Evolving Bargaining Strategies with Genetic Programming: An Overview of "AIE-DA, Version 1"

Shu-Heng Chen
AI-ECON Research Group
Department of Economics
National Chengchi University
Taipei, Taiwan 11623
E-mail: chchen@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 1*, and then indicate what can be expected from the future of it.

Key Words: Double Auction, Bargaining Strategies, Genetic Programming, Object-Oriented Programming, AIE-DA

1 Motivation and Introduction

AIE-DA is written by using SFI (Santa Fe Institute) *Double Auction Market* as the blueprint. Useful references on the SFI double auction market can be found in Rust, Palmer and Miller (1993, 1994). This software system is written in the language of C++, 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 Tesfatsion (2000) 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* (uMarket.cpp, uMarket.hpp), *agents* (uTrader.cpp, uTrader.hpp, Buyers.cpp, Buyers.hpp, Sellers.cpp, and Sellers.hpp) and *adaptation mechanism* (GPV.cpp, GPV.hpp, Pop.cpp, Pop.hpp, GP.cpp, GP.hpp, GENE.cpp, and GENE.hpp). 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 Market Architecture

A specific market architecture, on which *AIE-DA* is built, is a market composed of four buyers and four sellers. Buyers are given the rights and means to buy a number of units of an arbitrary commodity, say *tokens*, and for each token they are given a *redemption value*, which is known only to the buyer of that token. Similarly, sellers are given a number of units of an arbitrary commodity, and each unit has been

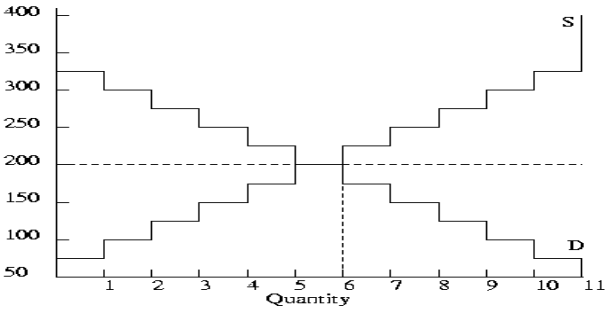

Figure 1: Demand and Supply Curve

assigned a *cost*, which is also known only to the seller of that unit. The array of sellers' costs determines the market supply curve, and the array of buyer's redemption values determines the market demand curve (for example, see Figure 1) .

2.1 Token-Value Generation Process

Traders' token values (buyers' redemption values and sellers' costs for each token) are drawn randomly from the joint distribution F, which is given as follows. The joint distribution F is communicated to players using a four-digit gametype variable. Token values are represented by $T_{j,k}$ where j indexes the trader, and k indexes the token assigned to the trader. Tokens are randomly generated according to

$$T_{jk} = \begin{cases} A + B + C_k + D_{jk}, & \text{if} j \;\; is \;\; a \;\; buyer, \\ A + C_k + D_{jk}, & \text{if} j \;\; is \;\; a \;\; seller, \end{cases} \tag{1}$$

where

$$A \sim U[0, R_1], B \sim U[0, R_2], C_k \sim U[0, R_3], D_{j,k} \sim U[0, R_4]. \tag{2}$$

Each of the four digits of the gametype variable correspond to $\{R_1, R_2, ..., R_4\}$ according to the base-3 coding,

$$R_i = 3^{k(i)} - 1 \tag{3}$$

where $k(i)$ is the ith digit of gametype.

Using *AIE-DA*, token-values tables are automatically generated and are well-ordered. One essential characteristic of *AIE-DA* is to be able to generate more than one token-value table, as it should for

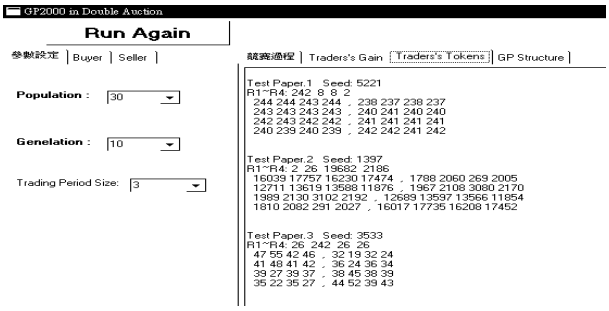

Figure 2: Token-Value Tables

The three token-value tables shown here are generated by using the parameters "**Trading Period Size**" shown on the control panel (the left side).

designing a fair competition process in a later stage. For example, in Figure 2, there are three token-value tables generated by setting control parameters "**Trading Period Size**" to 3. In the current version (Version 1), the trading period size can be set up to 10.

