Applied Intelligence 21, 7–24, 2004

Supply Estimation Using Coevolutionary Genetic Algorithms in the Spanish Electrical Market

ENRIQUE A. DE LA CAL MARÍN AND LUCIANO SÁNCHEZ RAMOS
Computer Science Department, University of Oviedo, Asturias, Spain

Abstract. The price of electrical energy in Spain has not been regulated by the government since 1998, but determined by the supply from the generators in a competitive market, the so-called "electrical pool". A genetic method for analyzing data from this new market is presented in this paper. The eventual objective is to determine the individual supply curves of the competitive agents. Adopting the point of view of the game theory, different genetic algorithm configurations using coevolutionary and non-coevolutionary strategies combined with scalar and multi-objective fitness are compared. The results obtained are the first step toward solving the induction of the optimal individual strategies into the Spanish electrical market from data in terms of perfect oligopolistic behavior.

Keywords: intelligent data analysis, coevolutionary genetic algorithms, evolutionary economic models

1. Introduction

The cost of production of electrical energy depends on the type of generator: one MW produced by a nuclear plant costs less than one MW produced by a thermal plant. The most economical plants are in operation most of the time while the more expensive ones connect to the electrical system only when the former cannot cover demand. Consequently, the cost of energy is higher during peak consumption and lower during the hours of less demand, for instance at night.

Many other factors intervene in the selection of the power plants that connect to the system at any given time, such as geographic location, which affects the loss of energy in the transport lines, or the precipitation rate with regard to the operation of hydraulic plants. In spite of this, the basic principle that "the cheaper power plants connect first" explains the majority of the fluctuations in the cost of energy.

The relationship between the cost of production and the selling price is not direct. Production cost determines price in a regulated market, such as the one that existed in Spain before 1998 and continues to exist in other European Community countries. This is not so in a competitive market. Before 1998, prices in Spain were fixed by a public agency that was also in charge of elaborating a list of the power plants that should connect at any given time. This list was calculated by means of numeric optimization algorithms, which minimized the global cost of the production necessary to cover domestic demand.

In the modern model, based on free competition among the different companies [1], the Market Operator (MO), a neutral agent appointed by the State to regulate competition, calculates the energy prices for every hour, starting from the supply of the generators and the demand of the consumers. The procedure by which production is planned again is based on the principle that "the cheapest power plants connect first". In this case, however, "cheapest" does not mean "low cost", but "low selling price", because each agent is free to choose the price it wants to charge for its power. It is interesting to note that the law stipulates that all power plants are to receive the same payment for each MW of energy sold, as occurred in the non-competitive model, and not the payment they asked for in their strategy. The second principle of the competitive market is "the most costly power plant connected marks the price".

This way of assigning production poses problems difficult for the planning departments of the generating companies to resolve. On the one hand, management of the power plants would be tempted to offer energy at zero cost, such that that power plant would always be selected (and paid according to the price marked by competition). However, this strategy, if applied by a considerable fraction of the companies, would cause a global decrease in prices in the short run, to the detriment of all the generators. On the other hand, the opposite temptation, to agree on prices, much more efficient in economical terms, is prohibited by law. The only legal way to maximize the profit of a company is to accurately predict demand as well as the price that competition will offer. This way, the agent can adjust its own price so that it is slightly under market price and thus be assigned the greatest possible amount of production at a high price.

Demand can be predicted using simple statistical techniques, but not supply. In reality, not only is it difficult to predict future supply, but also to know past supply: the MO guarantees that this information will remain confidential until a certain amount of time has transpired. In order to predict supply, the first step is to elaborate an intelligent data analysis tool that is capable of estimating the past supply of the agents based on the energy prices and the hours of consumption. That is the main objective of this work.

1.1. Formulation of the Problem in Terms of the Game Theory

Given that preliminary data are insufficient to carry out a statistical analysis, it is necessary to make conjectures regarding the results. This work assumes that the agents are intelligent and that the market is fair, such that the unit profits (euros/MW) are approximately the same for all the competitors.

With this hypothesis, if we know the cost of production of the agents (and we can estimate that using data prior to 1998), it is possible to simplify market operation and abstract it to a game, which can be explained as follows. Let us assume that a certain amount of energy is to be bought from several generators. None of them is capable of supplying the total amount and the amount supplied by all of them exceeds the needs.

Each player (one of the generators) gives a referee (the MO) a closed envelope with its sales strategy. It consists of a pair "quantity supplied—price demanded per unit". The referee opens the envelopes, arranges the strategies and chooses the cheapest ones until demand is covered. Each player selected is then paid for the amount it sells at the price of the most expensive strategy that was accepted. Each player receives the difference between the price paid and their unit cost, multiplied by the energy units sold.

The actual number of players is several hundred (one player per electrical power plant). To simplify calculations, we group the price-quantity pairs of all the power plants belonging to the same company into a single total quantity produced-unit price curve. In this way, we reduce several hundred strategies to four aggregate supply curves (there are four large electrical companies in Spain). The same is done with costs: each of the four participants in the simplified game will have a curve that relates the negotiated MW with their production cost. The mechanism of this new game is a bit more complex: each player gives the referee an aggregate supply curve. The referee adds up all the curves and intersects the results with a demand curve. The cross point determines the market price. Given the price and the supply curves furnished by the agents, the revenue of each player is calculated. Finally, the net profit of each player is calculated using the difference between the income received and the value of its cost curve at the point corresponding to the amount negotiated.

1.2. Genetic Formulation

In this work, we want to reconstruct the aggregate supply curves from a file that contains the price values and the amounts sold in various game repetitions. To do this, a coevolutionary genetic algorithm is used.

