Computational Economics (2006) 28:333–354
DOI 10.1007/s10614-006-9053-3

Revisiting Individual Evolutionary Learning in the Cobweb Model – An Illustration of the *Virtual Spite-Effect*

JASMINA ARIFOVIC[1] and MICHAEL K. MASCHEK[2]
[1]*Department of Economics, Simon Fraser University, 8888 University Drive, Burnaby, BC, Canada V5A1S6; E-mail: arifovic@sfu.ca*
[2]*Department of Economics, Simon Fraser University, 8888 University Drive, Burnaby, BC, Canada V5A1S6; E-mail: mmaschek@sfu.ca*

Accepted 10 July 2006

Abstract. We examine the Cournot oligopoly model in the context of social and individual learning. In both models of learning, firms update their decisions about how much to produce via variants of the genetic algorithm updating procedure. Arifovic (1994) found that both models of social and individual learning converged to the Walrasian, competitive equilibrium. Vriend (2000) reports that the model of social learning converges to the Walrasian equilibrium outcome, while the model of individual learning converges to the Cournot–Nash equilibrium. We revisit the issue and conduct simulations varying elements of the updating algorithms, as well as of the underlying economic model. In the analysis of the outcomes of our simulations, we conclude that the convergence to the Cournot–Nash equilibrium is due to two things: the specific way in which production rules' performance is evaluated coupled with a specific cost function specification.

Key words: social learning, individual learning, spite effect, robustness.

JEL classification: B41; C63; C81; D83; H41

1. Introduction

The models of evolutionary learning have been primarily used to model *social learning* where an entire population(s) evolves through imitation, exchange of ideas, and experimentation. However, as some applications show, these algorithms can also be used to model *individual learning*, where evolution takes place on a set of competing beliefs of an agent. Which evolutionary paradigm is more appropriate depends on the context and on the particular model in question. At the macroeconomic level, it seems plausible that, over time, learning takes place at the level of the economy; i.e. that agents observe each other's decisions and imitate those agents that have been successful in the past. Notions of imitation of successful firms or investors have been around in economic literature for a long time. Social learning represents explicit modeling of these notions. On the other hand, if the objective of research is the examination of strategic interactions in game-theoretic framework, then individual learning may be a more appropriate paradigm.

The Cournot oligopoly is one of the models that have been widely studied in the agent-based literature using different variants of the evolutionary algorithms. The research includes Arifovic (1994), Chen, Shu-Heng and Chia-Hsuan Yeh (1994), Dawid and Kopel (1998), Franke (1998), Brock and Hommes (1998), Vriend (2000), etc. In particular, both Arifovic (1994) and Vriend (2000) study the social learning and individual learning paradigm in this type of economic environment. Arifovic's results showed convergence to the Walrasian competitive equilibrium under both social and individual learning, while Vriend obtains convergence to the Walrasian equilibrium for his social learning model, and convergence to Cournot–Nash outcome for his individual learning model. His study pointed out that we had to be careful about how we applied evolutionary algorithms in the context of social versus individual learning. He explains the difference of the two outcomes in social and individual learning with the existence of the *spite effect* in the social learning model. This same effect is absent from his individual learning model.

The argument is that, in a social learning setup where firms are imitating the firms that earned higher profits, the forces of the underlying Cournot-oligopoly model will drive them towards the Walrasian competitive equilibrium. Thus, when the price is above the equilibrium one (implying that the average per firm quantity and the aggregate quantity are below the competitive equilibrium ones) firms producing higher levels of output will be imitated more frequently and will drive the individual and the aggregate quantities towards the Walrasian equilibrium outcome. Conversely, if the average per firm production and aggregate production are greater than the Walrasian competitive equilibrium ones, the firms producing relatively lower quantities will be imitated, and thus their rules will gain more copies. As a result, the average and the aggregate quantity will be reduced, thus again driving the price towards the Walrasian equilibrium. In Vriend's individual learning setup, while the spite effect still exists in the market itself, it does not interfere with the learning process. As a result, firms are able to reach Cournot–Nash outcome.

On the other hand, in the Arifovic's individual evolutionary learning setup,[1] the spite effect is still present in the learning process itself, and thus, the convergence to the Walrasian competitive outcome takes place. The objective of our paper is to investigate the role of the spite effect in the model of individual evolutionary learning. We start with Arifovic's (1994) model as our baseline model, and then introduce, one at the time, changes to the baseline model, and examine whether these changes have an impact on the existence of the spite effect.

The two approaches differ in a number of implementation details. One of the main differences is the use of hypothetical profits. In the model of individual evolutionary learning, in each time period, a firm chooses one of its rules as its actual decision rule. However, once the market clearing price is determined, all other rules are assigned hypothetical profits, i.e. profits they would have earned in the market at the current market price.

The difference in the way we updated profits in the models of individual learning should be ultimately settled by testing them against empirical data. In this case, the obvious way to do it is through conducting experiments with human subjects. This has now been established as a well recognized and accepted methodology for testing the models of individual learning that have been aimed at modeling behavior in the game theoretic framework. The algorithm that updates only the profits (fitness values) of those strategies in the agent's set that were actually used is very much in the spirit of, for example, *Reinforcement Learning* (Erev and Roth, 1998). Updating the hypothetical profits of all the strategies in the set based on the information from the previous period is in the spirit of *Experience Weighted Attraction Learning*, (Camerer and Ho, 1999). Both of these types of algorithms have been extensively used in the learning literature, and often, compared to and tested against the data obtained in the experiments with human subjects. For an overview, see Camerer (2003).

Arifovic (1994) and Vriend (2000) implement their learning algorithms in the economic environments with different cost specifications. We chose to use Vriend's cost specification and parameter values in order to examine what changes in Arifovic's algorithm are required to obtain Vriend's results in the environment that he used in his work. The parameter values that Vriend used result in the zero Walrasian equilibrium price. In order to examine the behavior of the model in the environment with non-zero Walrasian equilibrium price, we also perform simulations over, what we call, an alternative set of parameter values where we keep Vriend's cost specification, but with non-zero Walrasian equilibrium price. Overall, our findings show that the convergence to the Cournot–Nash equilibrium is due to two things: the specific way in which the performance of firms' rules is evaluated coupled with a specific cost function specification.

