

Adaptation in the Muth Model

Jasmina Arifovic
Simon Fraser University

September 2002
Notes for Econ 428

September 15, 2002

1 Description of the model

There are n firms in a competitive market that are price takers and that produce the same good.

Since the production takes time, quantities produced must be decided before market price is observed.

The cost of a production of a firm i is given by:

$$C_{i,t} = xq_{i,t} + \frac{1}{2}ny(q_{i,t})^2 \quad (1)$$

where $C_{i,t}$ is a firm i 's cost of production for sale at time t , $q_{i,t}$ is a quantity it produces for sale at time t , and x and y are parameters greater than zero. The profit of an individual firm, $\Pi_{i,t}$, is:

$$\Pi_{i,t} = P_tq_{i,t} - C_{i,t} \quad (2)$$

where P_t is the price of the good at time t , and $P_tq_{i,t}$ are revenues that a firm i earns by selling quantity $q_{i,t}$.

Substituting (1) into (2), we obtain the following expression for the profits:

$$\Pi_{i,t} = P_tq_{i,t} - xq_{i,t} - \frac{1}{2}ny(q_{i,t})^2 \quad (3)$$

How does a firm make a decision about the quantity it produces at t ? The objective of the firm is to maximize its profits. So, at $t - 1$, each firm chooses a quantity $q_{i,t}$ to maximize its expected profit $\Pi_{i,t}^e$ on the basis of its expectations, $P_{i,t}^e$, about the price that will prevail at time t . (The exact way in which the price expectations are formed will be discussed later.)

To find the quantity that maximizes firm's expected profits, the derivative of the expected profit function with respect to quantity is set to zero, i.e.

$$\frac{\partial \Pi_{i,t}^e}{\partial q_{i,t}} = 0 \quad (4)$$

or

$$P_{i,t}^e - x + ynq_{i,t} = 0 \quad (5)$$

$$P_{i,t}^e = x + ynq_{i,t} \quad (6)$$

or the quantity produced by firm i at time t is given by:

$$q_{i,t} = \frac{1}{yn}(P_{i,t}^e - x). \quad (7)$$

The aggregate supply of the good at time t , Q_t^s is then the sum of individual quantities supplied, i.e.

$$Q_t^s = \sum_{i=1}^n q_{i,t}. \quad (8)$$

Now that we have looked at the supply side of this market, let's now look at the demand side. The total quantity of the good demanded in the market at t , Q_t^d is defined by the following equation:

$$Q_t^d = \frac{A}{B} - \frac{1}{B}P_t. \quad (9)$$

The inverse demand curve is then given by:

$$P_t = A - BQ_t^d. \quad (10)$$

The market clearing price is the price at which aggregate supply equals aggregate demand, i.e. $Q_t^s = Q_t^d$. We then compute this market clearing price by substituting (8) into (10). The resulting market clearing price is thus given by:

$$P_t = A - B \sum_{i=1}^n q_{i,t} \quad (11)$$

The market clearing price depends on the quantities supplied, and on the intercepts and slopes of the aggregate supply and demand functions determined by parameters x , y , A , and B .

Individual quantities supplied depend on price expectations of individual firms and the shape of their cost function. Up till now, we have not said anything about how these expectations are formed. In order to figure out what the market clearing price P_t is, we have to be more specific about the price expectations. There are many possible ways in which these expectations can be modelled. They can be *homogenous* (meaning that all firms have the same expectations) or *heterogenous* (expectations differ across firms). Firms may or may not change (update) their expectations in every time period. They might also learn the correct price prediction over time.

In our analysis of expectations formation we will start by assuming that all the firms have homogenous expectations, i.e.

$$P_{i,t}^e = P_t^e \quad \text{for all } i \quad (12)$$

In addition, we will assume that this expectation is correct, i.e. that

$$P_t^e = P_t \quad (13)$$

Thus (6) becomes:

$$P_t = x + ynq_t \quad (14)$$

These expectations are called *rational expectations*. We can now compute *rational expectations* equilibrium price and quantity produced.

In the rational expectations equilibrium:

1. firms maximize their profits,
2. aggregate supply equals aggregate demand
3. and the firms' price expectations equal the actual price.

We set (refmktprice) equal to (14) and since $q_{i,t} = q_t$ for all i , we obtain the following expression

$$x + yq_t = A - Bnq_t \quad (15)$$

which we solve for q_t . Then, q_t is equal to the rational expectations quantity, q^* :

$$q_t = q^* = \frac{A - x}{(B + y)n}. \quad (16)$$

We would then use q^* to compute P^* . Note that if there are no changes in the model's parameter values the rational expectations values of the quantity and the price level are constant over time.

How plausible are the rational expectations? In other words, in real world economies, do firms (economic agents) always make accurate predictions. On one hand, we know that firms make errors in their forecasts of the relevant economic variables. On the other hand, if they made systematic errors over long periods of time, firms would go out of business. If they don't try to correct their predictions and update them in the direction of the actual price movements, they will make losses and will eventually go bankrupt.

The main criticism of the rational expectations hypothesis is that economic agents do not know the underlying variables of the economy, and that they do not have computational ability sufficient to calculate equilibrium prices and quantities.

But, agents act in their own best interest. Since their production (or consumption) decision depend on their forecasts, mistakes in forecasts have direct effect on agents' welfare (utility or profits). So, agents will try to make these mistakes smaller and, eventually, they will learn the correct price prediction. Thus, it is plausible to assume rational expectations if we think of situations which agents have been exposed to over longer periods of time (so they had time to adjust their expectations and make accurate predictions).

The question is how to model behavior of agents in the environments with which they are faced for the first time.

2 Application of Learning Algorithms

2.1 Cobweb Expectations

The assumption of *cobweb expectations* is that agents expect a price at t to be equal to the price at $t - 1$ (Ezekiel (1938)),

$$P_t^e = P_{t-1}. \quad (17)$$

A firm bases plans for future production on the assumption that present prices will continue, and that its own production will not affect the market. Thus, a firm i ($i = 1 \dots n$) decides to produce quantity $q_{i,t} = (P_{t-1} - x)/yn$ (from equation for optimal quantity) for sale at time period t .