Briefly, a genetic algorithm works as follows: first, we define as many populations of strategies as players. To score a strategy, we will simulate a game, making this strategy compete with the best strategy from each of the other players. The strategy will receive a high score if it fulfils these two criteria:

- That the price obtained in the simulation is similar to the price in the real game.
- That the unit profits obtained by each player are similar.

Observe that genetic algorithms have been applied to solve economic problems similar to the one considered int this paper (see [2–7]) and a market model that shares some of the characteristics of this one has been related to a coevolutionary genetic programming-based model above [8]. Unfortunately, in our opinion, none of these approaches can be extended to solve the precise problem we pose here.

1.3. Summary

The remainder of this paper is arranged as follows: In Section 2, our methodology is described. In Section 3, a simple problem is solved to illustrate the use of the method proposed here. A self-criticism of the proposed methodology is made in Section 4. In Section 5, co-evolutionary, evolutionary and classical methods, applied to a semi-synthetic problem, are compared. The paper finishes with concluding remarks and future work, and with an appendix containing numerical data from experiments depicted in Section 5.

2. Proposed Methodology

The algorithm studied in this work serves to obtain the aggregate supply curves of the companies competing in the market using the historical information from the results of several previous markets.

In the introduction, it was mentioned that these aggregate supply curves provide the price at which a company is prepared to sell its energy, depending on the amount bought. Each supply curve represents a market strategy and the companies elaborate them based on their assumptions with regard to the evolution of demand and the strategies of the other competitors.

2.1. Definition of a Supply Curve

Each preliminary datum is a pair formed by two numbers: the total amount of energy produced in Spain during a certain hour and the price of energy at that hour. This pair of values corresponds to a point on the curve obtained by adding up all the supply curves furnished by the players in the market corresponding to that hour, as explained in the introduction. Unfortunately, unless we assume that an agent uses the same curve several times at different hours, the preliminary data do not contain sufficient information to reconstruct the supply curves.

An extreme case consists in assuming that the agents always use the same curve in all the games. This hypothesis is overly simplified. The behavior expected from an agent consists in its demanding higher prices when the forecasted demand is higher: this way, it maximizes its income without risking its offer being rejected. We indeed know that the supply curves that a company uses depend on specific factors. Strategic Planning Departments take into consideration the day of the week, the hour of the day, the season, the weather forecast (rain, temperature) and some other indicators before posting prices to the Market Operator. Our analysis would be very imprecise if we did not consider some of these factors. Following our own experience, three features should be considered: the hour (which is related to the amount of energy negotiated, depending on labor hours and daylight), the day of the week (the dependence between labor hours and demand changes on weekends and holidays) and the season (electrical cooling or heating, affects both previous dependencies).

Given this information, we decided to stay in an intermediate position between (a) assuming that the supply curve is always the same for each agent, and (b) assuming a different curve for every market. Since (a) is too imprecise and (b) is intractable, in this work we will allow each agent to select its curve from a restricted set of choices, depending on the values of the features mentioned before. In other words, a strategy comprises:

- a rule-based classification system, that produces a segmentation of the market points into a certain number of classes depending on hour, day of the week and type of day, and
- as many supply curves as market segments.

That is, each individual is a set of rules whose antecedents are assertions with regard to market characteristics and whose consequents are the supply curves that the player can use. We shall call these consequents “prototype strategies”.

The simplest representation of a prototype strategy is a straight line. Linear models can approximate the behavior of a competitive electrical market in the neighborhood of its equilibrium point. Unfortunately, in spite of this kind of simplification, which is valid for studying the response of the market under small changes, it is not accurate enough to estimate complete supply curves of the agents, which are highly non linear. We have decided to use piecewise linear supply curves instead (see Fig. 1). The number of their segments will be a compromise between the accuracy of the model and the amount of available data (three segments in most of the experiments in this paper.)

2.2. Genetic Representation of Individuals

We will solve this problem either with regular genetic algorithms or with coevolutionary genetic algorithms, to compare their relative performances.

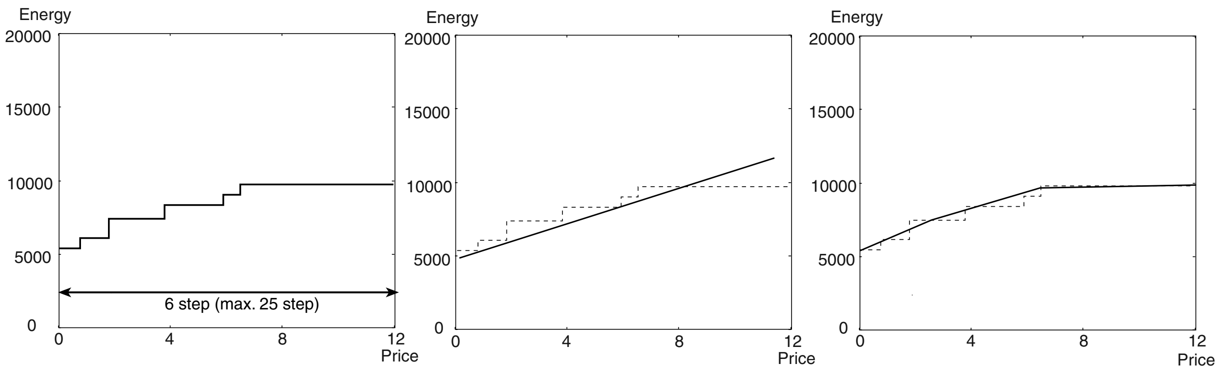

Figure 1. Actual (left), linear (center) and polygonal supply curves (right). Representation by a polygonal line is closer to reality than the linear supply and does not depend on an excessive number of parameters.