Alkemade, La Poutre and Amman (2006) is a second paper in this issue that studies the robustness of evolutionary algorithm design in a version of the Cournot oligopoly game. They consider two approaches to modeling relationship between economic agents and their strategies. According to the first approach, each agent is equated with a strategy. Thus a population of strategies also represents a population of agents that they are representing. According to the second approach, economic agents and population of strategies are separated, and an economic agent can choose from the entire population of strategies. They show that the first approach may lead to premature convergence of the genetic algorithm. The authors argue that this is due to the excessive sensitivity of the first approach to the changes in the evolutionary parameter settings. Their results support the view that the second approach, where agents and strategies are decoupled, is a preferred approach with respect to robustness of the evolutionary algorithm simulations. Both Alkemade et al. and our paper demonstrate the importance of the selection of modeling strategies, and implementation of robustness tests.

The following section (Section 2) contains a brief description of the economic environment in which the subsequent analysis takes place. In Section 3., we first describe Arifovic's (1994) algorithms for individual learning. This is followed by a description of Vriend's algorithm. Finally, the characteristics distinguishing these two approaches are summarized. In Section 4, we describe the results of our simulations. Section 5 contains a detailed analysis of our results. We report our conclusions in Section 6.

2. Description of the Economic Framework

The economic environment under consideration is a partial equilibrium model of a market with a single good where demand is exogenously given, and the supply is determined as an aggregate of firms' individual quantity decisions. As mentioned in the introduction, we adopt Vriend's (2000) cost specification. In the competitive equilibrium version of the model, firms derive their optimal quantity decisions not taking into consideration the quantities produced by other firms. In a Cournot–Nash equilibrium version, they derive optimal quantity decisions utilizing the quantity decisions made by other firms.

The demand side of the market is given by the following inverse demand equation

$$P(Q) = a + bQ^c.$$

where $Q = \sum_{i=1}^{m} q_i$, and q_i is quantity supplied by firm i, $i = [1, \ldots, m]$. Each firm, $i = [1, \ldots, m]$ faces an identical cost function.

$$C_i(q_i) = K + kq_i. \tag{1}$$

Then, the equation for finding the optimal aggregate quantity in case of the Walrasian equilibrium outcome is

$$Q^W = \left(\frac{k-a}{b}\right)^{1/c}$$

and the solution characterizing the Cournot–Nash equilibrium is determined by

$$Q^N = \left[\frac{k-a}{b(\frac{c}{m}+1)}\right]^{1/c}.$$

3. Application of Learning Algorithms

In the following two sub-sections, we outline the algorithms utilized in Arifovic (1994) and in Vriend (2000). We refer to these algorithms as the *individual evolutionary learning* (IEL) and the *classifier system* (CS), respectively.[2] Once we outline the ways the two algorithms work, we will focus on the examination of the type of changes that are required to dampen the spite effect in the IEL model.

3.1. INDIVIDUAL EVOLUTIONARY LEARNING

In the model of IEL, each firm has a population (collection) of decision rules that are represented by binary strings. A decision rule specifies a quantity to be produced. In each period, an active rule is selected using a roulette wheel selection mechanism where each rule's probability of being selected as an active rule is given by its relative fitness. Supplies of individual firms are aggregated, and the market clearing price is computed. Based on this price, a *hypothetical* profit of each rule is computed and becomes its new fitness value. The intuition behind this is the following. Once a firm observes the price level, it looks back at the entire list of rules and re-evaluates them based on the new information that became available in order to see how each one of them would have performed given the actual price level.

Genetic algorithm (GA), see Holland (1992), updating takes place after each period, using *reproduction*, *crossover* and *mutation*. A more detailed implementation of these genetic operators is outlined in the Appendix. Before a new period begins, the firm has to assign *hypothetical* profits or hypothetical fitness values, to the newly generated rules. It does this by using the last period's price level. Thus, these rules can also take part in the selection of the active rule for the following period. The algorithm consists of the following sequence of steps:

- In each time period t, using roulette wheel type of selection, one of the rules is selected to become the rule that determines firm's actual production decision. This rule is referred to as the active rule. The probability that a particular rule is selected is given by its relative fitness, i.e.

$$\mu'_{i,t} = \frac{\mu_{i,t}}{\sum_{j=1}^{n} \mu_{j,t}}$$

- Based on the individual production decisions, and using the inverse demand equation, market clearing price, P_t, is computed.
- Using P_t, hypothetical profit that each rule would have earned had it been used as an actual rule is computed. This is determined taking the market price as given. Then, the fitness of each rule is set equal to this hypothetical profit.
- Updating of the population of rules that will be used at period $t+1$ takes place via the application of the genetic algorithm operators.
  - First, a new collection (population) of N copies of rules is selected using the roulette wheel selection process. This new collection (population) of copies represents the mating pool that undergoes the crossover operator.
  - Two copies are selected out of the mating pool without replacement. Crossover takes place with probability p_{cross}. Thus, $N \cdot p_{\text{cross}}$ rules undergoes application of the crossover operator on average.
  - Next, mutation is performed with probability p_{mut} independently across the positions.[3]

  - Each newly generated binary string is first decoded into an integer number, and the integer value is then normalized to the range of values between 0 and $\overline{q}$ where $\overline{q}$ is the maximum production capacity.
  - Finally, the fitness value of the newly created rules is determined by computing the hypothetical profit that the rule would have earned in the previous time period.

- Each firm uses its population of rules to make production decision at period $t+1$.
- Initial populations of binary strings are generated randomly.

It should be noted that the above description of IEL does not include the election operator that is discussed in the enhanced version of the algorithm described in Arifovic (1994). This operator controls the mutation rate endogenously. The results reported in Arifovic (1994) show that, with the application of this operator, the exact convergence to the competitive outcome can be achieved. In this paper, we look at the convergence in the statistical sense as we did not want to add another layer of complexity to the basic algorithm in our analysis.

3.2. CLASSIFIER SYSTEM

Each firm has again an entire population of rules, represented by binary strings. As with IEL, each rule specifies a quantity to be produced. Each rule has a fitness value associated with it. In each time period, an active rule, the one that determines the firm's production decision, is selected via an auction-like type of mechanism. The chosen rule becomes the active rule in that time period. The market clearing price is computed, and the fitness value of the active rule in each firm's population is updated.

