Evolving Bargaining Strategies with Genetic Programming: An Overview of AIE-DA Ver. 2, Part 2

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

Bin-Tzong Chie
AI-ECON Research Center
Department of Economics
National Chengchi University
Taipei, Taiwan 11623
E-mail: chie@aiecon.org

Chung-Ching Tai
AI-ECON Research Center
Department of Economics
National Chengchi University
Taipei, Taiwan 11623
E-mail: g8258018@mail.nccu.edu.tw

Abstract

The purpose of this paper is to introduce the software system *AIE-DA*, which is designed for the implementation of an agent-based modeling of *double auction markets*. We shall start this introduction with the current version, *Version 2*, and then indicate what can be expected from the future of it.

1 AIE-DA: An Overview

AIE-DA is written in the language of **Delphi**, and is largely motivated by *object-oriented programming* (**OOP**). The use of **OOP** has contributed to the growth of the research area known as *agent-based computational economics*. Its significance is well documented in [7] as follows.

> What is new about ACE is its exploitation of powerful new computational tools, most notably *object-oriented programming*. These tools permit ACE researchers to extend previous work on *economic self-organization* and *evolution* in four key ways. (Italics added).

Based on the idea of *OOP*, the software *AIE-DA* is composed of a series of *objects* with parallel or hierarchical interference. The major objects can be organized into three categories, namely,

- **market architecture**: Market.pas, uMarket.pas, vMarket.pas, PaperToken.pas,
- **agents**: uTrader.pas, Buyer.pas, Seller.cpp, dfGP.pas iFunc.pas,
- **adaptation mechanism**: GPMain.pas, GP.pas, GPV.pas, Pop.pas, Gene.pas, uFunction.pas, uTerminal.pas, Symberg.pas.

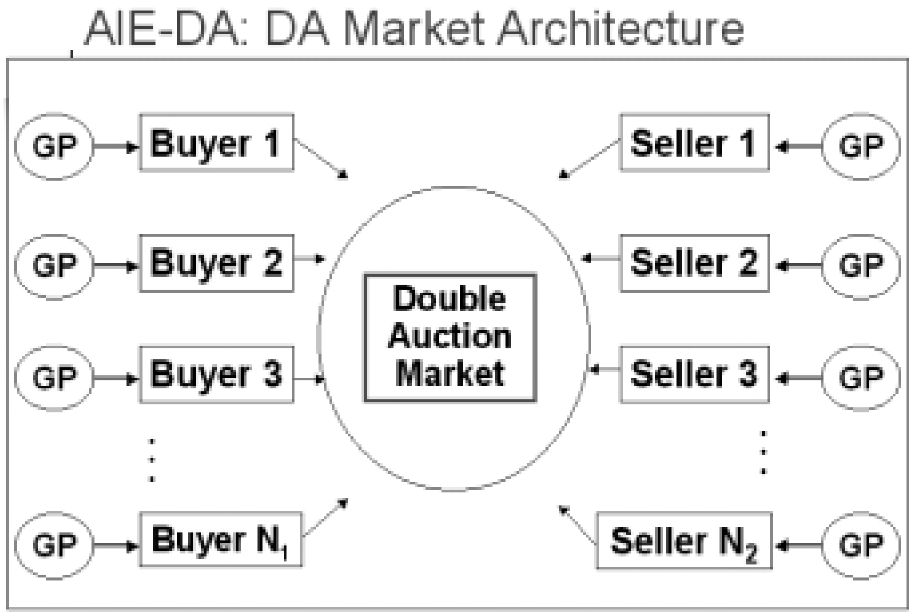


Figure 1. The AIE-DA Architecture: Multi-Population Genetic Programming

The OOP allows us to modify any of these objects without too much involvement in other objects. For example, if it is only the agents' perception our concern, we can basically leave the *market architecture* and *adaptation mechanism* alone.