Each individual in the coevolutionary approach [9, 10] codifies a possible set of strategies (i.e., a rule-based classifier system and a set of prototype strategies) of one of the agents; we will keep as many populations of individuals as agents exist. Fitness is not assigned to an individual but to a combination of individuals extracted from all populations [11, 12]. Conversely, individuals in the regular (non coevolutionary) approach are sets that contain one set of strategies for each agent involved in the market, therefore they can be directly assigned a fitness value. The differences in representation, fitness and genetic operators between both approaches are discussed in more detail in the sections that follow.

An individual in the coevolutionary approach will be codified with a chain of numbers. This chain comprises two real numbers to define every segment in a prototype, plus a list containing the numerical parameters on which the linguistic terms in the antecedents of the classifier depend.

To clarify the codification of an individual, let us consider the example in Fig. 2. Let us suppose we have two input variables, called "hour of the day" and "type of day." The first variable can take values from 0 to

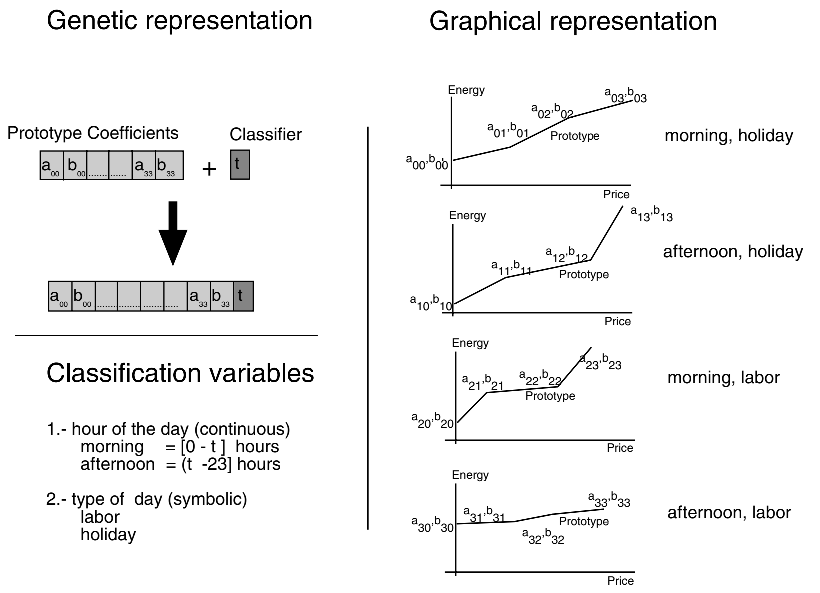

Figure 2. Polygonal supply curve comprising three segments and a classifier with two variables that segments the markets into 4 clusters; (top left) genetic representation, (bottom left) classifier variable values, (right) graphical representation.

23, and the second one can take two linguistic values, “labor” and “holiday”. The antecedents of the rules that compound the strategy must span all values of the input variables; we discretize all continuous variables into linguistic terms first, and then enumerate all possibilities. Let us call “morning” the hours before a time called t, and “afternoon” the hours after t. The complete strategy will be

```
if morning and holiday
    then prototype=(a00,b00,a01,b01,
      a02,b02,a03,b03)
if afternoon and holiday
    then prototype=(a10,b10,a11,b11,
      a12,b12,a13,b13)
if morning and labor
    then prototype=(a20,b20,a21,b21,
      a22,b22,a23,b23)
if afternoon and labor
    then prototype=(a30,b30,a31,b31,
      a32,b32,a33,b33)
```

and it can be codified by a chain of 33 numbers. Since each prototype strategy depends on eight values, we need 4×8 parameters to define all consequents and one more number to define the value of t. The antecedents need not to be codified, because they are implicit in the sorting of the rules.

In general, a strategy depending on n input variables, taking n_i different values each, will be codified by a chain of $8\prod_{i=1}^{n} n_i + C$, where C is the number of parameters defining the classifier.

The coevolutionary approach uses the number of populations equal to the number of firms participating the market (four, in our case.). Each individual represents the supply strategy for a firm (see Fig. 3