The fitness measure μ is computed as a monotonic transformation of the actual profit level:

$$\mu = \frac{[\log(\pi) - \log(\underline{\pi})]}{[\log(\overline{\pi}) - \log(\underline{\pi})]}$$

where π is the profit level associated with the rule, $\underline{\pi}$ is the lowest possible profit a firm may earn, and $\overline{\pi}$ is the maximum level of profit. In order to determine the minimum profit, the market clearing price is calculated using the maximum production of each firm (which is 2048). This price and the respective level output are used to compute $\underline{\pi}$. In order to determine the maximum profit, the mimimum (non-zero) production allowable is used to the determine the market clearing price. This price is then used in conjunction with the respective output level to determine these extreme values. The fitness values calculated in this way lie in the [0, 1] range.

Thus, the algorithm contains the following sequential steps:

- In each time period, an auction among the rules takes place, in order to determine which one of the rules becomes active. Becoming active implies that this will be the rule that the firm will use to make its production decision in the current period.

  - First, each rule's bid value is determined in the following way:

$$\text{bid} = \text{fitness} + \epsilon$$

    with $\epsilon \cong N(0, 0.075)$. With probability 0.025, the bid is ignored.
  - Next, the highest bid is determined. The rule with the highest bid becomes the active rule, i.e. the rule that determines the firm's action.

- The price that clears the market is computed by summing up over individual firms' production decisions and using the inverse demand equation.
- The profit that each firm earns is calculated and the fitness associated with the active rule is updated.
- Auction, computation of price level and updating of the fitness values of the active rules takes place for g periods ($g = 100$). This is refereed to as the GA-rate.
- Every g periods (referred to as the *GA-rate*), the genetic algorithm updating takes place. The updating takes place on the population of rules for each individual firm i, $i \in \{1, N\}$.

  - The rules' utilities (actual fitnesses) are linearly rescaled to the interval [0,1].
  - From the set of $N = 40$,[4] the roulette wheel selection operator is applied to the 30 fittest rules in order to select two mating parents.
  - With a given probability, p_{cross}, apply the crossover operator to the two strings selected in the previous step.
  - Select one of the two newly generated offspring randomly as a new rule.
  - Assign a fitness value to the new rule that is equal to the average fitness of the two parents.
  - Mutate each bit of the rule with probability p_{mut}, independently across the positions.
  - If the newly generated rule is not a duplicate of the existing rules, replace one of the 10 weakest rules.
  - Decode each newly generated binary string (that becomes the member of the population) into an integer number. The production levels in Vriend's implementation are given by integer numbers only.
  - Repeat these steps until 10 new rules are created. These new rules replace the 10 worst performing rules of the previous period.

- Initial populations of rules are randomly generated.

3.3. MAIN DIFFERENCES BETWEEN IEL AND CS

Table I summarizes the main differences between the individual evolutionary learning model, and the classifier system.

Table I. Differences between IEL and CS.

	IEL	CS
[1] Selection operator	Roulette wheel	Roulette wheel
[2] Binary encoding length	30	11 (*Integer encoding*)
[3] Reproduction eligibility	Entire population	Performance based subset
[4] Duplicate rules	Allowed	Not allowed
[5] Fitness	Level of profits	Monotonic transformation of profits
[6] *Active Rule* selection	Roulette wheel	Stochastic auction
[7(*a*)] GA updating (Rate)	Every period	Every *g* periods
[7(*b*)] Fitness updating	All rules (*hypothetical profits*)	Active rule

Table II. Parameter Specification.

		Baseline	Alternative
Demand parameter	a	−1E-97	−1E-97
Demand parameter	b	1.5E95	1.5E-95
Demand parameter	c	−39.99999997	−39.99999997
Fixed production costs	K	−4.097E-94	−2.060E-84
Marginal production costs	k	0	1.0E-90
Number of firms	N	40	

We wish to determine which of the characteristics distinguishing these two algorithms require inclusion for convergence to the Cournot–Nash equilibrium. Equivalently, our motivation is in determining which of these characteristics require exclusion for convergence to the Walrasian equilibrium.

As such, we begin with what will be referred to as the *baseline simulation* (Simulation 1). This simulation will utilize the IEL algorithm as described in Section 3.1 and the cost specification employed in Vriend (2000). The details of the cost specification are contained in Table II. Note that there is no marginal cost of production under this baseline specification, and a subsidy exists to ensure that the minimum possible level of profits remains positive. This allows for the application of the monotonic transformation of profits inherent in the CS algorithm.

We methodically alter this baseline simulation in order to incorporate the characteristics distinguishing it from the CS algorithm. Each characteristic is incorporated into the baseline simulation in isolation when possible. We consider in more detail each of these characteristics below.

3.3.1. *Integer Encoding of Individual Production Rules – Simulation 2*

An action is represented by a binary string of length k. Each binary string in an individual firm's set represents a level of production that falls between an upper and lower bound ($[\underline{q}, \overline{q}] = [0, 2048]$).

This is accomplished through the application of a normalization parameter. In order to convert an encoded production rule, the binary string representing that rule is first translated into its integer value equivalent. The normalization parameter is then applied to this integer equivalent in order to obtain a production level that is within the relevant bounds. A normalization parameter, $n_{\overline{q}}^{k}$, dependent on the upper bound of production, $\overline{q}$, and the string length utilized for encoding, k, is determined by the following equation.

$$n_{\overline{q}}^{k} = \overline{k}/\overline{q}$$

where $\overline{k}$ is the value of the largest possible integer represented by a binary string of length k.

Higher specifications for the binary strings encoding production levels are associated with a higher precision, or step value, in the resulting rules available to the individual firms. The baseline simulation utilizes a string length, k, of 30 bits. In *Simulation 2*, we use the encoding length implemented in the CS. In this work, production levels are only allowed to be the integer numbers. Using the same upper bound on production, $\overline{q}$, string length, k, is set equal to 11. No other aspects of the baseline simulation are altered.

3.3.2. *Alternative Updating Procedure – Simulation 3*

Reproduction makes copies of individual rules via roulette wheel (proportionate selection). The criterion used in copying is the value of the fitness function. Rules with higher fitness value are assigned higher probability of contributing an offspring that undergoes further genetic operation. Each rule is selected probabilistically according to its relative fitness with replacement. When N copies of the rules are made (the number of rules in a population is kept constant and equal to 40), the reproduction is completed. These copies constitute a *mating pool* which then undergoes application of other genetic operators. Rules are paired, and *crossover* occurs with a given probability. The *mutation* operator follows.

