughts on genetic operators. They are not imported from computer science. This shows that the distinction between the GA as an function optimizer and the GA as a simulator of social evolution may calls for different methodologies on the study of genetic operators. However, economists did also received some insights from computer scientists on their application of genetic operators.

For example, on the *population size*, [4] justifies her choice of "30" as follows: "...which is usually taken as the *minimum number* of strings required for *effective search*." (p. 7. Italics added.) Effective search is meaningful when the target to search is clear, and hence is meaningful when we take the GA a function optimizer. But, in agent-based economic modeling, we are not searching for anything in particular. We are simply watching evolution. It is, therefore, not certain whether our experimental designs should be motivated by "effective search".

In addition to sample size, the usual argument to justify the specific control parameters values, such as *crossover rate* and *mutation rate*, is something like "These values are those that are recommended by Back (1986) and Michaelwicz (1996)" (Ibid, p.8), or "These parameter values are consistent with those suggested by Grefenstette (1986) and Goldberg (1989)" ([9], p. 194). But, this "computer scientists say so" argument helps us little in answering the following question: can those control parameters have relevant *empirical values* when agent-based economic modeling is actually used to simulate experiments with human subjects or to replicate some econometric properties from real observations.

5 Architecture

The last issue concerns the economic interpretation of a *population* of building blocks (hierarchies, subroutines, programs or strings). [19] provided the following two interpretations.

> Depending upon the model, an agent may be represented by a single string, or it may consist of a set of strings corresponding to a range of potential behaviors. For example, a string that determines an oligopolist's production decision could either represent a single firm operating in a population of other firms, or it could represent one of many possible decision rules for a given firm. (p.367.)

The first interpretation is often called the single-population design, and the second is called the multi-population design. [1] is probably the first study which compares these different designs of GAs in agent-based economic modeling. In terms of convergence, converging to the rational expectations equilibrium (REE), the single-population GA and the multi-population GA behaves the same, namely, they both converged to the REE when the election operator was used, and failed to converge otherwise. Since then Arifovic no longer considered the MGA in her subsequent studies, and other researchers also did not care too much on whether their simulation results are robust to these two designs. To run the GA within either design turns out to be an arbitrary decision made with almost no qualification.

By saying that "The difference between these two approaches to modeling learning is often neglected...(p.2)" [33] is the first one to give a thorough analysis of the consequences of this choice. In [33], the difference between the SGA and the MGA is more than just a matter of *coding*. They can be different in the interaction *level* at which learning is modeled. For the MGA, learning is modeled at the *individual* level, i.e., agents learn exclusively on the basis of their own experience, whereas for the SGA, learning is modeled at the *population* level, i.e., agents learn from other agents' experiences as well. It is due to this distinction that the SGA is also called *social learning* and MGA *individual learning*. [33] then argued that there is an essential difference between individual and social learning, and the underlying cause for this is a so-called *spite effect*. The spite effect may occur in a social learning GA, but can never occur in an individual learning GA. To see how the spite effect can influence the outcome of the evolutionary process, [33] used the two different GAs to simulate the learning process of a oligopoly game. The simulation results show that while the individual learning GA moves close to the Cournot-Nash output level, the social learning GA converges to the competitive Walrasian output level. As a result, unlike what was observed in [1], the SGA and MGA generally can lead to non-trivial different results.

[18] pointed out a fundamental flaw of the architecture of **SGP** in agent-based economic modeling. A solution to Harrald's criticism is proposed in [14]. Due to the size limit of this paper, the interested reader is referred to [14].

6 Concluding Remarks

A list of four fundamental issues of GP in agent-based economic modeling is given in order. Unless these questions are answered, genetic programming is not well grounded in consideration of human behavior, and it would be premature to claim that we have a model for a population of agents learning over time. The four issues addressed in this paper concern primitives, grammar, genetic operators and architecture. Later breakthroughs in GP such as automatically defined functions, adaptive GP, strongly-typed GP, interactive GP, and distributive GP, can be seen as a series of efforts to cope with these issues.

References

[1] Arifovic, J. (1994) Genetic algorithms learning and the cobweb model, Journal of Economic Dynamics and Control **18(1)**, 3–28.

[2] Arifovic J. (1995) Genetic algorithms and inflationary economies, Journal of Monetary Economics **36(1)**, 219–243.

[3] Arifovic J. (1997) Strategic uncertainty and the genetic algorithm adaptation, . In: Amman H., Rustem B., Whinston A. (Eds.), Computational Approaches to Economic Problems. Kluwer Academic Publishers, Dordrecht, 225–236.

[4] Arifovic J. (1998) Stability of equilibria under genetic–algorithm adaptation: an analysis. Macroeconomic Dynamics **2(1)**, 1–21.

[5] Arifovic J., Bullard J., Duffy J. (1997) The transition from stagnation to growth: an adaptive learning approach. Journal of Economic Growth **2(2)**, 185-209.

[6] Arifovic J., Eaton B. C. (1995) Coordination via genetic learning. Computational Economics **8(3)**, 181–203.