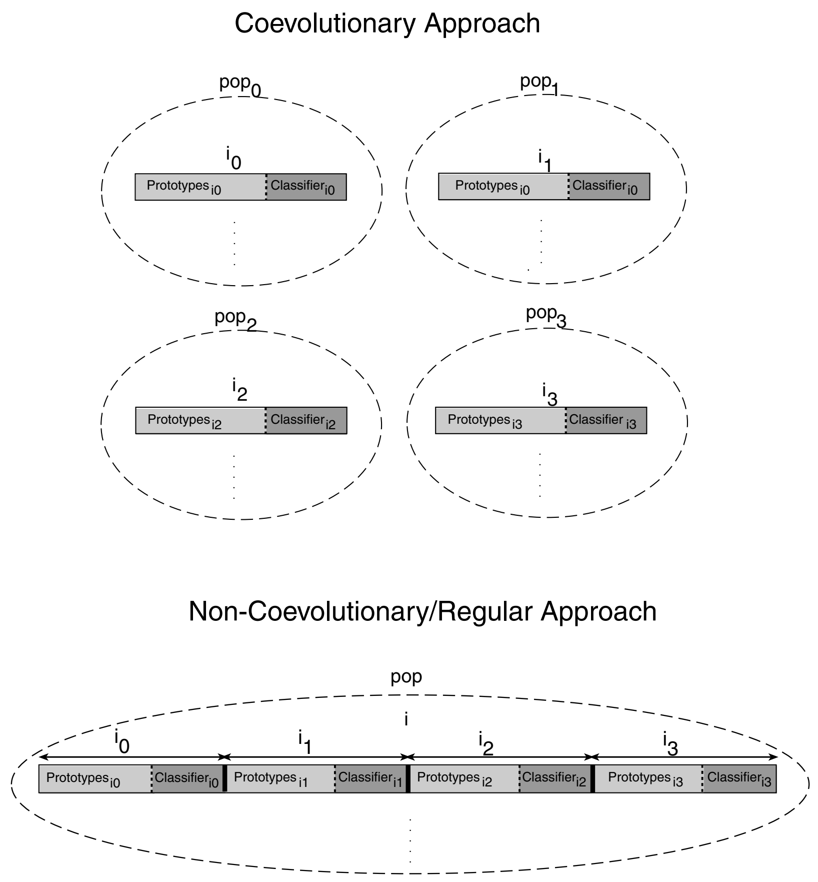

Figure 3. Genetic representation for coevolutionary approach individuals (top), and an aggregate individual (bottom) valid for the non-coevolutionary approach.

top.) Individuals in the non-coevolutionary approach are formed by concatenating as many individuals of the former type as agents exist (see Fig. 3 bottom.) Observe that the segmentation of the markets is different for every agent, thus they do not share the parameters of the classifier.

2.3. Genetic Operators

Individuals in the coevolutionary approach are represented by chains of real numbers, thus there is no need to define custom genetic operators. However, the relative sizes of the subchain codifying the consequents and the subchain codifying the list of parameters of the classifier are very different. We have opted to let only one of these parts be modified in every genetic operation, thus we can manually balance the evolution of both and speed up the evolution of the classifier part. This is the only difference between our operators and the standard versions of uniform arithmetic crossover and mutation [13, 14]. Observe that the offspring is always valid, because:

- The consequents produced by the either the mutation or the crossover operators are weighted sums of monotonic functions, which are also monotonic functions.
- The classifier arising from crossover or mutation is checked, and repaired if needed, so that all parameters defining it are constrained to their respective ranges.

When two individuals are to be crossed, a coin is tossed to decide whether we select from each of them (a) the subchain codifying the classifier definition or (b) one subchain that codifies the definition of one of the prototypes (the consequent of a single rule is modified after applying crossover.) The selected subchains are recombined by means of standard arithmetic crossover, but the remaining part of the individual remains untouched (see Fig. 4 top).

The mutation operator is defined as the crossing of an individual with another one, generated at random. Both

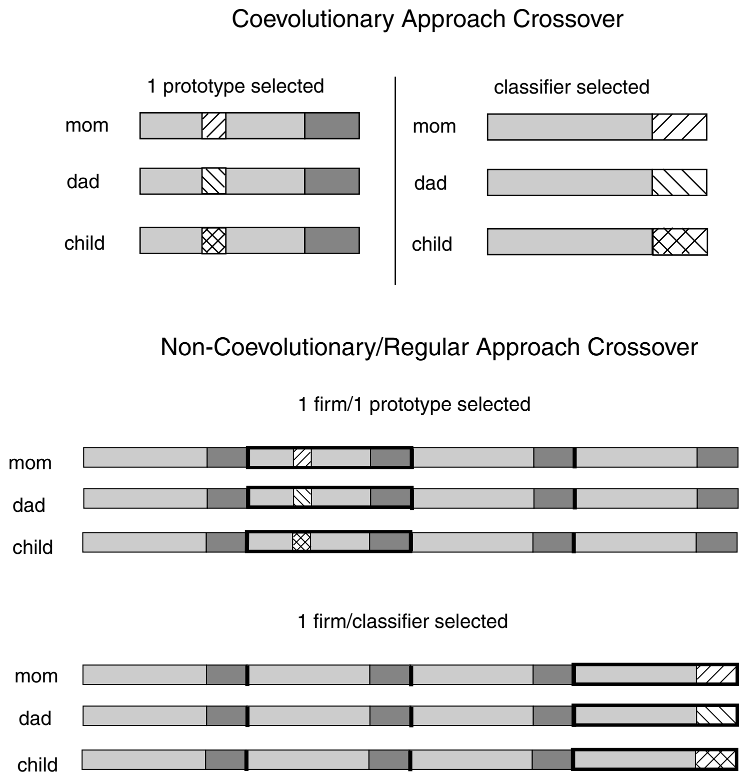
Coevolutionary Approach Crossover
1 prototype selected
classifier selected
mom
dad
child
mom
dad
child
Non-Coevolutionary/Regular Approach Crossover
1 firm/1 prototype selected
mom
dad
child
1 firm/classifier selected
mom
dad
child

Figure 4. Crossover operations in coevolutionary (top) and evolutionary (bottom) representations. Evolutionary crossover consists of selecting one agent at random and performing the coevolutionary crossover over the corresponding parts.

recombination and mutation are immediately extended to the canonical case (non-coevolutionary): the same strategy is selected from both parents (i.e., crossover affects subchains with the same relative position inside the individual) and the preceding operations are applied to the selected subchain (see Fig. 4 bottom).

2.4. *Fitness Function*

We have mentioned that we needed to assume that the unitary profits of all firms are the same in order to obtain a good model. Otherwise, we would not be able to determine the market share of every firm, only an aggregation of the supply curves of all firms that could be broken down in many different ways.