2.2 Agent Representation

Among the four buyers, one, and only one, is assigned as the **GP** buyer. The **GP** buyer is modeled as the *Lucasian version* of an *adaptive economic agent* (Lucas, 1986). Basically, we view or model an agent as *a population of bargaining strategies*. Genetic programming is then applied to *evolve* this population of bargaining strategies. Population size is given by the control parameter "**Population**" in the control panel (Figure 2). The specific example shown in Figure 2 allows the **GP** trader has a population of 30 bargaining strategies.

As to other traders, they are *exogenously* assigned as different level of strategies. For buyers these strategies are all detailed and implemented in the files **Buyer.hpp** and **Buyer.cpp**; for sellers they are done in the files bf Seller.hpp and **Seller.cpp**. At the current version, seven different strategies are prespecified. From the simple to the complex, they are classified into three levels. Consider "**Buyers**" as an illustration. Bargaining strategies from the most simple level (Level 1), represented as *GP parse trees*, to the most complex level (Level 3) are given in Figure 3, 4, and 5 respectively.

Here, we start with very *simple* and *naive* strategies. For example, the left one in Figure 3, "*The Value of the Highest Token Minus One*", would help the buyer earn only one dollar if the trade is successfully made. Certainly, it is not a very ambitious strategy. The right one in Figure 3 is a little aggressive and would help the buyer earn the difference between the highest token value and the next token value plus one dollar. Nevertheless, it can also be *too rigid* so that a match may not occur. As opposed to those in 3, strategies appearing at Figure 4 are a little more *adaptive*, and hence, more complex. All these strategies share one essential feature, namely, they can adapt to the new information revealed from the market, such as "$Time_2$" (number of steps elapsing since the last successful trade), "**CASK**" (the current ask), and "**PvAve**" (the average price in the previous trading period).

The bargaining strategy shown in Figure 5 is effectively similar to the one shown in Figure 4 (the lower left one). The only difference is that it is *complete* in the sense that it does not assume that "current ask" is always available. When it is not available, it will indicate trader an alternative to move rather than taking the *predicate* (either 1 or 0) seriously as the ask or bid.

None of these strategies are comparable to the those submission to the SFI DA tournament in their level of sophistication and complexity. However, there are more close to the *zero-intelligence unconstrained* (**ZI-U**) traders (Gode and Sunder, 1993). When this software is employed as a teaching tool, students are cordially invited to replace these strategies with their own written program. By doing this, students are able to watch the competition between human hand-written, well-thought but not evolving, programs with those initially naive but evolving (growing) machine strategies, and can then evaluate *the significance of evolution*.

3 A Single Simulation

For the purpose of being a teaching software, AIE-DA is highly user-friendly. To have a *run* on the program, simply click "**RUN**" or "**RUN AGAIN**" (as shown in the control panel in Figure 2). A single run represents a single trading round with T trading periods, each with 25 alternating trading **BS** and **BA** steps on each token-value tables. Briefly speaking, each run opens with a Bid/Ask (**BA**) step in which all virtual traders are allowed to simultaneously post bids and asks. After the monitor informs the virtual traders of each others' bids and asks, the holders of the current bid (highest outstanding bid) and the holders of the current ask (lowest outstanding ask) enter into a buy/sell (**BS**) step.

All traders' asks and bids can be shown on screen by clicking on the "**Tournament Process**" (now in Chinese). An example of the time series of each trader's bid or ask is shown in Figure 6. From the leftmost to the rightmost indicates a counting of the trading step (Column 1), buyers' bids (Columns 2-5), sellers' asks (Columns 6-9). If a trade is successfully made, participants' indices will be shown in the

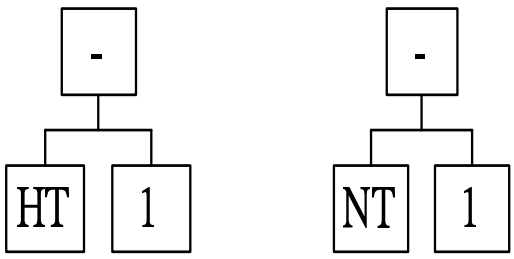

Figure 3: Bargaining Strategies: Level 1, Buyer

next two columns (Columns 10-11) followed by the transaction price (Column 12). If there is no trade achieved, the current bid and the current ask will be reported (Columns 10-11). At the very bottom of this time series is the profits earned by the **GP** trader. For example, in Figure 6, we can see that GP trader earn 423 dollars in the first trading period under the first token-value table. In addition, we can also see that **GP** trader successfully bought two tokens, whereas there are totally seven tokens being traded.