The time needed for production requires at least one full period before production can be changed, once the plans are made. The price is set by available supply. Since all firms have identical beliefs, using equation (11), the market price P_t is given by:

$$P_t = \left(A + \frac{x}{yn}\right) - \frac{B}{yn}P_{t-1} \quad (18)$$

Obviously, in this case, P_{t-1} does not have to be equal to P_t . The market price will converge to the equilibrium price for cobweb stable case, i.e. only if $B/y < 1$ (Nerlove (1958)). For the cobweb unstable case ($B/y > 1$) the price sequence diverges away from the rational expectations equilibrium.

1. $\frac{B}{y} < 1$ - supply steeper than demand - convergence
2. $\frac{B}{y} > 1$ - demand steeper than supply - divergence

If the rational expectations equilibrium point is reached, then $P_{t-1} = P_t = P^*$ for all t .

Rational expectations assumption is a good approximation for the economic environments in which agents had time to learn and adjust. If there is a change in the environment, then we have to worry about how economic agents adapt, how they learn about new conditions. For example, the instability of some markets has been attributed to these sorts of ‘cobweb expectations’ like effects.

Let us now look at what happens with the price level over time in the cobweb unstable case. Assume that $P_0 = 1.14$ and let’s look at the first four periods of the market.

2.2 Sample average of past prices

Average of the prices from two previous periods is given by:

$$P_t^e = \frac{P_{t-2} + P_{t-1}}{2} \quad (19)$$

2.3 Sample average of all past prices

If price expectation is given as a *sample average of past prices* (Carlson (1969)):

$$P_t^e = \frac{1}{t} \sum_{s=0}^{t-1} P_s \quad (19)$$

The price sequence converges to the equilibrium value for both cobweb stable ($B/y < 1$) and cobweb unstable case ($B/y > 1$).

2.4 Least Squares

In a model in which agents use *least squares* to update their price estimate, expectaton of price P_t is given by:

$$P_t^e = \beta_t P_{t-1} \quad (20)$$

where

$$\beta_t = \frac{\sum_{s=0}^{t-1} P_s P_{s-1}}{\sum_{s=0}^{t-1} P_{s-1}^2}, \quad (22)$$

given initial prices P_{-1} and P_0 . In each time period t , agents run regression on past values of prices to obtain an estimate of the coefficient β_t . The price sequence converges for the cobweb stable case and diverges away for the cobweb unstable case.

3 Single-population Genetic Algorithm

Firms' decision rules are represented by binary strings. A population of binary strings A_t represents a collection of firms' decision rules at time period t . A firm i , $i = 1 \dots n$, makes a decision about its production for time t using a binary string, $A_{i,t} \in A_t$, of finite length l , written over $\{0, 1\}$ alphabet. A decoded and normalized value of a binary string i gives the value of the quantity $q_{i,t}$ produced by a firm i at time period t .

For a string i of length l the decoding works in the following way:

$$x_{i,t} = \sum_{k=1}^l a_{i,t}^k 2^{k-1}$$

where $a_{i,t}^k$ is the value (0, 1) taken at the k^{th} position in the string.

After a string is decoded, integer $x_{i,t}$ is normalized in order to obtain a real number value $q_{i,t}$, quantity that firm i decides to produce and offer for sale at time period t :

$$q_{i,t} = \frac{x_{i,t}}{\bar{K}}$$

where \bar{K} is a coefficient chosen to normalize the value of $x_{i,t}$. The coefficient is chosen in such a way to result in quantities in the range $[0, q_{max}]$ where q_{max} is the maximum quantity that a firm can produce.

Fitness of a string i at time period t , $\mu_{i,t}$, is determined by the value of firm's profit earned at the end of time t :

$$\mu_{i,t} = \Pi_{i,t} = P_t q_{i,t} - C_{i,t}.$$

Rules that belong to the firms that earned higher profits receive higher fitness value. Notice that a firm's profit and a rule's fitness value depends on firms'

own quantity decisions and on the price that clears the market. This price in turn depends on the quantity decisions made by all the firms in the market. Thus, each firm's profit depends on both its own decision rule and on the decision rules of all other firms.

We will discuss two versions of the GA that are implemented in order to update firms' decision rules. The first one, the basic GA consist includes application of reproduction, crossover and mutation. The second one, the enhanced GA includes, in addition, the election operator.

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

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

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. A 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.

This type of reproduction operator is called *roulette wheel* or *proportionate selection*. There are several other ways in which reproduction can be performed. The objective is the same. It is to create a population of n copies of the strings such that, on average, the strings with higher fitness values receive more copies. Over time, strings that perform well will take larger and larger fraction of the entire population, while strings whose performance is poor relative to the other members of the population receive smaller number of copies or none and eventually disappear from the population.

Another commonly used reproduction operator is *tournament selection*. It works in the following way. Two strings are selected randomly. The fitness values of the two selected strings are compared and the copy of the one with the higher fitness value is made and placed into the population of copies. Strings that participated in the tournament are placed back into the original

population and thus each one can be selected for the tournament again. This procedure is repeated n times in order to obtain n copies of strings.

When roulette wheel reproduction is used, there is a possibility that a good performing string overtakes the population early on into a simulation resulting in a premature loss of the population diversity. Tournament selection maintains population diversity for longer periods of time. Thus the way in which the reproduction operator is implemented can affect the dynamics of a simulation. We will discuss a number of other reproduction schemes through applications in other models.

Crossover exchanges parts of pairs of randomly selected strings. It operates in two stages. In first, two strings are selected from the mating pool at random. Then in the second stage, a number k is selected, again, randomly from $(1 \dots l - 1)$ 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 even integer) pairs are selected and the crossover takes place on each pair with probability $pcross$. An example of the crossover between two strings for $l = 8$ and $k = 4$ is given below:

$$\left\{ \begin{array}{l} 1 \ 0 \ 1 \ 0 \ | \ 1 \ 1 \ 1 \ 1 \\ 1 \ 1 \ 0 \ 1 \ | \ 0 \ 0 \ 1 \ 0 \end{array} \right.$$

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

$$\left\{ \begin{array}{l} 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \\ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \end{array} \right.$$

Mutation is the process of random change of the value of a position within a string. Each position has a small probability, $pmut$, 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 or equal to the probability of mutation, $pmut$, 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 $pmut$, the value of the position is left unchanged. This is repeated ln times, i.e. for every position in every string.