This decision implies that we need to rank strategies according to two different criteria: a strategy is good when, after being combined with the strategies of the remaining players, (a) either the price and the quantity sold are near the actual price or quantity and (b) similar unitary profits are achieved by all players –where "unitary profit" is defined as the difference between cost and income, divided by the number of MWs sold. The first goal measures how the curves fit real data, and the second one measures the degree of fulfillment of the restriction "all unitary profits are the same." We will use the square error to quantify the first objective, and the mean of the variances of the unitary profits to quantify the second one.

Different methods exist for mapping multi-objective fitness into scalar fitness [15]. We have studied the weighted average of values (a) and (b), but, according to our experiments (see Section A.2), there is a significant improvement if we use a multi-objective approach instead [16–19].

The same fitness function was applied to both the coevolutionary process (as many populations as companies) and the regular process (one population). Note that in the coevolutionary approach, the fitness of an individual is calculated making it compete with individuals selected at random from other populations, while in the evolutionary approach, an individual is a set of compound strategies from all companies, thus this selection is not necessary.

3. Demonstration Case

A simple problem is presented here for the sake of illustrating the basic aspects of the proposed methodology. This example models 10 repetitions of a game in which we know each player always uses the same supply curve, thus we do not need the classifier. Supplies are straight lines, each depending on two parameters.

Table 1. Market points in the demonstration case.

Price	12.5	15	17.5	20	22.5	25	27.5	30	32.5	35
Quantity	75	80	85	90	95	100	105	110	115	120

The inputs for this problem are:

1. The cost functions. q is the quantity of energy produced, C_0 and C_1 are the prices demanded:
$$C_0(q) = 8q + 12$$
$$C_1(q) = 8.5q + 10$$
2. The market scenario, a series of 10 demand functions (D_m) with the same elasticity (i.e., the same steepness):
$$D_m(p) = -2p + (100 + 10m), \quad \text{for } m \text{ in } 0 \ldots 9$$
3. The set of market points in Table 1, (price$_i$, quantity$_i$). They were generated from the intersections of the demand functions (D_i) and the aggregate strategy.
The original strategies of the agents (those we want to recover) are:
$$q_0^{\text{orig}}(p) = 0.10p + 16.60$$
$$q_1^{\text{orig}}(p) = 1.89p + 33.55$$
therefore, the intersections of $q_0^{\text{orig}} + q_1^{\text{orig}}$ and the preceding demand functions produce Table 1.

The algorithm is fed with Table 1, functions D_0 to D_9 and the cost functions C_0 and C_1. It produces approximations q_0, q_1 to the actual strategies q_0^{orig} and q_1^{orig}. Individuals in the coevolutionary genetic populations are chains of two numbers (coefficients defining q_0 or q_1). For the sake of clarity, let us evaluate the combination of two individuals generated at random, (0.2, 15.0) and (2.0, 10.0) which correspond to the pair of strategies:
$$q_0(p) = 0.2p + 15.0$$
$$q_1(p) = 2.0p + 10.0.$$

Table 2. Price of energy and total number of units sold for the pair of offers defined in Section 3.

Price	17.86	20.24	22.62	25	27.38	29.76	32.14	34.52	36.90	39.29
Quantity	64.29	69.52	74.76	80	85.24	90.48	95.71	100.95	106.19	111.43

Table 3. Number of units sold by each player.

q_0	18.57	19.05	19.52	20	20.48	20.95	21.43	21.90	22.38	22.86
q_1	45.71	50.48	55.24	60	64.76	69.52	74.29	79.05	83.81	88.57

One part of the fitness depends on the difference between the energy that was actually sold and the energy that would be sold if the strategies being evaluated were issued. We must first calculate the estimated prices for each market. These prices are graphically determined by the cut of $q_0 + q_1$ with each demand D_m. The results are in Table 2. The squared difference between the quantities in this table and those in Table 1 is 93.5. The other term in the fitness is the variance of the unitary benefits in all markets. Given the expressions of q_0 and q_1, the amount of energy each player sells in each market is contained in Table 3. Unitary profit is calculated by dividing the difference between income $p \cdot q(p)$ and cost $C(q(p))$ into the number q of sold units. The variance of all these values is 0.014. This is the second field in the fitness value.

Once the models of representation and the combinations of the individuals with the demand curves are defined, a coevolutionary genetic algorithm is applied to obtain an estimate of the original individual strategies. Estimated individual strategies were obtained after running the algorithm with two subpopulations with the size of 400, multicriteria fitness, 200 generations, tournament selection (size 4) and linear descending crossover probability, from 100 to 0% (this decision is experimentally justified in Appendix A.6). The output of our method is:

$$q_0(p) = 0.10p + 16.70$$
$$q_1(p) = 1.90p + 33.29$$

4. Self-Criticism: Non Coevolutionary Analysis of the Data

One may wonder whether this method offers significant advantages over simpler techniques. Let us suppose all agents share the same classifier, i.e., the antecedents of all rules are the same for all agents (for instance, let us suppose all players agree in the meaning of 'morning' and 'afternoon' in the previous example.)

In this case, we can think of applying cluster analysis to the input data (the pairs price-quantity) to estimate the antecedents of the classifier or, in other words, to do a *segmentation* of the different markets, according to their properties. Then, the input data can be divided into different subsets (market segments), and an aggregate supply curve can be fitted into each market segment by non linear regression (see Fig. 5). The share of all agents is obtained with the help of the unitary profit assumption.

To assess this method, we will set up a second algorithm, in which k-means is used to cluster the points and a genetic algorithm is used to fit a piecewise linear curve to each cluster. We will call this approach "K-means Genetic Model" (KGM). From now on, the methods proposed here in their canonical and coevolutionary versions will be called "Genetic Model" (GM) and "Coevolutionary Genetic Model" (CGM), respectively. We will show, by means of numerical simulation of a simplified real problem, that the adjustment of CGM improves that of KGM and GM.