The CS algorithm utilizes a procedure wherein reproduction occurs over the top 30 rules in order to replace the worst performing 10. Rules are ranked in terms of their fitness, and the 10 worst performing rules are dropped from the set. From the set of rules not discarded, two are selected probabilistically according to their relative fitness. With a given probability, these rules undergo crossover (with mutation) and one of the offspring replaces a rule discarded in the previous step. This process continues until the ten discarded rules are replaced. This alternative specification for the updating process of the learning algorithm is implemented in *Simulation 3*. No other aspects of the baseline simulation are altered, save this process of updating.

3.3.3. *Alternative Updating Procedure without Duplicate Rules – Simulation 4*

In the CS's specification of the genetic algorithm, no duplicate rules are allowed in the relevant set. Specifically, each rule in the relevant set must be unique. If in the process of creating new rules, a potential rule is identical to one already in the set, it is thrown away and a new rule is created. Building on *Simulation 3* (where the updating process utilized by Vriend (2000) is substituted for the baseline updating process), *Simulation 4* does not allow duplicate rules within the set of rules available to an individual firm.

3.3.4. *Alternative Fitness Transformation – Simulation 5*

In the baseline IEL simulation, fitness is calculated using the hypothetical level of profit. On the other hand, in the classifier system, a monotonic transformation of the actual profit level in order to determine a rule's fitness measure, μ, is implemented. This way of calculating the fitness values is implemented in *Simulation 5*. All other aspects of the baseline simulation remain the same.

3.3.5. *Alternative Fitness Transformation with Bid Rule Selection – Simulation 6*

Utilizing the transformed profit levels as fitness, Vriend (2000) employs a procedure for selecting the active rule that differs from that of the baseline simulation. Thus, each rule has a *bid*, which is determined by adding to the rescaled values of fitness a random variable drawn from a normal distribution ($N[0, 0.075]$).[5] With a given probability, the bid of a rule is ignored, or tossed out of the set of relevant bids. The remaining bids are ranked and the rule associated with the highest bid becomes the active rule used to determine production level in the current period.

Simulation 6 builds on its predecessor (*Simulation 5*) by utilizing this form of active rule selection. Therefore, with the exception of utilizing the fitness transformation used in *Simulation 5* and the stochastic auction determining the active rule described above, all other aspects of the baseline simulation remain unchanged.[6]

3.3.6. *Increasing the GA-rate and Dropping Hypothetical Profits – Simulation 7*

The GA-rate determines the frequency with which the rule set is updated using the genetic operators. That is, irrespective of whether the relevant rule set is over the individual or over the entire population, the rule set is updated only every g periods, defined and parameterized by the GA-rate. When the GA-rate is not a single period, the fitness of a rule is only updated following its selection as the active rule. As each individual firm has a single active rule in each period, only one rule's fitness is updated in each period per firm. Hypothetical profits are not used. Thus, the GA-rate must be sufficiently large in order to allow rules to be played before selection, and therefore their fitness updated from the previous application of the operator. In *Simulation 7*, we will follow Vriend (2000) in his selection of this free parameter, setting the GA-rate to 100.

4. Results

In order to ensure that the results are robust to different sequences of random numbers, each specification of the individual learning algorithm outlined above is simulated 25 times, each run utilizing a different seed value for the initialization of the random number generator. All simulations have a duration of 200 generations; the duration of simulations as in Arifovic (1994). The exception is simulation number seven where, due to the high specification of the GA-rate, an increased simulation duration of 5000 periods is required (Vriend, 2000). The Gray coding is used for all of the design specifications.[7] In all of our simulations, following Vriend (2000), we used the probability of crossover, $p_{\text{cross}} = 0.3$, and the probability of mutation, $p_{\text{mut}} = 0.033$. In all of the simulations, the number of firms, m, in the market is equal to 40, and the number of rules, N that each firm has is also equal to 40.

Each of the seven specifications of the algorithm are simulated over two distinct specifications for cost. The *baseline* cost specification is adopted from Vriend (2000). Our *alternative* cost specification increases the marginal cost faced by firms from its baseline specification of zero.[8] This also ensures positive Walrasian equilibrium price. The values each cost parameter takes in the baseline and alternative specification are contained in Table II along with the demand parameters that do not change across different specifications. Again, the demand parameters are taken from Vriend (2000). The resulting Cournot–Nash and Walrasian (competitive) equilibria associated with each specification are outlined in Table III. We use two different cost specifications in order to examine the impact that a specific cost parametrization may have on the dynamics of the learning behavior.

For each specification, average per firm production levels and the standard deviation around these average production levels are presented in Table IV. Graphical illustration of the results reported in Table IV are presented in Figure 1 where

Table III. Equlibrium quantities and prices.

	Baseline		Alternative	
	Quantity	Price	Quantity	Price
Cournot–Nash	942.4	1.3313E-88	629.83	1.3336E-81
Walrasian	1593.5	0	1065.0	1.0E-90

Table IV. Individual learning – average per firm output (in thousands).

Simulation	Baseline	Alternative
1	1.5482 (0.1347)	1.1007 (0.1057)
2	1.5474 (0.1349)	1.1044 (0.1055)
3	1.5837 (0.0320)	1.0648 (0.0218)
4	1.5831 (0.0323)	1.0660 (0.0230)
5	1.4790 (0.1132)	1.0159 (0.1571)
6	1.4819 (0.0896)	1.0592 (0.0950)
7	*1.0935(0.0832)*	1.0228 (0.0881)

Walrasian and Cournot–Nash equilibrium quantities are represented by straight line. Average quantities (over 25 runs for each design and cost specification) with plus and minus one standard deviation are presented as vertical lines for each simulation design that are enumerated from 1 to 7 on the x-axis.

Table IV and Figure 1 show that the average quantities reach the values close to the Walrasian equilibrium values in simulation designs 1–6, for both cost specifications. The only simulation that converges to the Cournot–Nash equilibrium is Simulation 7, for the baseline cost specification (within two standard deviations). With alternative cost specification, the outcome of the Simulation 7 design is again the Walrasian equilibrium. It appears that for convergence to the Cournot–Nash equilibrium, the IEL framework requires *both* a cost specification that has no marginal costs and a learning algorithm that does not use hypothetical profits.