The *election* operator tests newly generated offsprings before they are permitted to become members of a new population. The string value of each new offspring, obtained at time t , is decoded in order to obtain the value of the production that an offspring would represent were it used as an actual decision rule. Profit that results from such production decision is computed

using the price that prevailed in the market at time $t - 1$ and it represents the offspring's *potential* fitness value. This potential fitness of an offspring is compared to the *actual* fitness values of its parents, i.e. the fitness values of the two parent strings that were evaluated at the end of period $t - 1$. (Parents are pairs of strings that are taken from the mating pool for the crossover application.)

Then, two parents and two offspring are ranked according to their fitness value, from the highest to the lowest, and the top two are taken as members of the new population of decision rules. In case of a tie between a parent and an offspring, an offspring becomes a member of the new population.

The exact algorithmic procedure is the following: Two strings that are selected for crossover are recorded as a parent pair. (This is done regardless of whether the crossover actually takes place.) The two strings that result from the crossover are recorded as two offspring. If the crossover does not take place, than two strings which are just the parent copies are recorded as offspring. The two offspring then undergo mutation. Once mutation takes place, potential fitness values of the resulting two offspring is calculated. The parents and the offspring are ranked according to their fitness and the first two become the members of the new population of generation $t + 1$.

In case of the basic genetic algorithm, a population of chromosomes that will represent decision rules of firms at time period $t + 1$ is obtained in the following way: First, the application of the reproduction operator yields a population of n copies. Then, the crossover and mutation operators are applied to this population to yield a new population of decision rules that will determine firms' production at $t + 1$. If the enhanced genetic algorithm is used, the application of reproduction, crossover and mutation is followed by the application of election operator.

After the members of the new population are determined, a quantity that will be produced and offered for sale at time $t + 1$ is computed for each firm. Next, individual quantities are summed up and the market price of period $t + 1$, P_{t+1} , is computed.

Costs associated with produced quantities are computed for each firm (equation (1)). A profit for firm i (a fitness value of a string i), $i = 1 \dots n$, is calculated using the price of generation t , P_t (equation 2).

The above described steps are applied iteratively for T generations. Initial population at time period 0 is randomly generated.

The whole process may be given the following economic interpretation. Reproduction works like the imitation of successful rivals where the produc-

tion decision rules of those firms whose beliefs are given by well-performing strings are copied by others, by virtue of them earning higher profits in the market. Strings with lower fitness values, which means worse production decisions and lower profits, get less copies (or none) in the next generation, as investors or financial intermediaries are not willing to allocate investment funds into an unprofitable production. Crossover and mutation are used to generate new ideas (beliefs) on how much to produce and offer for sale, recombining existing beliefs and generating new ones with crossover and mutation.

If election operator is included, the above interpretation may be modified in the following way. In each period firms generate new production decisions using genetic operators. They compare the fitnesses of these new potential proposals to the old set, under the market conditions observed in the past. Only new ideas that appear promising on such grounds are actually implemented (whereas the generation of new ideas is random, their implementation is not).

It is worth noting that with the genetic algorithm learning, individual firms do not use first-order conditions for decision making, as they do in the case of the other learning algorithms previously studied in the context of the cobweb model. They do not equate marginal cost to the marginal revenue (the expected price) and need not calculate either in order to decide how much they are going to produce in the following period. Still, by the time the algorithm converges, firms have learned not only how to predict the correct rational expectations equilibrium price, but also how to make production decisions that will maximize their profits.

3.1 Results of Simulations

Simulations were conducted for seven different sets of the cobweb model parameter values (see table 1) using both basic and enhanced algorithm. GA populations consisted of thirty strings and a string length was set to thirty bits.¹ Eight different sets of crossover and mutation rates that were used (see Arifovic, 1994). Each simulation was conducted for two hundred

¹Some applications in artificial intelligence that studied the behavior of the algorithm in the optimization of various types of functions (De Jong (1975, 1980)) showed that the best performance of the algorithm was achieved using the population of thirty chromosomes with the length of a chromosome equal to thirty bits.

periods.²

Simulations of basic GA did not result in the convergence to the rational expectations equilibrium values. In every simulation, individual quantities and prices fluctuated for its entire duration. First set of genetic operator rates ($pcross = 0.6$ and $pmut = 0.0033$) had consistently the smallest magnitude of fluctuations. However, after initial adjustment, prices and quantities continue fluctuating around the rational expectations values.

On the other hand, simulations which were conducted by using the enhanced genetic algorithm resulted in the convergence of the algorithm to rational expectations equilibrium values for all sets of cobweb model parameter values, which include both stable and unstable cases. All the strings in a GA population of each simulation became identical, i.e. the beliefs of all firms about how much to produce and offer for sale converge to the same value which is equal to the optimal quantities when the market price is known.

Second set of genetic operator rates ($pcross = 0.6$ and $pmut = 0.033$) resulted in the fastest convergence for all of the cobweb model parameter values that were examined. Use of other sets of genetic operators rates resulted in greater fluctuations that lasted longer, relative to the results obtained when the second set of rates was used. Nevertheless, convergence is reached in less than thirty periods for all sets of crossover and mutation rates.

3.2 Discussion of the election operator effect

Necessary condition for the algorithm's convergence to the equilibrium values is that no rule (string) in the GA population deviates from the quantity that maximizes profit at a market clearing price. This also implies that the variance of the population of rules is equal to zero. In equilibrium, the population of identical rules decode to the quantity that equals the value which maximizes profits at a correctly perceived equilibrium price.