5. Practical Application: A Simplified Real Problem

This section describes the application of this method to a semi-synthetic problem. This problem was designed to reproduce current scenarios in the Spanish electrical market, while being originated by theoretical supply curves, thus we can quantify the accuracy of GM, CGM and KGM methods.

The exposition is organized as follows:

1. A description of the semi-synthetic problem is made.
2. The experimental framework (decisions about the genetic process) is defined.
3. KGM, GM and CGM methods are compared.
4. The results are discussed and the practical application of the models studied.

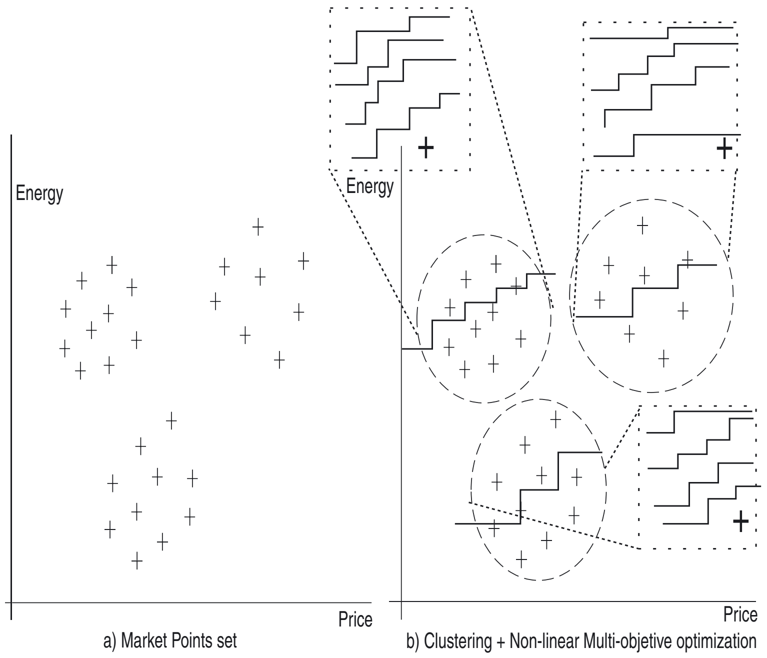

Figure 5. Approximation by classical methods: Clustering + Non-linear multi-objective optimization.

5.1. Semi-Synthesized Problem Description

The problem proposed here is more complex than the demonstration scenario shown in Section 3. Market points have been distributed resembling market points in a specific real situation (see Fig. 6(a) for actual data points for two days in the year 2000). The real problem in Fig. 6(a) has been simplified to 6(b). The reasons that forced us to use semi-synthetic data are:

- Supply curves: we have assumed a small number of aggregate supply curves exists, and a simple behavior for every firm (i.e., the supply curve only depends on the market segment given by the classifier). This is

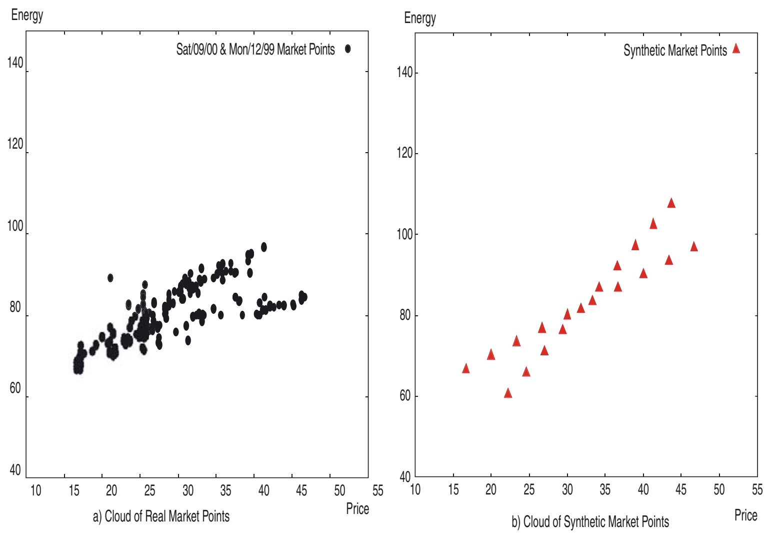

Figure 6. Saturday (September 2000) and Sunday (December 1999) market points (a) compared to semi-synthetic data (b).

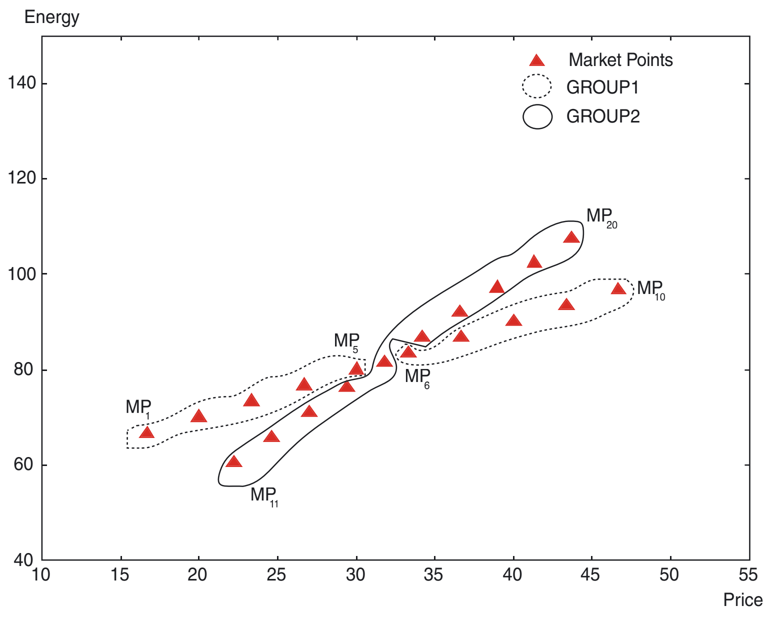

Figure 7. Cross-shaped cloud of market points for the semi-synthetic problem and the two clusters that should be obtained.

a simplification of the real case. In practice, each generator operates with certain autonomy, thus the actual aggregate supply curves are never exactly the same in two different markets. By replacing them with semi-synthetic curves, we can know whether the deviations between desired add real output of our algorithm are due to the genetic algorithm, or they are inherent in the dispersion of the data.