Figures 2–6 illustrate the behavior of the average per firm output observed in simulations 2–6 for the alternative cost specification. The two straight lines indi-

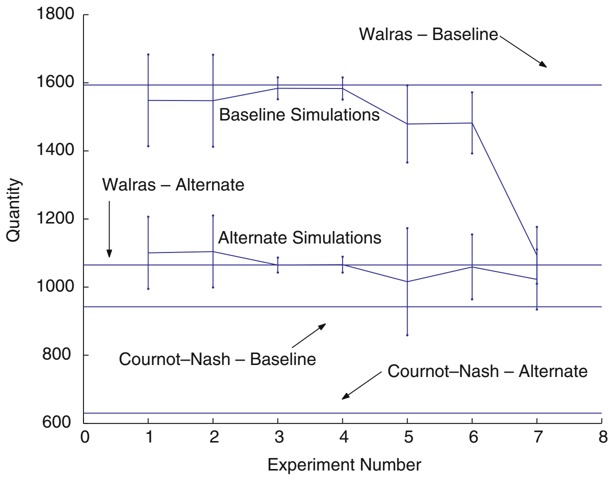

Figure 1. Graphical representation of Table IV.

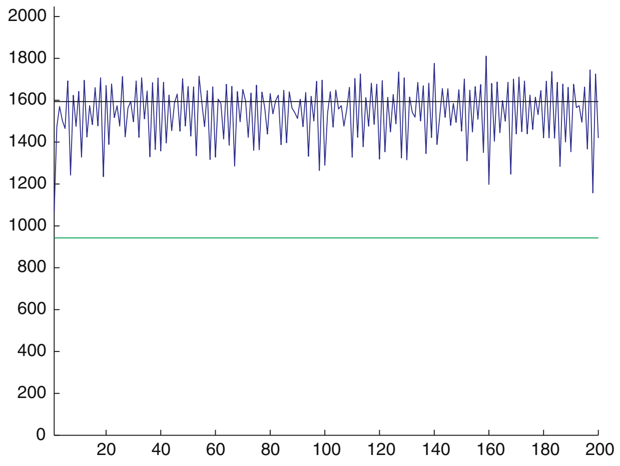

Figure 2. Average per firm output – Simulation 2 (Baseline cost specification).

cate the Cournot-Nash per firm quantity (the lower), and the Walrasian competitive equilibrium quantity (the higher). They show that the average per firm output is in the neighborhood of the Walrasian competitive outcome. Figures 7 and 8 illustrate the behavior of the average per firm output for the simulation design 7. As can be observed in Figure 7, which represents a simulation with the baseline cost specification, the average output remains close to the Cournot–Nash equilibrium outcome. In Figure 8, which represents a simulation with the alternative cost specification, we can see that the average output is back close to the Walrasian outcome for the alternative cost specification. (Note that all of the Figures, 2–8,

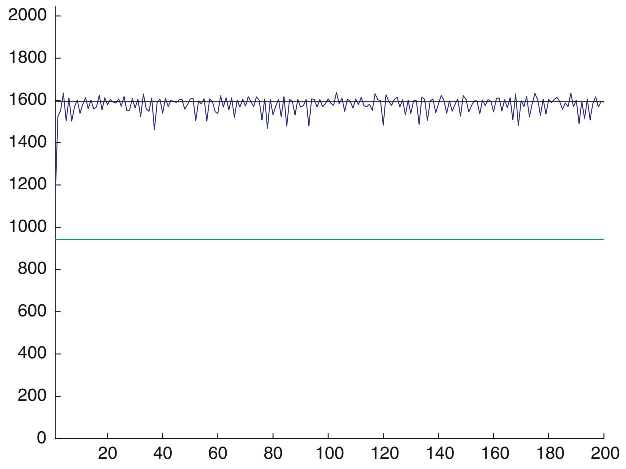

Figure 3. Average per firm output – Simulation 3 (Baseline cost specification).

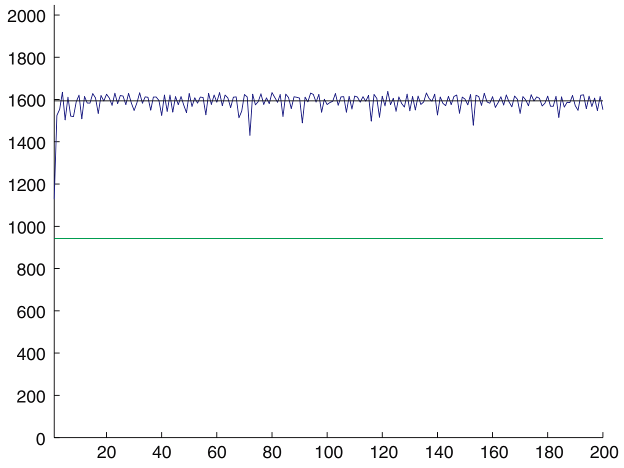

Figure 4. Average per firm output – Simulation 4 (Baseline cost specification).

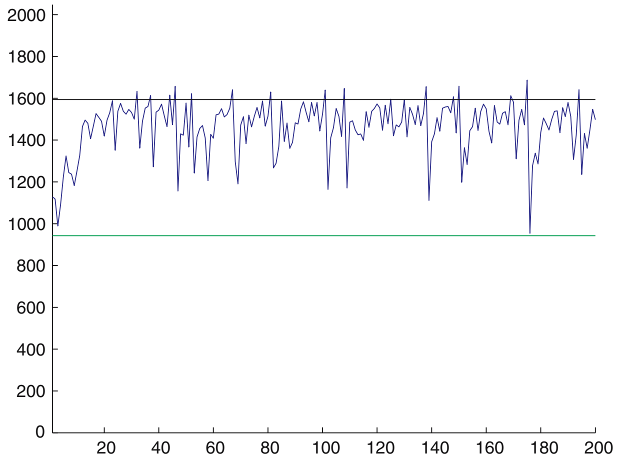

Figure 5. Average per firm output – Simulation 5 (Baseline cost specification).

represent behavior observed in one of the 25 runs of each of different simulation designs.)