While reproduction operator works towards a reduction in the variance of the rules, mutation works towards maintaining a degree of diversity. In simulations with basic GA, the extent to which the diversity brought in by

²Multiple runs (using different seed values for the initialization of the random number generator) of simulations were conducted for each combination of the cobweb model parameter values and set of genetic operators rates to ensure that the results are robust to different sequences of random numbers. The same procedure was used for the simulations of multiple-population GA whose results are reported in the following section. Results of individual runs, not averages over multiple runs, were examined.

mutation offsets the effects of reproduction operator depends on the rate of mutation and on the complexity of learning environment. In any event, continuing effects of mutation will maintain a variance of rules greater than zero as long as a simulation is conducted, resulting in rules which decode to quantities different from the quantities that maximize profits at given prices.

On the other hand, application of election operator results in the reduction of rules' deviation from the quantity that maximizes profit at market clearing price and in the reduction of the population variance over the course of a simulation. Finally, election operator brings the variance of the population rules to zero as the algorithm converges to equilibrium values.

Once the algorithm converges to the equilibrium values, all strings become identical and thus, all are decoded into the same, equilibrium value of production q^* . Mutation will continue to generate new, different strings at a rate that is exogenously given, but none of these strings will be accepted into the population. Any string that decodes into the quantity q such that $q \neq q^*$ will have lower fitness value, evaluated at the equilibrium price P^* . Thus, election operator enables an endogenous shut-off of mutation resulting in its effective rate equal to zero in equilibrium.

This does not, however, mean that adjustment of the algorithm to the new equilibrium values will be prevented if the underlying parameters of the economy change (for example, a change in the production function or a change in the demand schedule).

Suppose that for a given set of the cobweb model parameters, x , y , A and B , the genetic algorithm converges to the equilibrium with the equilibrium price P^* and equilibrium individual quantity q^* . All strings are identical and decode to this quantity. Then, at time period $T + 1$, a parameter of the demand schedule changes from the value A to the value A' . This will also result in the change of equilibrium values of price and individual quantity to the new values of $P^{*'}$ and $q^{*'}$.

When genetic operators are applied to yield a population of production decision rules at $T + 1$, reproduction and crossover will have no effect on the population of identical strings. Mutation may generate some new strings with quantities different from q^* , but they will not be accepted into the population of period $T + 1$ since they are evaluated at the price $P_T = P^*$. Thus, at the period when the change occurs, $T + 1$, all strings that determine market supply will still decode to quantity q^* . However, these strategies will

no longer result in price P^* . The price at $T + 1$ will be given by:

$$P_{T+1} = A' - Bnq^* \neq P^* .$$

At time $T + 2$, after the application of reproduction, crossover and mutation, strings are evaluated at price P_{T+1} and any string that decodes to quantity $q_i \neq q^*$ which has a higher fitness value (which is now possible since q^* is no longer optimal quantity) will be accepted into the population to give decision rules for production at $T + 2$. If such a string (or strings) is not created at $T + 2$, it is created in one of the subsequent periods. In this way, diversity is brought into the population and reproduction, crossover and mutation will work their way through towards the adjustment to the new equilibrium values. (For a detailed discussion of the behavior of the genetic algorithm with election operator in the environments in which a parameter of the economic model changes see Arifovic (1991)). In computer simulations, the adjustment of the genetic algorithm populations of identical strings to new equilibrium values when the change in parameter occurs does not last longer than the adjustment of initial the genetic algorithm populations of strings that are randomly generated. ³

It should be noted that the information about economic environment available to cobweb firms that use the enhanced genetic algorithm is the same as the information available to firms that use the basic genetic algorithm. The application of election operator requires only the use of the previous period market price to test newly generated strings. Market price is also observed and used by basic genetic algorithm firms to compute profits they earn, i.e. the fitnesses of the decision rules they used.

4 Multiple-population Genetic Algorithm

An alternative way to represent economic agents' learning by means of GA is to endow every individual agent with a whole population of strings. Then

³Another possible way to deal with the problem of the mutation rate effects which prevent algorithm's convergence is to use exponentially decaying mutation rate which is used in simulated annealing and in some genetic algorithm applications. The rate at which mutation decreases as the number of simulation periods increases is given exogenously. While the inclusion of the election operator does not impair the system's capability for new adjustment in case of change in underlying parameters of the model, this ability does get reduced if mutation rate is exogenously decreased over time.

we can think of these strings in such a population as being agent's mutually competing ideas about what his behavior in a given environment should be. In each time period, only one string is selected as a string that determines agent's behavior. The probability of choosing a particular string is proportional to its performance under predefined conditions. Although an agent chooses only one string from among the whole collection, he still evaluates *all* of his alternative ideas, upon obtaining the information about variables whose values he did not know at the time of his decision making. This way, the agent gets the information as to what the value of his objective function would have been had he used a particular string from the population of competing decision variable values. The agent uses this information in the process of updating his beliefs (updating is performed using genetic operators) to assign higher probability of reproduction to those strings that yielded higher values in the performance evaluation.

Multiple-population GA is richer compared to the single-population GA in a sense that we can now think of firms as having a number of different ideas about their possible production quantities, which they evaluate and choose one among them, using some adopted selection rule. On the other hand, it still does not require any excessive computational capacity on the part of agents. Also, as in the case of the single-population GA, there is no assumption of profit maximization on the part of firms preceding their production decision. At the same time, it is not *simplistic* in a way of the single-population GA, where each agent has only one strategy at his disposal.

Single-population GA uses GA as a model of *decentralized* learning in economies in which individual agents do not learn at all (since they follow fixed rules), or simply imitate their successful neighbors, but the economy as a whole is, in effect, employing a sophisticated algorithm for learning. Multiple-population GA design uses GA as a model of *learning by individual agents*.

4.1 Description of the Model

There are m firms in a competitive market for a single good. As with the single-population GA model, firms do not know the price of the good that will clear the market in the next period and they have to make production decision before observing this price. Each firm has a collection A_t^j of n possible decision rules which is given as a population of binary strings. String i , member of j^{th} firm's collection at generation t , is denoted by $A_{i,t}^j$. Decoded

value of each string represents a possible value of a next period production, $q_{i,t}^j$. Decoding and normalization of a binary number given by each string is performed in the same way as in the single-population GA. Then, *competition* fitness is calculated for each string. Competition fitness of a string i ($i = 1 \dots n$) is equal to the profit at the price that prevailed at generation $t - 1$ and the quantity represented by string i .