- Demand elasticity: The steepness of the demand curves was not provided by the MO. Therefore, we assume a plausible value.
- Both the demand and the cost functions are linearizations of the actual ones.

We decided to generate 20 points in two clusters, assuming that all points belong to weekends in the same season (and therefore the segmentation depends only on the hour of the day, see Fig. 7). We wish to recover the four individual supply curves (the Spanish electrical market is made up of four major generating agents) that originated the situation in Fig. 6(b).

The input data includes:

1. 20 market points, MP_i, $i = 1, \dots 20$ (pairs price-quantity)
2. 20 Demand functions, which were assigned the same elasticity (-2). For instance, demand function for the market point characterized by the pair (price = 16.67, quantity = 66.67) is $-2 * \text{price} + 100$.
3. 4 linear production cost functions, (one for each firm) which are

$$C_0(q) = 8q + 10$$
$$C_1(q) = 10q + 20$$
$$C_2(q) = 12q + 30$$
$$C_3(q) = 11q + 25$$

4. The quotient between unitary profits of all firms (1, in this case).
5. The number of prototypes (2).

The prototypes used to generate the market points (the desired output of the process) are shown in Table 4.

5.2. *Experimental Framework*

5.2.1. Crossover Operation. The first decision that must be made in CGM algorithm concerns the method for selecting prototypes in the crossover operation. We have evaluated two alternatives: only 1 prototype, selected at random, is interchanged between the individuals being crossed (“coevol-1-random”) or all prototypes are used in a kind of uniform crossover

Table 4. Objective individual strategies for the cross shaped problem. Prototypes are piecewise linear, with 3 segments. Each compound strategy comprises 2 prototypes, and the classifier system has two rules.

	MP_{1-10}
E0	(16.67,2.13)(26.67,2.17)(36.67,2.21)(46.67,2.25)
E1	(16.67,7.39)(26.67,7.69)(36.67,7.99)(46.67,8.29)
E2	(16.67,42.52)(26.67,51.02)(36.67,59.52)(46.67,68.02)
E3	(16.67,14.55)(26.67,15.65)(36.67,16.75)(46.67,17.85)
	MP_{11-20}
E0	(16.67,1.93)(26.67, 2.08)(36.67,2.23)(46.67,2.38)
E1	(16.67,6.40)(26.67,7.33)(36.67,8.26)(46.67,9.19)
E2	(16.67,29.35)(26.67,46.45)(36.67,63.55)(46.67,80.65)
E3	(16.67,10.63)(26.67,14.33)(36.67,18.03)(46.67,21.73)

Table 5. Framework to compare GM and CGM. Since the crossover is very disruptive, the mutation operator did not improve the convergence and it was not used. For the same reason, the probability of crossover has to be lowered as the search advances.

Selection type:	Tournament
Tournament size:	4
Number of pops.:	4 populations in coevolutionary model
Size of pops.:	100–1000
Iterations:	100–500
Crossover probability:	Linear descending from 100.0%
Mutation probability:	0.0%

that we have labelled "coevol-all" (see Fig. 8). The framework for this and future genetic experiments in this section is shown in Table 5. GM crossover is different because individuals do not codify the strategies of one firm but a combination comprising a set of strategies for every firm. CGM crossover can be extended to this schema: evol-1-random selects one firm's strategies from both parents and applies "coevol-1-random" crossover to them. "Evol-all" consists of applying "coevol-1-random" to all firms (see Fig. 9).

Observe that the settings for some of these operators are not the usual ones: mutation did not improve the convergence speed so it was not used (see Section A.7). Our experiments have also shown that the crossover was very disruptive, even after the modifications proposed in Section 2.3 were applied. Notice that the probability of crossover is lowered as the search advances due to the same reason (see Section A.6).

5.2.2. Fitness Function. Regular fitness calculation of a GM individual consists of making the firms compete with each other. This will be called "intra-i" fitness. An alternative fitness involves the competition of each firm with firms from another individual selected at random from the population. This fitness will labelled "extra-i" (see Fig. 10).

5.2.3. Summary of the Decisions Made. The diagram in Fig. 11 shows the decisions made to obtain the final genetic configuration. The first decision consists in deciding between multi-objective and scalar fitness, using CGM to evaluate both fitnesses. Our experiments (see Section A.2) conclude that multi-objective fitness is more effective than scalar fitness. Therefore, scalar fitness will not be used in the remaining decisions.

Again, in CGM, we have to choose one crossover operator between "coevol-1-random" and "coevol-all." Results in Section A.3 state that "coevol-1-random" is the best one. Therefore, "coevol-1-random" is selected as the choice crossover operator for individuals in CGM for all tests.