5. Analysis

According to Vriend (2000), the difference between social and individual learning lies in the fact that, while social learning is exposed to the *spite effect* inherent within the Cournot framework, the rules that evolve in the individual model of learning are not directly exposed to this effect. The effect drives a wedge between the results for simulations utilizing the individual and social learning algorithms.

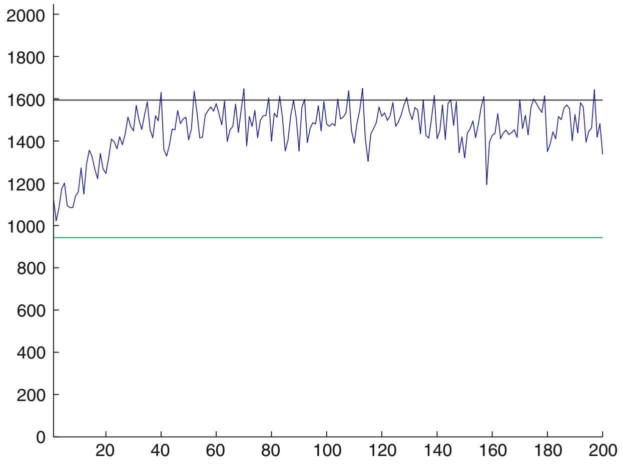

Figure 6. Average per firm output – Simulation 6 (Baseline cost specification).

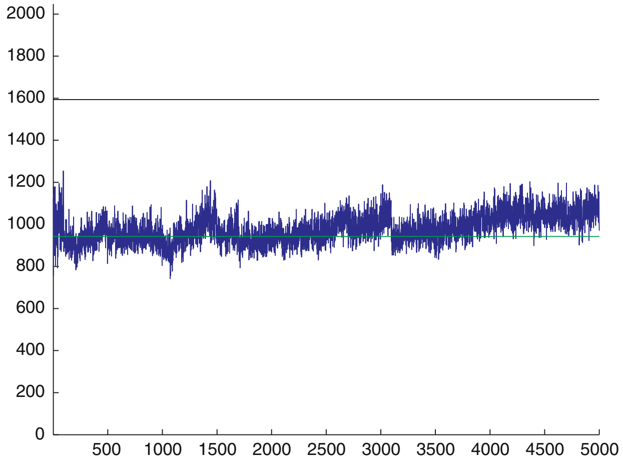

Figure 7. Average per firm output – Simulation 7 (Baseline cost specification).

5.1. THE SPITE-EFFECT

The spite-effect in the social lerning environment works in the following way. Suppose that the average output per firm is lower than the Walrasian competitive equilibrium quantity. This implies that the aggregate quantity is also going to be lower (and the market clearing price higher) than the Walrasian equilibrium values. If we have in mind an agent-based model with heterogenous decisions about how much to produce, firms that are producing above this average quantity are going to earn higher profits. Consequently, as a result of reproduction in a GA setup, they will be imitated by more firms, and thus the aggregate production level will increase. This will drive the price towards the Walrasian competitive equilibrium level.

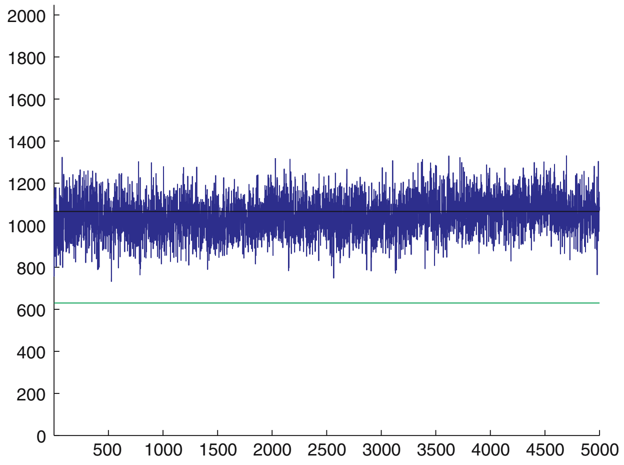

Figure 8. Average per firm output – Simulation 7 (Alternative cost specification).

Now, suppose that the average output per firm is higher than the Walrasian competitive equilibrium quantity. As a result, the price is going to be below the Walrasian equilibrium levels. In this case, firms producing below this average quantity are going to be more frequently imitated during the GA updating. As such, the individual and aggregate quantities will decrease. A decrease in the aggregate quantity will drive the market clearing price towards the Walrasian equilibrium level. As a result, a population of social learners will converge to the Walrasian output.

Vriend (2000) argues that there is no existence of the spite-effect in his individual learning model. In this model, the individual rules that compete with each other in the learning process do not interact in the Cournot market. In each time period, a firm uses only one rule and evaluates its performance. Hence, instead of looking how well other firms with different rules were doing, a firm now checks how well it had been doing in the past when it used these rules itself. The spite effect will still exist in the Cournot market, but will not effect the learning process itself.

However, what we call *virtual spite-effect* exists in the IEL models. First, consider Arifovic's IEL model where all of the hypothetical profits are updated in each time period. The market-clearing price that can be above or below the Walrasian equilibrium price will affect the fitness values of the firm's decision rules. Suppose that the market price is above the Walrasian equilibrium price. Regardless of what the actual rule a firm chose, it is going to evaluate all of the rules in its set. Thus, the rules that proposed higher production will be rewarded by more copies during the reproduction process, and will also increase their probability of being chosen as active rules in the following period. This, in turn, would increase the aggregate supply and lower the market clearing price. Hypothetical profits and resulting fitness values of the rules recommending relatively high production would go down, and the rules recommending relatively low production would be promoted in each firm's collection. As a result, aggregate supply would fall, pushing the market clearing price upward, toward the Walrasian equilibrium value (If there is slight overshooting, the same process will be repeated.)

On the other hand, if the market price is below the Walrasian equilibrium level, regardless of what the firm actually chose, all of the rules' fitness values will be updated using that price. As a result, a population of firm's decision rules will be driven towards producing less. This will also result in a higher probability of selection of the rules that propose lower quantities. Overall, the virtual spite-effect will still work its way through a model where firms update the hypothetical profits of all of their rules.

5.2. NECESSARY CONDITIONS FOR COURNOT-NASH CONVERGENCE

Next, we argue that both exclusion of hypothetical profits (fitness values), and utilization of a specific cost specification (baseline) are conditions necessary for a reduction of the impact of the virtual spite-effect that is sufficient for the convergence to the Cournot–Nash equilibrium outcome.