Every string i in a firm j 's population of strings has a chance to be selected as a string that will supply the value representing the quantity of firm j 's next period production. A probability that a particular string is chosen as the firm j 's decision rule for production at t is proportional to its competition fitness and is given by:

$$\pi_{i,t}^j = \frac{\mu_{i,t}^{j,c}(P_{t-1})}{\sum_{i=1}^n \mu_{i,t}^{j,c}(P_{t-1})}$$

where $\pi_{i,t}^j$ is a probability that a string i , a member of firm j 's collection of possible decision rules at generation t , is chosen and $\mu_{i,t}^{j,c}$ is a competition fitness of a string i evaluated at the price that prevailed in the market at generation $t - 1$. Selected string becomes firm j 's *realized* decision rule that gives quantity $q_t^{r,j}$, as the quantity to be produced in time period t .

Information that firms have in this setup is the same as in the single-population framework. They know only the previous period price and their own profits earned. They do not know the quantities supplied by other firms. The assumption of the original cobweb model that a firm believes its own quantity is not going to affect the market is maintained here as well. It is also the assumption of all the adaptive algorithms described in the second section

The market clearing price is given by:

$$P_t = A - B \sum_{j=1}^m q_t^{r,j}. \quad (21)$$

Firm j 's costs of producing a quantity $q_t^{r,j}$, represented by a chosen string, at generation t are given by:

$$C_t^j = xq_t^{r,j} + \frac{1}{2}y(q_t^{r,j})^2 \quad (22)$$

and its profit at t is given by:

$$\Pi_t^j = P_t q_t^{r,j} - C_t^j(q_t^{r,j}). \quad (23)$$

Price P_t is used to determine *reproduction* fitness values of all the strings in all m populations. This fitness is equal to the profit at the price that cleared the market at t and at the quantity represented by a particular string:

$$\mu_{i,t}^j = \Pi_{i,t}^j = P_t q_{i,t}^j - C_{i,t}^j \quad (24)$$

where $\mu_{i,t}^j$ is a reproduction fitness value of a string i at t in a population that belongs to j^{th} firm, $\Pi_{i,t}^j$ is a profit that a firm j would have earned at price P_t had it used string i .

Note the difference between the competition and the reproduction fitness.

It should be emphasized that there is a difference in the meaning of the fitness value between the single-population GA model and the multiple-population GA model. In case of the former, each firm has only one string as its decision rule and string's fitness value is given by the actual profit that a firm realizes in the market. In the latter case, after a firm observes the price at generation t and before it makes a production decision for $t + 1$, it evaluates its collection of possible decision rules, assigning each member of the collection the fitness equal to a profit at price P_t and the quantity given by that string. These fitness values thus correspond to hypothetical profits.

After the evaluation of their fitness takes place, operators of the basic GA, reproduction, crossover and mutation are applied within each firm's collection of strings to yield the new sets of possible decision rules for production at $t + 1$.

Decoded and normalized values of members of a collection A_{t+1}^j give quantities $q_{i,t+1}^j$ ($i = 1 \dots n$), which are possible quantities for production at $t + 1$. In order to reach a decision about production at $t + 1$, each firm j computes competition fitnesses of newly generated strings in a collection, using the price P_t and chooses a single string which will give the quantity to be actually produced.

The process is repeated for T generations. Initially, all collections of strings are randomly generated.

In the enhanced version of the multiple-population GA, election operator is added. Before entering the competition for a string that determines next period's production, strings that are generated at period $t + 1$ as the result of application of genetic operators have to pass a *qualifying* test given by the election operator.

Within each collection of strings, the election operator is applied in the same manner as was described in case of the single-population GA.

4.2 Results of Simulations

Simulations were conducted for the number of firms $m = 2$, $m = 3$, $m = 5$ and $m = 6$ and for the parameters values of the cobweb model reported in table 1 (the same that were used for the single-population GA application). For each of these specifications, both basic and enhanced GA were applied, using the sets of values of genetic operators' rates given in table 2.

Results of simulations show that the enhanced GA converged to rational expectations equilibrium values for all sets of cobweb model parameter values, including stable and unstable case, and for all sets of genetic operators. At the same time, basic GA kept oscillating until the end of every simulation. These oscillations are the result of the effects of mutation which were discussed in the previous section. No significant decrease in the magnitude of oscillations was observed when simulations were conducted with a low rate of mutation (0.0033) and for a large number of periods (10,000).

Although only a single string determines a firm's next period production, by the time enhanced GA achieves convergence to rational expectations equilibrium, the entire firm's population of strings converges to the same value. By contrast, at the end of each simulation, there is still difference between the strings of a population that belongs to a firm which uses the basic GA.

5 Comparison With Experimental Results

Wellford conducted the total of twelve experiments in which both stable and unstable cobweb case were simulated. Each experiment had five participants and lasted for thirty periods.

Sellers in the market for a single good had to make decisions on how much to offer for sale in the next period, which is equivalent to the decision making of agents in the GA model. Furthermore, the market clearing scheme is the same in both, i.e., exogenously given demand schedule and total market supply (determined as the sum of individual supplies) give market clearing price of the current time period.

Experimental design features made distinction between stable and unstable cobweb case, between treatment of per period reward for estimation versus treatment in which no per period estimation reward was provided and between experiments with complete and incomplete information (i.e. whether

or not subjects were provided with the Cournot required complete information on the deterministic demand curve, total quantity per period and other subject's quantities in the previous period). Combinations of these different design features gave the total of six different designs.

Constraint on individual capacity was at 20 units. Selection of unstable parameters was such that the theoretical divergence does not explode beyond the market constraints price and quantity must be nonnegative and quantity must not exceed market capacity) within 30 periods of the experiment.

Since the price paths of unstable cases exhibited greater fluctuations than those of stable ones, Wellford tested the hypothesis that the price variance across all periods was the same for both stable and unstable treatments. The hypothesis was rejected (at 5% significance level) in favor of the alternative hypothesis (that the price variance of the unstable treatments exceeds that of the stable treatments.)