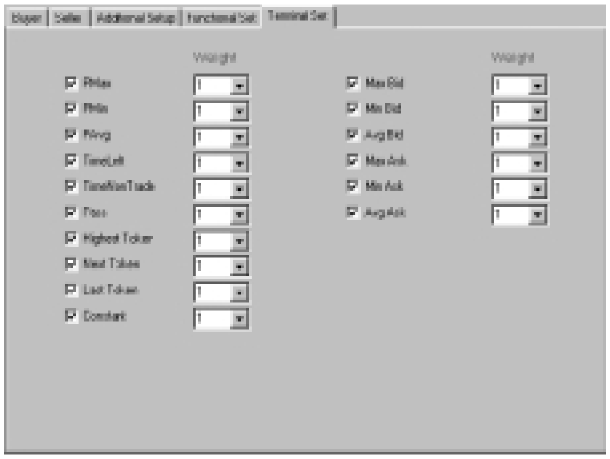
2 Bargaining Agents

All buyers and sellers in AIE-DA are *artificial adaptive agents* in John Holland's sense ([3]). Each artificial adaptive agent is built upon *genetic programming*. The architecture of genetic programming used in AIE-DA is what known as *multi-population genetic programming* (**MGP**). Briefly, we view or model an agent as *a population of bargaining strategies*. The number of bargaining strategies assigned to each bargaining agent, called *population size*, is supplied by the user. AIE-DA Version 2 allows each agent to have at most 1000 bargaining strategies. Genetic programming is then applied to *evolving* each population of bargaining strategies. In this case, a society of bargaining agents consists of many populations of programs. This architecture is shown in Figure 1.

2.1 Primitives: Terminal Set and Function Set

In genetic programming, bargaining strategies 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*. While there is no simple answer for the choice of primitives[1], that choice should at least be consistent with what *traders may know* and *what traders can do*. Based on the SFI market described in Part I of the paper 2 and the SFI double auction tournament reported in [5] and [6], we consider the following sets of primitives relevant.

[1]See [2] for a detailed discussion of it. However, that discussion is mainly motivated by taking GP as a function optimizers. It remains to be examined how the agent-based simulation results may sensitively depend on the set of primitives employed.



	Weight		Weight
✓ PMax	1	✓ Max Bid	1
✓ PMin	1	✓ Min Bid	1
✓ PAvg	1	✓ Avg Bid	1
✓ TimeLeft	1	✓ Max Ask	1
✓ TimeNoTrade	1	✓ Min Ask	1
✓ Pass	1	✓ Avg Ask	1
✓ Highest Token	1		
✓ Next Token	1		
✓ Last Token	1		
✓ Constant	1		

Figure 2. AIE-DA: List of Primitives, the Terminal Set

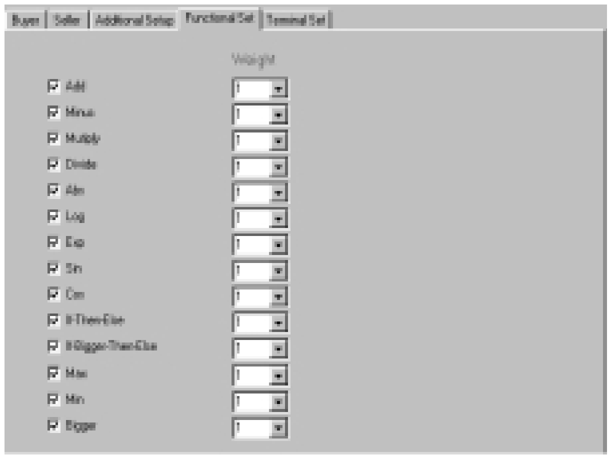
What included in the terminal set are "Highest Token" (**HT**), "Next Token" (**NT**), "Lowest Token" (**LT**), " Current Ask" (**CASK**), "Current Bid" (**CBID**), time left before the termination of a trading period(**T1**), time elapse since the last successful trading (**T2**), the average, the minimum and the maximum price of the previous trading period (**PAvg**, **PMin**, **PMax**), the average, minimum and the maximum bid of the price of previous trading period, (**PAvgBid**, **PMinBid**, and **PMaxBid**), the average, minimum and the maximum ask of the the previous trading period (**PAvgAsk**, **PMinAsk**, and **PMaxAsk**), quiet (**Pass**), and ephemeral random constants. These variables are listed in Figure 2. What included in the function set are arithmetic operators, absolute value, logarithmic and exponential functions, the trigonometric functions (sine and cosine), logical operators, extreme operators, and a comparison operator. These functions are listed in Figure 3.

These primitives are good enough to generate those championship strategies in the SFI DA tournamnet, such as the Kaplan strategy and Ringuette strategy.[2] However, to take into account the possibility that simulation results can sensitively depend on the choice of these primitives at large, AIE-DA Version 2 provides *options* for users. By checking ($\surd$) or not checking the box in front of each primitives, users are able to select a subset of the listed primitives.