Detail reports on profits can be listed by clicking on "**Traders' Gain**" (Figure 7). In Figure 7, from the leftmost to the rightmost, the first column is a counting of the "trading period". Since, in this illustrative example, all bargaining strategies are tested with three token-value tables in a single trading period, profits performance is presented by bundling three runs, each with a unique token-value table. The buyers' and sellers' profits in each run (corresponding to each token-value table in a single period) are listed in columns 2-5 (buyers), and columns 6-9 (sellers). At each end of bundle, there is a summary of the profits earned by the GP trader. Since the GP trader is represented as a *population* of 30 bargaining strategies, the performance of each strategy is evaluated one by one. For example, in Figure 7, the first one earned a profit 17174 over the first three runs; the second one suffer a loss at -2900 over the first three runs; etc.

4 Evolving Bargaining Strategies

In addition to bargaining behavior (bids, asks and profits), what distinguishes the *computerized DA markets* with the *experimental markets* with human subjects is that these artificial DA markets provides

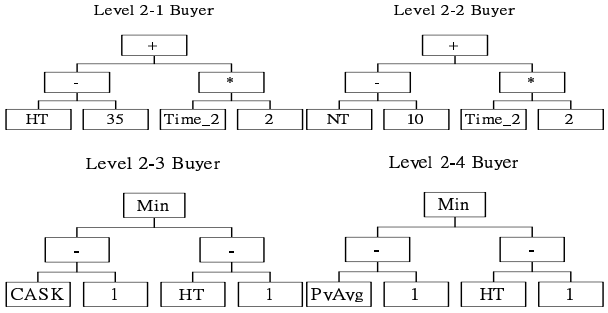

Figure 4: Bargaining Strategies: Level 2, Buyer

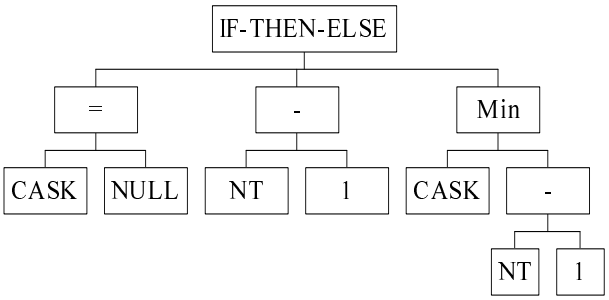

Figure 5: Bargaining Strategy: Level 3, Buyer

us with an opportunity to observe *the evolution (growing) of bargaining strategies*. By clicking "**GP structure**", one can observe the best strategy in each generation (period). The information about the best strategy includes the *LISP program*, *complexity* and the *profits* (*fitness*). In addition to this *elitist*, description of the average complexity and the mean profits of the specific generation are also given.

In the illustration (Figure 8), the best strategy in the initial generation is randomly generated as ((Abs HT)). Notice that it is not a very profitable strategy per se. However, this naive strategy makes the trader have a good chance to win the current holder in the **BA** step, and then the negotiation in the **BS** step may help the trader to earn profits. That is exactly what happens in this showcase. As time goes on, we can see that, in this specific example, the best strategy grows very rapidly in a complex fashion. For example, in Generation 2, it becomes
((Abs (- HT (Min (Min LT (> Pass (sin TimeLeft))) (+ (Min TimeLeft PMax) HT)))))

5 Genetic Programming

To give a further examination of the bargaining strategy, one needs to know the *search space* upon which genetic programming is operated. Search space in genetic programming is, *in principle*, infinite, and is defined by the *function set* and *terminal set*. The function set and terminal set are detailed and implemented in the file **Symbreg.cpp** (Figure 9). The elements in this function set and terminal set are determined by extracting the functions and terminals from by the *Skeleton strategy*, *Kaplan strategy* and *Ringuette strategy*. Rust, Miller and Palmer (1993, 1994) made a report on the performance of these strategies.