Three aspects of the Wellford's experimental data, namely the absence of divergent patterns in cobweb unstable case, fluctuations around cobweb model equilibrium values and the greater price variance of the unstable case will be used for the evaluation of the performance of cobweb expectations, sample average of past prices, least squares and GA.

Results of the Wellford's experiments conducted under conditions of incomplete information were used for the comparison. In the experiments with incomplete information, sellers received the information only about the last period price and about their own earned profits, but no information about the total quantity sold or about quantity choices made by other market participants. This corresponds to the GA environment in which strings do not obtain any other information except for their fitness (profit) and the price of the last period. That is also the only information available to agents who adapt by using cobweb expectations, sample average of past prices or least squares.

Fluctuations of prices and quantities around their rational expectations values that characterize models in which agents form cobweb expectations are observed in Wellford's experiments as well, but as was already discussed, for the parameter values of the unstable case, Wellford's experiments did not follow the divergent price path predicted by the model in which agents have cobweb expectations. Figure 6 exhibits the pattern of individual quantity for the cobweb stable case.

Adaptive scheme in which firms make their price forecast by taking the sample average of past prices converges for both stable and unstable case,

but since its convergence is smooth it does not exhibit resemblance to experimental data which is characterized by fluctuations around equilibrium values. Moreover, the price variance of the stable case is not greater than the price variance of the unstable case. The behavior of this algorithm in computer simulation for the Wellford's unstable set of parameter values is given in figure 7.

For the parameter values of the cobweb stable model, least squares algorithm exhibits price and quantity fluctuations that die out as the algorithm converges to rational expectations values. On the other hand, least squares algorithm does not converge in computer simulations for the unstable cobweb case. Price and quantity fluctuations become larger and larger as the algorithm diverges away from the rational expectations value of price. Figure 8 shows the pattern of the individual quantity for 30 periods, for the parameters of Wellford's unstable case.

The assumption that each agent believes all other agents will behave in the same way in next period as they did in the last one underlies both election operator and cobweb beliefs. The difference is that cobweb agents' decision is profit maximizing decision (a single value). In GA, on the other hand, any quantity that yields higher profit, given the last period price, than a quantity that was offered last period will be accepted as a new strategy.

When comparing the processes generated by the cobweb experiments and single-population GA, one cannot interpret each single decision rule of GA firms as being a decision rule of a market participant in the cobweb experiments. In a single-population GA simulation, there are thirty suppliers, while there are only five human subjects participating in Wellford's experiments. On the other hand, single-population GA simulation with only five strings would not be feasible, since diversity of an initial population would not be sufficient to create the process comparable to the experimental data. In general, dynamics of such a process would hardly be of any interest, since the crossover would not have a base diversified enough to search usefully through a state space. Besides, the application of reproduction operator would result in a premature convergence of such a small population and most likely to a non-equilibrium position.

Multiple-population GA gives the opportunity of conducting simulations in which the number of GA decision makers equals the number of participants in Wellford's cobweb model experiments, yet at the same time preserves the sufficient diversity of GA populations. In this case, GA was set up with five populations since there were five experimental participants. Each popula-

tion represented a collection of possible decision rules for a single market participant. Thus, it was possible to relate quantities that GA firms decide to sell to the individual quantity choices observed in the experiments. (Each population consisted of thirty strings.)

Both single-population and multiple-population enhanced GA converge for both stable and unstable cobweb case. Prior to convergence, GA exhibits fluctuations around the equilibrium values.

GA price patterns also show that variance of unstable cases is greater than that of stable ones.

5.1 Variance of the stable and unstable cobweb case

If one wants to test the same hypothesis that Wellford tested on experimental data (i.e. that the price variance of the unstable treatments is equal to that of the stable treatments, on data generated by GA) two different approaches can be taken.

One approach is to assume that agents use the same set of values of crossover and mutation rate in all experiments, and for both stable and unstable cobweb case. With this assumption, data used for the hypothesis testing is generated by a single set of genetic operators values. The alternative approach is to assume that agents use different sets of values in different experiments. Besides the fact that the price variance observed in cobweb stable experiments was overall lower than the price variance in the unstable cases, there were also substantial differences in price variance across experiments of the same cobweb type. In some of the cobweb stable experiments, the price remained close to the rational expectations equilibrium price throughout the whole experiment with the resulting low price variance. In others, it exhibited substantial fluctuations. This was observed in the cobweb unstable experiments as well. On the other hand, in GA simulations, different sets of genetic operators values generate different values of price variance. Higher rates of crossover and mutation generate, on average, higher price variance, while lower genetic operators rates result in lower price variance. Thus, a set of observations used for the computation of variance in price of each cobweb type should consist of data generated with the whole range of values of crossover and mutation rates.

For the first approach, data was generated conducting 20 iterations for each set of genetic operators' values. Each iteration lasted for 30 generations and each was initialized with the different seed for the random number

generator. All sets of values were implemented for both stable and unstable cobweb case (20 iterations for each). The same sequence of seed numbers was used for each set of values. The null hypothesis that the variance in price across all periods for unstable cobweb type for a single set of genetic operators rates is equal to that of stable cobweb type for the same set of crossover and mutation rates was tested for each of the above specified set of values, at 5% significance level. It was rejected in favor of the hypothesis that the variance in price is greater in unstable than in the stable case for each set of values. Data for the second approach to hypothesis testing was generated conducting one iteration for each set of the genetic operators' values. Each iteration lasted for 30 generations. The whole simulation consisted of 20 iterations with 20 different sets of values. One simulation was performed for the cobweb stable case and one for the unstable case, using the same sequence of genetic operators' rates for both cases. Data generated in a single simulation for each cobweb type represented the basis for the calculation of respective price variances. Crossover rates varied from 0.6 to 0.9 and were combined with mutation rates values of 0.0033 and 0.033. Variance tests were performed on data generated using both single-population and multiple-population GA. For each of these tests, the hypothesis that the price variances of the stable and unstable case are the same was rejected in favor of the hypothesis that the variance of the cobweb unstable is greater than that of the cobweb stable case.