2.2 Initialization

Once the primitives are determined, a population of initial strategies are initialized by using the *grow method* constrained by the maximum depth of 5. During the strategy generation process, primitives are randomly selected with a specified distribution supplied by the user. While the default choice refers to the *uniform distribution*, the user may consider other distributions as well by finetuning the weights assigned to each primitive. For example, for the uniform distribution, one simply applies the same weight number to all primitives ("1" in the case of Figure 3). This option design is motivated by the fact that some primitives can be more productive than others. For example, the logical operators are expected to be more crucial for the construction of an effective bargaining strategy than the trigonometric functions. Therefore, the weights assigned for them should be higher. So,

[2]See [1] for details, in particular, Figures 11-13.



	Weight
✓ Add	1
✓ Minus	1
✓ Multiply	1
✓ Divide	1
✓ Abs	1
✓ Log	1
✓ Exp	1
✓ Sin	1
✓ Cos	1
✓ If-Then-Else	1
✓ If-Bigger-Then-Else	1
✓ Max	1
✓ Min	1
✓ Bigger	1

Figure 3. AIE-DA: List of Primitives, the Function Set

this design can serve this purpose.[3] However, we have to say that the real issue involved here is not just a choice of distribution. There is also a concern for *semantic validity*, and by this option, strategies generated are more likely to be semantically valid.

2.3 Evolving Population

Once a population of bargaining strategies is initially generated, these strategies are continuously under review and revision. New bargaining strategies are tried and tested against experience, and strategies that produce desirable outcomes supplant those that do not. This is basically what genetic programming does: generating an *evolving population*. The technicalities involved here are *fitness function*, *selection scheme*, and *genetic operators*.

2.4 Fitness Function

In economics, there is always a nature criterion for fitness evaluation, namely, profits or utilities. In our case, it is the *gains from trade*. Given a token-value table and other opponents' strategies, the gains from trade of using a particular trading strategy Bid_j (Ask_j) in each period of the DA game can be evaluated with Equation 12 (13) in Part I of the paper. Let $\pi^i_{j,s}$ be the gains of using Bid_j in the sth token value table, then the fitness of Bid_j is simply

$$\pi^i_j = \sum_{s=1}^{S} \pi^i_{j,s}, \tag{1}$$

where S is the number of token value tables used in a DA game. Similarly for Ask_j.

2.5 Selection Scheme and Genetic Operator

Once the fitness evaluation of each bargaining strategy is done, the survival of the fittest principle is then implemented via a chosen selection scheme. The most two popular selection

[3] [**?**] proposed a a different approach, which can in effect eliminate some irrelevant terminals when their associated performance validate so.

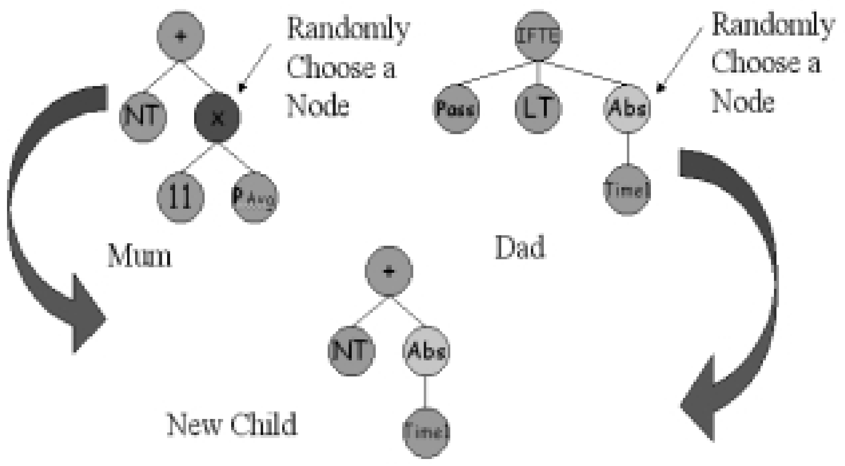


Figure 4. Crossover

schemes are *roulette wheel selection* (*proportionate selection*) and *tournament selection*. The one employed in in AIE-DA is the *tournament selection* scheme. Tournament size is 5. The best of these five is selected as the *mother*, and the second best is singled out as the *father*. This pair of parents will breed a child through the crossover operator. The crossover operator proceeds as follows. First, a node in the mother will be randomly selected, and then what below the node will be completely moved away and is replaced by a *cut-off* from the father, which is also randomly selected. Then an offspring is generated (See Figure 4).

Once an offspring is generated, it will be perturbed with a small probability, i.e., the *mutation rate*. In genetic programming, mutation is commonly proceeded in two ways, namely, the *node mutation* and *tree mutation*. In the case of *node mutation*, we first randomly target a node. Then, if the node is an n-array function node, the n-array function in the node will be replaced with another randomly-selected n-array function. If it is a terminal node, then the terminal in the node will be replaced with another randomly-selected terminal node. For example, in Figure 5, for the left case, the one-array function "Abs" is targeted, and it is replaced with the one-array function "-". For the right case, it is a terminal "Time1" selected, and is replaced with the terminal "LT".

