Genetic Algorithms and The Cobweb Model

Giulio Frey

January 21, 2024

1 Introduction

The Cobweb model is a model of a market where firms are price takers (cannot directly choose the price of a good). Firms choose the quantity of a single and identical good that will be produced before observing market price. The total quantity supplied by all the firms in the market determines the price and consequently profits. This model has been traditionally solved using the Cobweb theorem. This theorem states that market prices will converge to a stable equilibrium point if the slope ratio of supply and demand curves is less than one. This implies that the demand curve is more elastic than the supply curve. Otherwise, the equilibrium is unstable

Experimental studies and rational expectations models prove the existence of a stable equilibrium point also when the condition for the Cobweb theorem is not satisfied. In a nutshell, this happens because adaptive agents will learn to choose a Nash equilibrium point and select one of the equilibria (when there is more than one).

A genetic algorithm provides the same solution as the ones proposed by rational expectations equilibrium models. This is not achieved by firms knowing how to maximize their profits, as it happens in those models. On the other hand, firms in a genetic algorithm learn and adapt to achieve profit maximization starting with very limited knowledge. This approach could better represents how real word firms chose production. We build a similar GA as the one showed in Arifovic (1992).

2 The Cobweb Model

We will cover in this section the basic functioning of the Cobweb model in order to understand how the GA works. Each firm faces the following costs and profit functions:

$$\begin{aligned} C =& xq + \frac{1}{2} yn(q)^2 \\ \Pi =& P^e q - xq - \frac{1}{2} yn(q)^2 \end{aligned} \tag{1}$$

Were x and y are cost parameters. Price that clears the market is given by the following equation:

$$P = A - B \sum_{i=1}^{n} q_i \tag{2}$$

Were i represents each individual firm and n th etotal number of firms in the market.

We can solve for the rational equilibrium model by assuming that $P = P^e$. Solving the first order condition of the profit maximization we find the equilibrium quantity and price:

$$
\begin{aligned}
q^* &= \frac{A-x}{n(B+y)} \\
P^* &= (A-B)\frac{A-x}{B+y}
\end{aligned}
\tag{3}
$$

3 Genetic Algorithm Set Up and Parametrization

In the genetic algorithm model each firms decides the quantity to produce based on its phenotype. Each firm's phenotype derives form its genotype and we chose a one to one mapping of phenotypes to genotypes. Genotypes are n-bit strings, and get decoded into phenotypes by a two step process. Firs the n-bit string is converted from binary to a decimal number. Secondly the number found is normalized by dividing it by a threshold. It follows an example with a 5-bit string and a threshold of 2.

$$
[0,1,0,1,1] \underset{Conversion}{\Longrightarrow} 11 \underset{Normalization}{\Longrightarrow} 0.6875
$$

The basic functioning of the genetic algorithm is described in the pseudocode below.

Algorithm 1 Genetic Algoritm Pseudo-code

```
Random genome initialization (duplicates are not removed)
while n ≤ n° of generations do
    Decode genotype
    Extract market prices given aggregate firms production
    Extract profits
    Select parents
    Generate children
    Repair children
end while
```

Each population is initialized randomly without removing duplicates. A loop is started to go over all generations. inside this loop each firm encodes the genotype into the phenotype, thus extracting its individual supply decision. All the supply decisions are summed and the market price is extracted for this generation. Individual firms profits are computed. Profits are used as the fitness function.

Parents are selected using tournament selection. Because we do not need full population knowledge this method can be applied also to big populations. This selection method picks a number of firms in the population (in our case 2) and select the one with the best fitness, then iterates for the whole population.

Children are generated with one point crossover. A random point in the parents string is selected. each children will inherit the first part of the genotype from the one of the parents and the second one to the other. In our case two parents will generate to children. Note that we exclude the beginning and the end of the parents string from the possible crossover points. This allows us always have children that will be different from parents. But that result is not always desirable as the best solution may be already in the population. We thus choose a percentage of the children to be identical to the parents. This is defined by the crossover rate

Children mutations happens by flipping one of the bits in the genotype. Note that we control the number of mutations that happens with the mutation rate.

After all those processes children may have deviated from the optimal quantity of production. whe thus construct a repair mechanism that compares both the parents and the children and select the best two of them. We do not have yet profits for the children thus those are computed with the prices observed by their parents. This means that if the children were more fit for their parents prices, they

	GA 1	GA 2	GA 3	GA 4	GA 5
Number of Individuals	100	10000	100	100	100
Number of Generations	10000	100	100	100	100
Number of Bits	13	13	3	13	13
Crossover Rate	0.9	0.9	0.9	0.9	0.9
Mutation Rate	0.005	0.005	0.005	0.05	0.005