[7] Bhattacharyya S., Pictet O., Zumbach G. (1998) Representational semantics for genetic programming based learning in high-frequency financial data. In: Koza J. R., Banzhaf W., Chellapilla K., Deb K., Dorigo M., Fogel D. B., Garzon M. H., Goldberg D. E., Iba H., Riolo R. (Eds.), Genetic Programming 1998: Proceedings of the Third Annual Conference. Morgan Kaufmann, 11–16.

[8] Bhattacharyya S., Mehta K. (2001) Evolutionary induction of trading models. In: S.-H. Chen (Ed.), Evolutionary Computation in Economics and Finance, Physica-Verlag.

[9] Bullard J., Duffy J. (1998) A model of learning and emulation with artificial adaptive agents. Journal of Economic Dynamics and Control **22**, 179–207.

[10] Chen S.-H. (1997) On the artificial life of the general economic system (I): the role of selection pressure. In: Hara F., Yoshida K. (Eds.), Proceedings of International Symposium on System Life, 233–240.

[11] Chen S.-H. (2000) Toward an agent-based computational modeling of bargaining strategies in double auction markets with genetic programming. In: Leung K.S., Chan L.-W., Meng H. (Eds.), Intelligent Data Engineering and Automated Learning- IDEAL 2000: Data Mining, Financial Engineering, and Intelligent Agents, Lecture Notes in Computer Sciences 1983. Springer, 517–531.

[12] Chen S.-H., Yeh C.-H. (1997) Toward a computable approach to the efficient market hypothesis: an application of genetic programming. Journal of Economic Dynamics and Control **21(6)**, 1043–1063.

[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. (2001) 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.

[15] Chidambaran N., Lee C.-W. J., Trigueros J. (1999) Option pricing via genetic programming. In: Abu-Mostafa Y. S., LeBaron B., Lo A. W., Weigend A. S. (Eds.), Computational Finance – Proceedings of the Sixth International Conference. MIT Press, Cambridge, MA.

[16] Duffy J., Engle-Warnick J. (2001) Using symbolic regression to infer strategies from experimental data. In: S.-H. Chen (Ed.), Evolutionary Computation in Economics and Finance, Physica-Verlag.

[17] Eaton C., White W. D. (1992) Image building. Manuscript, Simon Fraser University and University of Illinois at Chicago.

[18] Harrald, P. (1998) Economics and Evolution. The panel paper given at the Seventh International Conference on Evolutionary Programming, March 25-27, San Diego, U.S.A.

[19] Holland J., Miller J. (1991) Artificial adaptive agents in economic theory. American Economic Review **81(2)**, 365–370.

[20] Iba H., de Garis H., Sato T. (1994) Genetic programming using a minimum description length principle. In: Kinnear K. Jr. (Ed.) Advances in Genetic Programming, Vol. 1, The MIT Press: Cambridge, MA, 265-284.

[21] Johnson H. E., Gilbert R. J., Winson K., Goodacre R., Smith A. R., Rowland J. J., Hall M. A., Kell D. B. (2000) Explanatory analysis of the metabolome using genetic programming of simple, interpretable rules. Genetic Programming and Evolable Machines **1(3)**, 243-258.

[22] Kaboudan M. A. (1999) A measure of time series's predictability using genetic programming applied to stock returns. Journal of Forecasting **18**, 345–357.

[23] Koza J. (1992), Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press.

[24] McCain R. A. (1994) Genetic algorithms, teleological conservatism, and the emergence of optimal demand relations: the case of stable preferences. Computational Economics **7(3)**, 187–202.

[25] Montana D. J. (1995) Strongly typed genetic programming. Evolutionary Computation **3(2)**, 199-230.

[26] Montana D. J., Czerwinski S. (1996) Evolving control laws for a network of traffic signals. In: Koza J., Goldberg D., Fogel D., Riolo R. (Eds.), Genetic Programming 1996: Proceedings of the First Annual Conference. MIT Press, Cambridge, MA, 333-338.

[27] Nikolaev N.I., Iba H. (2000) Inductive genetic programming of polynomial learning networks. In: Yao X. (Ed.), Proceedings of the IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks. IEEE Press, 158-167.

[28] Novkovic S. (1998) A genetic algorithm simulation of a transition economy: an application to insider-privatization in Croatia. Computational Economics **11(3)**, 221–243

[29] Ok S., Miyashita K., Nishihara S. (2000) Improve performance of GP by adaptive terminal selection. In: Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence.

[30] Rodrigruez-Vazquez C., Fonseca M., Fleming P. J. (1997) An evolutionary approach to non-linear polynomial system identification. Im: Proceedings of 11th IFAC Symposium on System Identification, 2395-2400.

[31] Rodriguez-Vazquez, K. (2000) Identification of non-linear MIMO systems using evolutionary computation. In: Genetic and Evolutionary Computation (GECCO'2000), Late Breaking Papers, 411-417.

[32] Tsang E., Li J., Markose S., Er H., Salhi A., Iori G. (2001) EDDIE in financial decision making. Journal of Management and Economics **4**.
http://www.econ.uba.ar/www/servicios/publicaciones/journal4/contents/contents.htm

[33] Vriend N. J. (2000) An illustration of the essential difference between individual and social learning, and its consequences for computational analyses. Journal of Economic Dynamics and Control **24**, 1–19