6 Extensions of the Basic Model

6.1 A model with Fixed Costs and Entry/Exit Decisions

Dawid and Kopel (1997) use a version of the cobweb model with fixed costs where for certain relative size of the fixed cost no homogenous rational expectations equilibrium exists (where homogenous implies that all firms are making identical production decisions). They use this model to illustrate how the outcomes of genetic algorithm simulations can depend on details of a particular coding scheme that is implemented. They also develop a stability criterion which can be used in order to assess the robustness of the simulations results. In their version of the cobweb model, the costs are given by: $C_{i,t} = x + yq_{i,t}^2$ if $q_{i,t} > 0$ and equal to 0 if $q_{i,t} = 0$. The market clearing

price is determined using (11).

The restriction on the size of fixed costs, x , that insures that there are at least some combinations of the expected price and quantity for which the expected profits are positive is given by: $z < A^2/4y$. The unique, homogenous expectations equilibrium (equivalent to the one described in the previous section) exists for

$$z < \frac{A^2 y}{(2y + Bn)^2}, \quad (25)$$

in which case $p_t^e > 2\sqrt{zy}$, and optimal quantity for each firm i is $\hat{q}_{i,t} = p_{i,t}^e/2y$. Notice that given the expected price and optimal quantity, the expected profits, for this relative size of the fixed cost, are positive.

In the rational expectations equilibrium $P_t^e = P_t = P^*$ and the unique value of q^* is given by:

$$q^* = \frac{A}{2y + Bn}$$

However, if (25) holds with equality, the expected price P_t^e is equal to $2\sqrt{xy}$. The expected profits are equal to 0, and there are two optimal values for the quantity, $\hat{q} = \sqrt{x/y}$ and $\hat{q} = 0$. In this case, there is no rational expectations equilibrium with homogenous decisions. In fact there is only the equilibrium with heterogenous decisions. In equilibrium, the fraction f of firms is producing positive quantity, and the remaining fraction $1 - f$ is staying out of the market, i.e. producing 0 quantity. The size of the fraction f is endogeneously determined and depends on the parameter values. Set $P_t = P_t^e$ and then compute f :

$$2\sqrt{xy} = A - Bynf\sqrt{\frac{x}{y}} \quad (26)$$

Their implementation of the genetic algorithm is the same as the one described in the single-population algorithm, i.e. each firm's production decision is represented by a binary string. The simple version of the algorithm with three operators, reproduction, crossover, and mutation was used. In order to avoid negative fitness values, Dawid and Kopel scaled the profit by adding a positive constant equal to $z + y(\frac{A}{Bn})^2$.

6.1.1 Results

For the set of the parameter values for which the unique, homogenous rational expectations equilibrium exists ($A = 5$, $B = 5$, $z = 0.25$, and $y = 1$) the genetic algorithm population converged to the rational expectations equilibrium quantity, $q^* = 5/7$. For the set of the parameter values for which only the heterogenous equilibrium exists ($A = 5$, $B = 5$, $z = 1$, $y = 1$), the genetic algorithm population converges to the positive optimal quantities given the observed price. The quantity is given by $q = 0.71387$ and the string k that encodes this quantity is given by: 1011011011. However, the fixed costs are so high that firms are making negative profits. Thus each individual firm would be better off by exiting the market and making, instead, zero profits. Now, if all firms decide to exit the market and set their production decisions to zero, this does not constitute an equilibrium either since, in that case, the market price would be so high that an individual firm would have an incentive to make a unilateral decision to start producing again. Notice also that in this setup a change from q^* to the production decision equal to 0 requires a change of 7 bits in the string. The probability that they all change within a single generation is very small.

Dawid and Kopel show that a uniform state in which all firms produce positive quantities, optimal given the observed price level, but resulting in negative profits, is locally stable for the dynamics of a GA with $p_{mut} = 0$ and a one-point crossover with $p_{cross} \in (0, 1]$. The stability check is based on a condition presented in Dawid (1996) which states that a uniform state e_k is locally asymptotically stable (for the expected dynamics of GA) with $p_{mut} = 0$ and one-point crossover with probability $p_{cross} \in (0, 1]$ if

$$\frac{d(j, k)}{\ell - 1} > \frac{1}{p_{cross}} \left(1 - \frac{\mu_k(e_k)}{\mu_j(e_k)} \right) \quad (27)$$

for all $j \in \Omega$, $j \neq k$, (Ω is the set of all possible states that can be represented by a string of length ℓ) where e_k is state under consideration, e_j is the state with a higher fitness value, $d(j, k)$ is the distance between the two outmost bits where j and k differ in value. If there is one $j \neq k$ such that the inequality holds the other way round, e_k is unstable.⁴

⁴For further details on how this condition is derived see Dawid, 1996. Using the theory of Markov chains, he derives the characterization of the possible long run states of the GA population and, analyzing a deterministic system of difference equations, provides conditions for local stability of of an arbitrary state.

This condition shows that a state consisting almost only of strings k will converge to the uniform state e_k if the strings receiving a higher payoff in the current state differ from k in bits positioned far apart. The number of strings j will grow in proportion to their fitness. However, as long as the ratio between $\mu_j(e_k)$ and $\mu_k(e_k)$ is not too large, there is relatively large probability that a string j will be paired with string k and thus destroyed during crossover (any crossover point between the two outmost differing bits will destroy j).

Dawid and Kopel use the above condition to determine whether the state e_k in which firms produce positive quantities, represented by string k , $k = 1011011011$ and make negative profits is locally stable with respect to the string j that prescribes zero production, $j = 0000000000$, and has higher fitness value than string k . Even though $\mu_j = 2.4$ and $\mu_k = 1.5$, it is not enough to offset the large value of the left-hand side of the inequality which is equal to 1 ($1 > 0.245$).

The implementation of a modified coding scheme where extra bit is added to the binary string that is interpreted as an entry/exit decision (if it is equal to 0, stay out of the market and set $q_{i,t} = 0$). Notice now that the decision to switch between q^* and 0 requires a change of a single bit only, the entry/exit bit. Notice that with this coding scheme the above condition for the stability of the state where all firms are producing positive quantities is no longer satisfied ($1/9 < 0.245$).