We begin with a thought experiment in which the simulation is at or near the competitive equilibrium outcome. Assume that one firm experiments (via the mutation operator) with a rule that reduces its production level below the equilibrium level. This pushes the market price above that of its competitive value. Under the zero marginal cost assumption of the baseline model, prices are now positive.

Consider first the experimenting firm. As hypothetical profits (and associated fitness values) are not utilized, and only the rule played by this firm has its fitness updated. This fitness is below that of the old rule the agent was playing prior to this experimentation. As such, the old rule has higher fitness, and is more likely to be re-adopted rather than playing the deviating rule. There is, therefore, pressure on the firm to revert back to the equilibrium strategy.

Now, consider all other firms in the simulation. These individuals were still playing the equilibrium strategy. As the experimenting firm pushed the price into positive territory, the fitness of the equilibrium rule now increases. As hypothetical profits and fitness values are not utilized, no other rules' fitness is altered. As such, the equilibrium strategy gains relative fitness against all others. It remains likely to be played in the subsequent periods.

Taken together, there is a significant pressure in the simulation to revert back to the competitive equilibrium. The equilibrium is stable with respect to a single firm's deviation from an equilibrium strategy to one in which they supply slightly less. This is not the case, however, if this deviation is towards producing more output.

Now, let us consider a single firm deviating from the competitive equilibrium strategy towards producing slightly more. Under the baseline cost specification, this pushes prices into negative territory. Again, consider first the agent that deviated. The strategy played will obtain lower fitness than that of the equilibrium strategy. Its fitness is updated. Again, only the rule representing the deviation has its fitness updated. As this rule is now associated with a fitness value less than that of the equilibrium strategy, it becomes less likely to be played. As in the previous deviation, the firm is now likely to revert to her equilibrium strategy.

Next, consider all other firms not deviating from the competitive equilibrium. The fitness of the rule associated with this equilibrium now falls considerably. (This is opposite of the observation in the previous deviation we considered.) This rule's fitness falls, and it makes the equilibrium rule less likely to be played in the subsequent periods. As all other firms are less likely to play the equilibrium rule, it becomes more likely for these agents to experiment with deviations from the equilibrium production value.

This is where the particular, baseline specification of costs becomes important. Thus far, the argument made applies equally well in qualitative terms to a situation, in which equilibrium price is positive. However, when prices are pushed into negative territory due to a single firm's deviation, the profit maximizing output level for all other firms becomes zero. This has the effect of encouraging rather large cuts in production. That is, for the firms that begin experimenting with non-equilibrium strategies, those whose rules propose the highest cut in the production will observe the highest fitness values for these rules in the subsequent periods assuming prices remain negative.

This is not the scenario when equilibrium prices are positive. Here, when deviation encourages a cut in production, this cut does not have to be as drastic as that associated with a zero price simulation, as the profit maximizing level of output will not be zero.

The cost specification of Vriend encourages and rewards very big cuts in production, pushing the simulation towards the Cournot–Nash equilibrium. Aggregate production falls considerably, and as such those playing the reduced output strategies enjoy the higher profits associated with cooperative cuts in output. As big cuts were rewarded, the Cournot–Nash outcome is fostered.[9]

However, sustaining the Cournot–Nash outcome requires sustained coordination on lower individual quantities. If this is the case, why does the simulation then not proceed toward converging back to the competitive equilibrium? The answer is linked to the fact that hypothetical profits or potential fitnesses are not being utilized.

If the simulation is to drift back towards competitive outcomes, individual firms must begin increasing production levels. The likelihood of these strategies being played is directly linked to the fitness level associated with them. However, these fitness values are only updated when they are played. As such, when firms consider playing these rules, the fitness value they associate with them is that remembered from the last instance they were played. Firms only "remember" the situation in which price was pushed into negative territory and these rules suffered low fitness values. As such, the likelihood of adopting these strategies may be very low, aiding to sustain the low production simulation outcome.

Thus, although there will still be a virtual spite effect, it is mitigated by two factors: only updating the fitness of those rules that are actually used, and the cost specification with zero marginal cost.

In contrast, when we consider firms that update the hypothetical profits of all of their rules based on the newly available information, the virtual spite effect becomes an important characteristic within the model of individual learning. The same forces that work at the level of social learning will now be in place on the sets of rules updated by individual firms. This implies that greater speeds of learning, which characterize the model where all hypothetical profits are updated enables the existence of spite effects. Being slower, and 'less sophisticated', might in this case help reach the preferred outcome.

6. Conclusions

We investigated the changes that are required for the convergence of the individual evolutionary learning algorithm to the Cournot–Nash equilibrium. Two things appear to be crucial for this convergence: one is the specific way in which the fitness function is updated, and the other is the specific cost function. The fitness updating requires that only the fitness values of those strategies that are actually used for production decisions are updated. The required specification of firms' costs is one that results in the zero (Walrasian) equilibrium price level. As our simulations show, both of the above are required in order to dampen the spite-effect and allow for the convergence to the Cournot–Nash equilibrium. For example, if this fitness updating is combined with the specification of the economic parameter values that result in the positive equilibrium price, we do not observe departures from the Walrasian equilibrium. Moreover, implementation of the cost specification associated with a zero equilibrium price level and an alternative updating of rules' fitness (one in which all strategies update their fitness utilizing hypothetical profits) again results in the Walrasian equilibrium outcome.

The question of whether a certain model of individual learning is appropriate rests on testing them against empirical data. The existing experimental evidence (e.g. Wellford, 1989) suggest that under the conditions that correspond to the individual learning setup, i.e. where human subjects are not able to observe the production decisions of other participants, the quantities and prices converge to the Walrasian competitive equilibrium outcome.

Notes

[1] This algorithm is referred to as the multiple-population algorithm in Arifovic (1994). However, we will refer to it as the individual evolutionary learning algorithm as this term

better reflects the developments and applications in the learning literature. For an application of a very similar learning algorithm see Arifovic and Ledyard (2004).

[2] We choose these names for the following reasons. Arifovic and Ledyard (2004) have recently used the type of algorithm very similar to Arifovic (1994) algorithm, and have called it Individual Evolutionary Learning. Vriend (2000) refers to his individual learning algorithm as a type of Classifier System.