The selection scheme used in *AIE-DA* is, by default, the *tournament selection* with a tournament

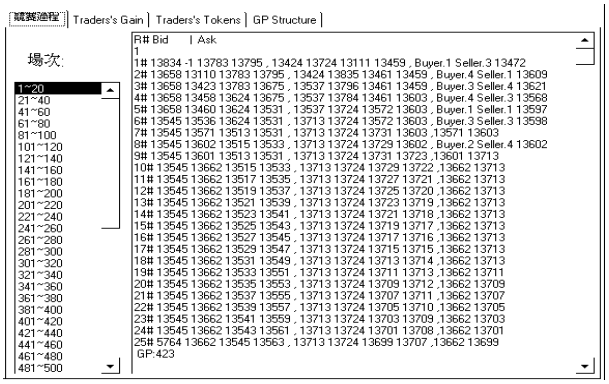

Figure 6: Time Series of Bids and Asks in A Single Trading Period

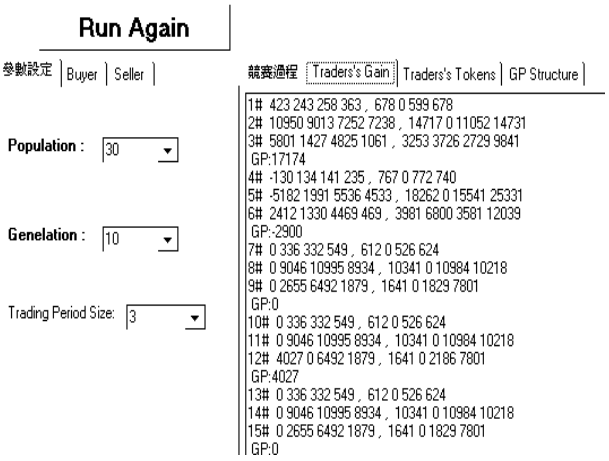

Figure 7: Time Series of Profits Earned by All Traders in Trading Periods

size 5. Given the population GP_{t-1}, 5 trees are *randomly* selected from it. The best two are selected as *parents*, and two children are generated by applying the standard crossover on it. At this moment, the *mutation* operator is not employed here. This selection-and-crossover process is continued until the whole new generate, GP_t, is created. The whole sources codes of *AIE-DA* are extended from **GP C++** (Fraser, 1994).

Using genetic programming to evolve bargaining strategies is not entirely new, Andrew and Prager (1994) and Olsson (1999) conducted similar studies. However, in the case of Olsson (1999), all traders are assumed *homogeneous*, i.e., a *representative buyer* bargaining with a *representative seller*, and the objective function is not *traders' profits* but *social welfare*. This set-up makes the scope of his study very different from what we have here. In spirit, our project is more similar to Andrew and Prager (1994), while differences in technical details between us may not be trivial.

6 Teaching Economics with AIE-DA

The software *AIE-DA* can be used as a standard *teaching toolkit*. One of the fundamental issues in economics is about the origin of *allocative efficiency*, or the operation of the "invisible hand" in real-world markets. To date, a general wide-accepted conclusion is: *only trading institution matters, whereas traders' backgrounds are irrelevant*, i.e., the so-called *intelligence-independent property* (**IIP**), which has been well captured by the terms "*zero-intelligence*" in Gode and Sunder (1993), , "*zero-intelligence plus*" in Cliff (1977) and "*rules of thumb*" in Rust et al. (1993, 1994).

Using *AIE-DA*, we may revisit the issue by asking the significance of *evolution*. Notice that the GP trader starts *with knowing nothing*. All of his bargaining strategies are initially randomly generated. However, the GP trader can *learn from experience*, whereas other traders have only fixed bargaining strategies and are not able to learn. Therefore, this asymmetry provides us a chance to see whether the advantage due to evolution and learning can actually make the GP trader be very competitive in the end. For example, in one simple experiment, by choosing the Kaplan, Ringuette, and Skeleton as the buyer, the profits earned by the GP buyer started with 4895 and then grew up to 7322. In another experiment, the opponents of GP buyer are set as Level 2-1, 2-2 and 2-3, and it is found that the profits of GP traders started with 13608 and ended up with 17502 in the 20th generation.

Certainly, the results can be different from one run to another. It is therefore interesting to know how other traders' bargaining strategies can *grow* the GP trader differently. Also, in addition to players' background, number of generations, population size and other GP parameters may also affect the performance of the GP trader.