Separation of the decision to enter or exit the market and a quantity determination leads to the behavior of the algorithm where a fraction of the firms f produce positive quantities and make positive profits, while the remaining $1 - f$ firms stay out of the market. (Note that both quantities result in zero profits.) The fraction corresponds to the fraction required for the existence of the heterogenous rational expectations equilibrium. The oscillations around the equilibrium values are higher than in the case of the convergence to a uniform equilibrium.

6.2 Coevolution of Different Forecasting Rules

Another interesting extension of the basic genetic algorithm application is a paper by R. Franke (1999). This paper studies the coevolution of different strategies in the cobweb type of model. In addition, there is a disturbance to the demand curve that follows AR(1) process. Firms can make their decisions using four different types of rules. The first one is the ‘fixed-production’ rule

(QF strategy) according to which quantities that firms decide to produce in the following period are represented by binary strings. The second one is the ‘adaptive expectations’ rule (AE strategy) where a binary string encodes the value of the parameter α that controls the speed of adjustment. The expectational rule is given by

$$P_t^e = P_{t-1}^e + \alpha[P_{t-1} - P_{t-1}^e]. \quad (28)$$

The third is the ‘regressions of order 1’ rule (R1 strategy). With this rule, binary strings are used to encode the length of the sample period that is used to compute the least squares estimate of the coefficient β . This coefficient is then used to form price expectations. The fourth is the ‘regressions of order 2’ rule (R2 strategy). According to this rule, least squares estimate takes only prices of every second period. A binary string encodes again the length of the sample period.

These different rules compete against each other during the genetic algorithm evolutionary process. The system’s ‘collective memory’ consists of the pools of four types of strategies from which extincted types of strategies can be re-activated. Franke uses proportional selection as the reproduction operator with the following modification. The original fitness values $\mu_{i,t}$ s are transformed in the following way:

$$\hat{\mu}_{i,t} = \max[0, \sigma(\mu_{i,t} - \bar{\mu}_t) + \delta_t] \quad (29)$$

$i = 1, \dots, n$, where $\mu_{i,t}$ fitness of a binary string i , $\bar{\mu}_t$ the mean value, δ_t the standard deviation across the population of firms, and σ is a nonnegative parameter.

The probability that a binary string i is copied is then given by: $\pi_i = \hat{\mu}_{i,t} / \sum_{j=1}^n \hat{\mu}_{j,t}$ if $\delta_t > 0$ or $1/n$ if $\delta_t = 0$. A binary string will be eliminated with certainty if it does worse than $1/\sigma$ standard deviations from the mean. If the constraint in (29) is not binding, then a binary string (strategy) whose fitness is one standard deviation above the mean has the standard deviation above the mean has a reproduction probability of $(1 + \sigma)/n$. A higher σ results in higher evolutionary pressure.

Other operators used for updating are one-point crossover, mutation and the *weak* form of the election operator. In the ‘strong’ version of this operator, Arifovic (1994, 1996), two parents and the resulting two offspring are ranked and the two with the highest fitness values are placed into a new population of rules. In the ‘weak’ version of this operator, developed in the paper, each

offspring is paired with a single, more similar, parent and the comparison of the fitness is done for each offspring/parent pair.

To understand better the effects of the strong versus weak version of the election operator put forward by R. Franke, let us first consider what the effects of crossover are. Consider two binary strings, A and B and the two offspring A' and B' which encode real numbers a , b , a' and b' respectively. Let A and A' coincide in the higher valued bit segment to the left of the crossover point (the same for B and B'). Suppose also that $a < b$. First, a' can be less than a . In this case $b' > b$. Or, a' and b' can be weighted sums of a and b . In this case (as long as A and B do not coincide in the leading bits), $a < a' < b' < b$. According to this, we can conclude that A' is the offspring closer to parent A and that B' is the offspring closer to parent B . Thus each offspring can be assigned to one of the two parents, A' to A , and B' to B .

The above analysis is relevant for the way the weak version of election operator is implemented. In this version, the fitness contest takes place pairwise, such that each offspring is only compared with the parent to which it was assigned during the crossover process. In this case, it is either A or A' and B or B' that enter as members of the new population. (Franke provides a a good discussion of why the weak election operator might be superior. However, he does not compare, through simulations, the effects of the ‘strong’ and the ‘weak’ version.)

6.2.1 Results

In a deterministic environment, without demand shocks, the algorithm converges to a Nash equilibrium in which all firms produce the same quantities, but use different rules to make these decisions. On average, 75% of firms use adaptive expectations. The QF-strategy disappears from the population with only 0.01 firms (on average) that use it.

On the other hand, the stochastic environment is characterized by the *coevolution* of strategies. During coevolution, different rules get wiped out from the population and brought back in from the ‘collective memory’. In the stochastic environment, all four types of strategies survive in the population. The largest percentage are still the AE binary strings, but the second highest percentage is represented by QF strategies. At the same time, R1- and R2 strategies take much smaller proportion of the adapted populations. During the co-evolutionary process, the total quantity produced by the GA strategies, Q_t remains close to the (moving) rational expectations equilib-

rium quantity.

Even though the co-evolution of strategies is driven by exogenous changes to the demand function, this represents an interesting examination of the competition of different ways in which expectations are formed.

6.3 Genetic Programming and Cobweb Model

Chen and Yeh (?) apply *genetic programming* to study adaptation in a simple cobweb model. Each firm makes a forecast of the next period's price using a parse tree, ($gp_{i,t} \in GP_t$) written over the *function set* and *terminal set*. The terminal set (inputs) consists of h lagged values of the price and a random floating-point constant. The function set has the following elements: $\{+, -, \times, \%, EXP, RLOG, Sin, Cos\}$. These inputs are manipulated using the functions specified by a particular tree. The result is the forecast of the next period's price that is used to determine the quantity that will be produced next period (from the expression for the optimal quantity given the price forecast). A population of trees is updated using the genetic algorithm operators, reproduction, crossover and mutation, adapted for the genetic programming parse trees. Chen and Yeh find that the evolutionary process modeled using the genetic programming gets in the neighborhood of the rational expectations equilibrium for stable as well as unstable case.

More detailed description of the genetic programming is provided in the section that deals with the use of the evolutionary algorithms for search of optimal technical trading rules.