[3] More detailed account of reproduction, crossover and mutation is provided in the Appendix.

[4] Vriend (2000) used 40 as the number of firms in the social learning implementation. He also kept 40 firms in the individual learning implementation. In addition, each of the firms had a collection of 40 production rules. We decided to adopt the same numbers.

[5] As descrubed earlier, fitness values are linearly rescaled from their existing values to the [0, 1] interval.

[6] The transformation of profit levels associated with Simulation 5 is employed in Simulation 6.

[7] All of our results remain unchanged with binary coding only. We introduced Gray coding at the request of one of our referees. A short description of how it works and why it is useful in dealing with genetic algorithm binary strings is given in the Appendix.

[8] In order to ensure that the lowest possible profit level remains positive, we must increase the size of the subsidy each firm faces. Hence, the positive marginal cost specification is associated with a more negative fixed cost of production.

[9] What is required is large cuts in production. In this scenario, this is fostered by the particular cost specification of Vriend. However, other possibilities exist. For example, if one was to reduce the binary string length for the genetic encoding of the simulation and adjust the normalization parameter appropriately, deviations in output would have a larger step value. As such, small deviations may not be made possible and Cournot–Nash outcomes may be observed.

Appendix

Reproduction makes copies of individual chromosomes via roulette wheel (proporionate selection). The criterion used in copying is the value of the fitness function. Chromosomes with higher fitness value are assigned higher probabiity of contributing an offspring that undergoes further genetic operation. Thus, the probability that a chromosome $A_{i,t}$ will get a copy $C_{i,t}$ is given by:

$$P(C_{i,t}) = \frac{\mu_{i,t}}{\sum_{j=1}^{n} \mu_{j,t}} \quad i = 1, \ldots, n.$$

The algorithmic form of the reproduction operator is like a biased roulette wheel where each string is allocated a slot sized in proportion to its fitness. The number of spins of the wheel is equal to the number of strings in a population. Each spin yields a reproduction candidate. Once a string is selected, its exact copy is made. When n copies of strings are made (the number of strings in a population is kept constant), the reproduction is completed. These copies constitute a *mating pool* which then undergoes application of other genetic operators.

Crossover exchanges parts of pairs of randomly selected strings. It operates in two stages. In the first, two strings are selected from the mating pool at random. Then in the second stage, a number k is selected, again, randomly from $(1, \ldots, l-1)$ and two new strings are formed by swapping the set of binary values to the right of the position k. The total of $n/2$ (n is an even integer) pairs are selected and the crossover takes place on each pair with probability p_{cross}. An example of the crossover between two strings for $l=8$ and $k=4$ is given below:

$$\begin{array}{cccccccccc} 1 & 0 & 1 & 0 & | & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & | & 0 & 0 & 1 & 0 \end{array}$$

After the crossover is performed, the two resulting strings are:

$$\begin{array}{cccccccc} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \end{array}$$

Mutation is the process of random change of the value of a position within a string. Each position has a small probability, p_{mut}, of being altered by mutation, independent of other positions.

It works in the following way: A random number [0, 1] is drawn. If the number is less than or equal to the probability of mutation, p_{mut}, the value of the position is changed. If it is 1 it is changed to 0, and if it is 0, it is changed to 1. If the random number turns out to be greater than p_{mut}, the value of the position is left unchanged. This is repeated ln times, i.e. for every position in every string.

GRAY CODING

In general, adjacent integers in the binary representation often lie many bit mutations apart. It is therefore less likely that a mutation operator can effect small changes for binary-coded strategies or decisions. Gray coding is often used as a way to deal with this problem. It is an alternative encoding method in which adjacent integers differ by only a single bit (the Hamming distance between adjacent integers equals one). This is referred to as the adjacency property. For example, the binary coding of 7 is 000, 001, 010, 011, 100, 101, 110, 111, while a potential gray coding is 000, 001, 011, 010, 110, 111, 101, 100. Essentially, a Gray code takes a binary sequence and shuffles it to form a new sequence in which the adjacency property holds. Their use in the implementation of genetic algorithms has been shown to improve the performance over implementation utilizing binary encoding. This performance improvement is grounded on the increased potential for small perturbations through successive single mutations of the encoded string (see Holstein, 1971, for a consideration of genetic algorithm performance utilizing Gray-coded integers in a pure mathematical optimization problem). In general, adjacent integers in the binary representation often lie many bit mutations apart, it is therefore less likely that a mutation operator can effect small changes for binary-coded strategies or decisions.

References

Alkemade, F., La Poutre, H. and Amman, H. (2006). Robust evolutionary algorithm design for socio-economic simulation. *Computational Economics,* (forthcoming).

Arifovic, J. (1994). Genetic algorithm learning and the cobweb model. *Journal of economic dynamics and control*, **18**, 3–28.

Arifovic, J. and Ledyard, J. (2004). Scaling up learning models in public good games. *Journal of Public Economic Theory*, **6**, 205–238.

Brock, W. and Hommes, C. (1998). A rational route to randomness. *Econometrica*, **65**, 1059–1097.

Camerer, C. and Ho, T. (1999). Experience weighted attraction learning in normal form games. *Econometrica*, **67**, 827–873.

Camerer, C. (2003). *Behavioral Game Theory: Experiments in Strategic Interaction*, Princeton University Press.

Chen, S.-H. and Yeh, C.-H. (1994). Genetic programming learning and the Cobweb Model. in P. Angeline and K.E. Kinnear, Jr. (eds), *Advances in Genetic Programming 2*, MIT Press, Cambridge, MA, chapter 22.

Dawid, H. and Kopel, M. (1998). The appropriate design of a genetic algorithm in economic applications exemplified by a model of the Cobweb type. *Journal of Evolutionary Economics*, **8**, 297–315.

Erev, I. and Roth, A. (1998). Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. *American Economic Review*, **80**, 848–881.

Franke, R. (1998). Coevolution and stable adjustment in the Cobweb model. *Journal of Evolutionary Economics*, **8**, 383–406.

Holland, J.H. (1992) *Adaptation in Natural and Artificial Systems. An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, 2nd edition. Cambridge, MA: MIT Press.

Vriend, N. (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.

Wellford (1989). A Laboratory Analysis of Price Dynamics and Expectations in the Cobweb Model, Discussion Paper No. 89–15. University of Arizona: Tuscon.