Table 1: Parametrization of genetic algorithm

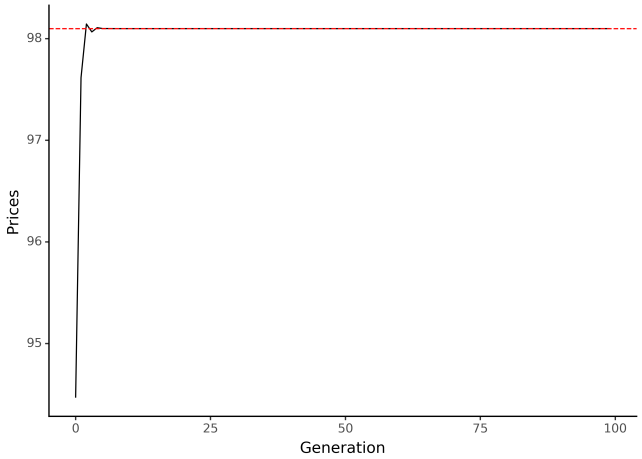


Figure 1: Prices over generations of GA 5

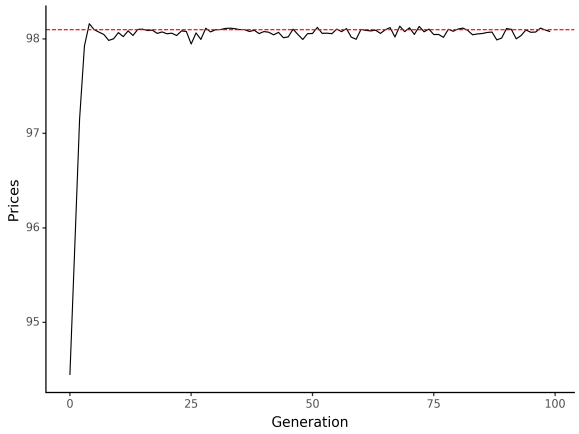
probably will be also for their generation prices. We will see the effect of this mechanism in action in the results section.

In GA 1 and GA 2 we tried increasing the number of both individuals and generations. We decided to settle on 100 as we obtain with those parameters both convergence in all the cases and quick computation times. The number of bits in our model relates to the how large is the phenotype space. We settled on 13 as this allowed to have 8192 phenotypes. This allows us to obtain quantities as precise as 10^{-3} with our chosen threshold of 5. In GA 3 we altered the number of bits but the density of the phenotypes was too low and caused cases of false convergence. In GA 4 we increased the mutation rate. This caused no problems if the repair mechanism was active but when we run the GA with no repair function no firm reached the equilibrium point. GA 5 is the parametrization that we settled on.

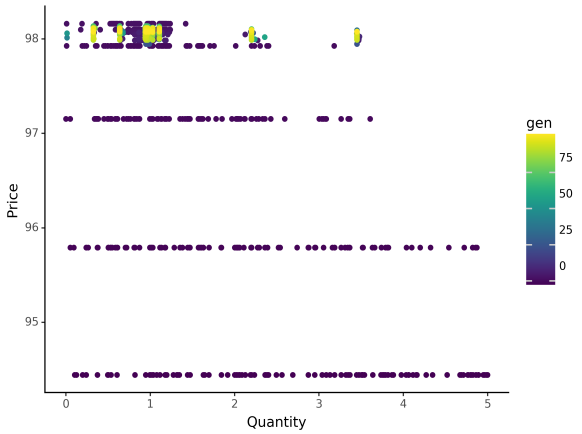
4 Results

We now move analyze results of the genetic algorithm parametrized as GA 5. Figure 1 shows how quickly our genetic algorithm converges to the equilibrium point of the rational expectations model (higlited in red)

We also first tried a version without the repair mechanism active and obtained no stable equilibrium. This is showed by Figure 2a. Plot 2b shows prices and individual firms quantity with generation based coloring each generation. We see how no generation reaches a stable equilibrium also in this plot. With no repair mechanism mutation and recombination will cause some children to decrease fitness compared to their parents, thus a stable equilibrium will never be reached in this setting. Another interesting graph is reported. We see how variation is reduced generation by generation but still no convergence point is to be found. Note that prices are identical for all firms as in the Cobweb model firms are price takers.



(a) Prices over generations of GA 5



(b) Prices and Quantity with generation based coloring

Figure 2: GA 5 with no repair mechanism

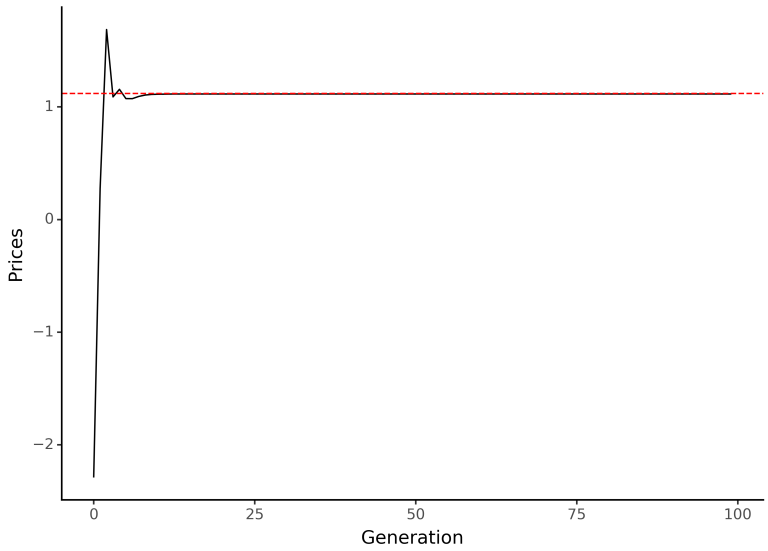


Figure 3: Prices over generations of GA 5 with unstable demand and supply

Lastly we show how there exists convergence with a genetic algorithm even with demand and supply functions that do not satisfy the Cobweb theorem. This variation is showed in Figure 3