In addition to the normal selection and breeding, the *elitist operator* is also used.[4] For each generation, the best e strategies is directly copied to the next generation without further modification. The number e can be chosen by the users.

2.6 Complexity Regularization

In addition to the survival-of-the-fittest principle, *simplicity* is another major concern. Unlike genetic algorithms, the strategies generated by genetic programming are not strings with fixed length, and they may not even be finite. Therefore, in practice, restrictions is needed to be imposed to avoid over-complex strategies. What has been done in *AIE-DA* is to set an adaptive version of the crossover operator. We, first, measure the *complexity*

[4]In agent-based economic modeling, another frequently used operator is known as the *election operator*. It has been shown that the long-term dynamic property, in particular, the convergence property, can sensitively depend on whether this operator is used. However, the election operator is not applicable here because opponents' strategies are not observable. Therefore, what we take instead is the elitist operator.

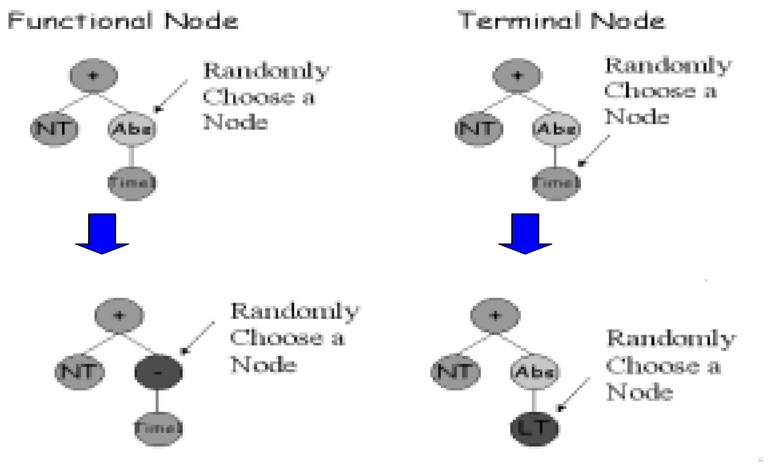


Figure 5. Mutation

of the mother strategy by counting the number of nodes of her LISP tree.[5] Second, if the node complexity is more than 10, and if the cut-off *is not* a function node, we shall try one more time of the random selection of the cut-off. If the code complexity is more than 20, we shall try two more times of the random selection unless a function node is selected. And three more times when node complexity is more than 30. The number of ditto increases with the node complexity in this manner.

References

[1] Chen, S.-H. (2000), "Toward an Agent-Based Computational Modeling of Bargaining Strategies in Double Auction Markets with Genetic Programming," in K.S.Leung, L.-W.Chan, and H.Meng (eds.), *Intelligent Data Engineering and Automated Learning-IDEAL 2000: Data Mining, Financial Engineering, and Intelligent Agents*, Lecture Notes in Computer Sciences 1983, Springer, pp.517-531.

[2] Chen, S.-H. (2001), "On the Relevance of Genetic Programming in Evolutionary Economics," forthcoming in K. Aruka (ed.), *Evolutionary Controversy in Economics towards a New Method in Preference of Trans Discipline*, Springer-Verlag, Tokyo.

[3] Holland J. H., and J. H. Miller (1991), "Artificial Adaptive Agents in Economic Theory," American Economic Review: Papers and Proceedings, 81(2), pp.365–370.

[4] Ok, S., K. Miyashita and S. Nishihara (2000), "Improve Performance of GP by Adaptive Terminal Selection," *PRICAI 2000*.

[5] Rust, J., J. Miller and R. Palmer (1993): "Behavior of Trading Automata in a Computerized Double Auction Market," in D. Friedman and J. Rust (eds.), *The Double Auction Market: Institutions, Theories, and Evidence*, Addison Wesley. Chap. 6, pp.155-198.

[6] Rust, J., J. Miller, and R. Palmer (1994), "Characterizing Effective Trading Strategies: Insights from a Computerized Double Auction Market," *Journal of Economic Dynamics and Control*, Vol. 18, pp.61-96.

[7] Tesfatsion, L. (2001), "Introduction to the Spcial Issue on Agent-Based Computational Economics," Journal of Economics Dynamics and Control, Vol. 25, pp. 281-293.

[5]This is also called *node complexity.* An alternative is *depth complexity.*