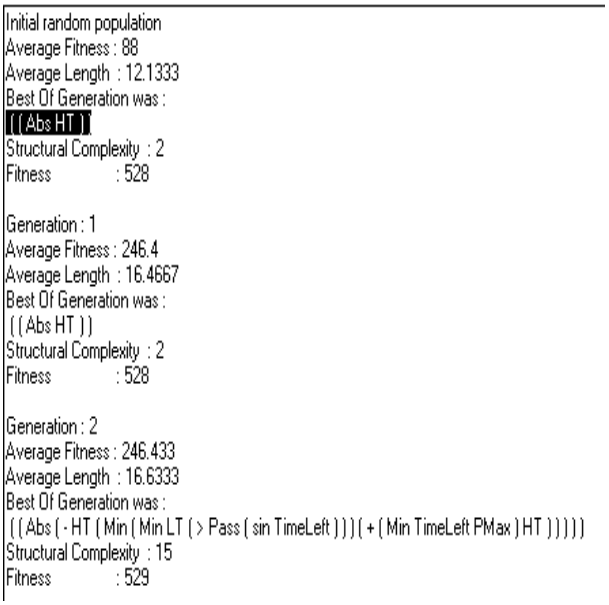

Figure 8: The Profile of the Best Bargaining Strategy

7 Concluding Remarks

In this paper, a documentation of the software *AIE-DA* Version 1 is provided. *AIE-DA* can be downloaded directly from the website:
http://econo.nccu.edu.tw/ai/staff/csh/Software.htm
or by contacting the author directly. We are now considering two extensions of this version. The first one is to introduce a *school* as an *object* to *AIE-DA* so that the social-learning behavior of traders can be simulated (Chen and Yeh, 2000). Furthermore, once the object "school" is introduced, by **OOP**, we shall proceed to establish the interface between the objects "agent" and "school", and after that, an agent-based modeling of DA markets is on the way.

References

[1] Andrews, M. and R. Prager (1994),"Genetic Programming for the Acquisition of Double Auction Market Strategies," in K. E. Kinnear (ed.), Advances in Genetic Programming, Vol. 1, MIT Press. pp.355-368.

[2] Chen, S.-H and C.-H. Yeh (2000), "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*, forthcoming.

[3] Cliff, D. (1997), "Minimal-Intelligence Agents for Bargaining Behaviors in Market-Based Environments," *HP Technical Report*, HPL-97-91, 1997.

[4] Fraser, A. (1994), *Genetic Programming in C++*, University of Salford, Cybernetics Research Institute, Technical Report 040. (The manuscript can be downloaded from the website:
http://www.cs.ucl.ac.uk/research/genprog/

[5] Gode, D. K., and S. Sunders (1993), "Allocative Efficiency of Market with Zero-Intelligence Trader: Market as a Partial Substitute for Individual Rationality," *Journal of Political Economy*, Vol. 101, No. 1, pp. 119-137.

```
if (!(FunctionSets[0] = new FS( 12, _fnAdd,2,
                                    _fnMutiply,2,
                                   _fnMinus,2,
                                   _fnDiv,2,
                                   _fnSin,1,
       //                            _fnCos,1,
                                   _fnLog,1,
        //                           _fnExp,1,
                                   _fnMax,2,
                                   _fnMin,2,
                                   _fnAbs,1,
                                   _fnBigger,2,
                                   _fnIfThenElse,3,
                                   _fnIfBiggerThenElse ,4
                                )
    ) ) ExitSystem( "Initialise");


//  (RandomReal,RendomRealMaxRange only mean 1)

if (!(TerminalSets[0] = new TS( 10,
                               _vPvMax, _vPvMin, _vPvAvg,
                               _vTimeLeft, _vTime_2, _vStop,
                               _vHT, _vLT, _vHT,
              //                 _vPrecent,//0.0~0.99
                                 RandomReal,30
                                 )
                                ) )
   ExitSystem( "Initialise" );
```

Figure 9: Function Set and Terminal Set

[6] Lucas, R. (1986), "Adaptive Behaviour and Economic Theory," in Hogarth, R. M. and M. W. Reder (eds.), *Rational Choice: The Contrast between Economics and Psychology*, University of Chicago Press, pp. 217-242.

[7] Rust, J., R. Palmer, and J. Miller (1993), "Behaviour 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.

[8] 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.

[9] Tesfatsion, L. (2000), "Introduction to the JEDC Special Issue on Agent-Based Computational Economics," *Journal of Economic Dynamics and Control*, forthcoming.