Two crossover operators for GM individuals are analysed: "evol-1-random" and "evol-all." In this case,

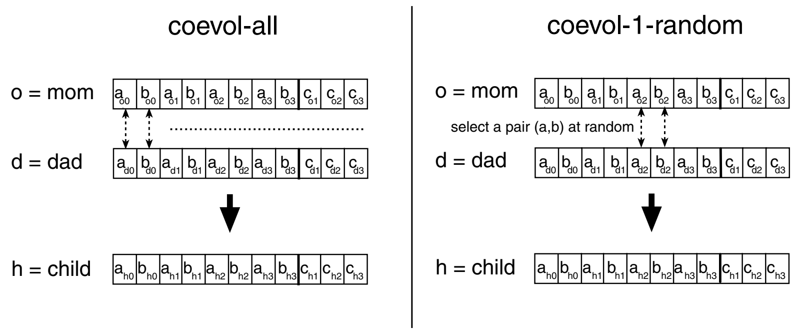

Figure 8. Types of crossover operators for CGM, (coevol-all and coevol-1-random).

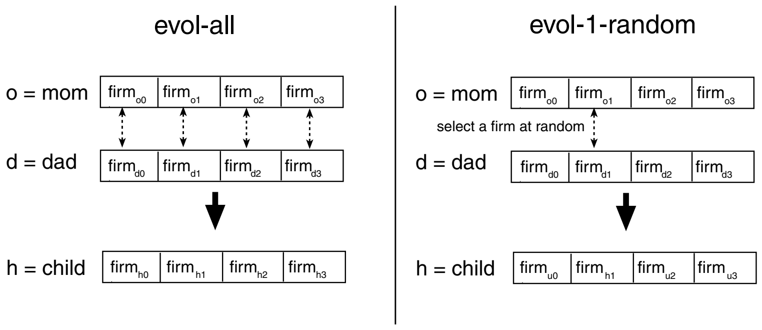
evol-all
o = mom
firm o0 | firm o1 | firm o2 | firm o3
d = dad
firm d0 | firm d1 | firm d2 | firm d3
h = child
firm h0 | firm h1 | firm h2 | firm h3
evol-1-random
o = mom
firm o0 | firm o1 | firm o2 | firm o3
select a firm at random
d = dad
firm d0 | firm d1 | firm d2 | firm d3
h = child
firm u0 | firm h1 | firm u2 | firm u3

Figure 9. Types of crossover operators for GM, (evol-all and evol-1-random).

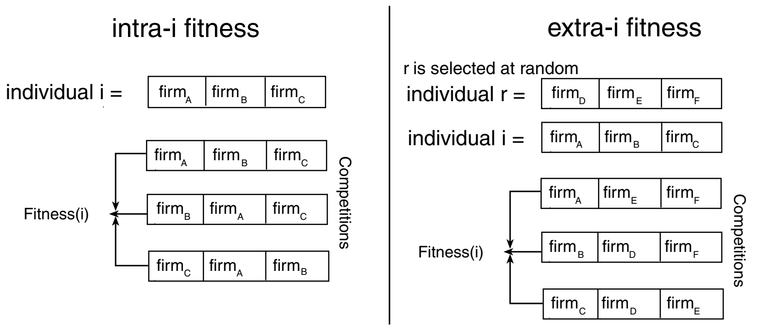

Figure 10. Types of fitness functions for GM, (intra-i and extra-i).

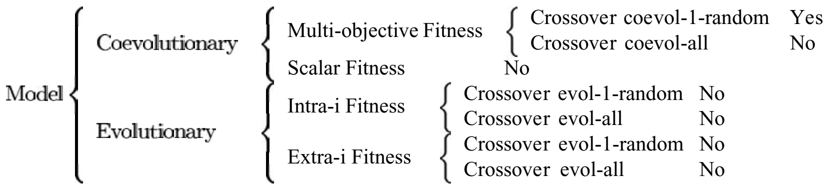

Figure 11. Diagram with the decisions taken to select fitness functions and crossover operators for GM and CGM.

results from Section A.5 show us that both crossover operators have a similar behavior.

"Extra-i" and "intra-i" fitness for GM have been studied, too. The results (see experiments in Section A.4) demonstrate that "extra-i" fitness is better than "intra-i" fitness in most of the cases studied.

Finally, after comparing GM and CGM results (see Section A.8), we can conclude that CGM with multi-objective fitness and the "coevol-1-random" crossover operator is more effective than GM.

5.3. Comparison Between KGM and CGM

We have selected the best model obtained in the previous section, CGM, to compare it with an alternative method based on "a priori" clustering (see Section 4). KGM takes two steps:

1. Clustering of market points using the K-means algorithm.
2. For each cluster of market points from the preceding step, CGM without the classification system (one prototype) is applied to fit one polygonal curve to the market points of every cluster.

After Section A.8 results, we can conclude that CGM is rather more effective than GM and KGM, but KGM can compete with the non-coevolutionary solution GM (see Table 14 in Appendix A.8). Therefore, the choice method is now CGM with coevol-1-random crossover operator and multi-objective fitness.

5.4. Analysis of Results and Applications of the Method

We will show the practical relevance of this algorithm by using it to solve a practical problem. Suppose we wish to predict what would happen if one firm increases its price.

One would expect that the total amount of total energy sold in the pool decreases, and, therefore, the final price increases. However, the share of the firms will change, too, and it is possible that the increase is profitable for the enterprise that issued it.

Let us assume the change is small enough so the remaining enterprises do not react and alter their respective strategies. In case this is not true, the tool we have developed here is not valid. One should run an algorithm similar to this in structure, but with a change in the fitness function. In order to simulate the reaction of the enterprises, it is no longer true that the strategies match the historical data, and the fitness must depend mainly on the profit of each enterprise, thus the genetic algorithm can evolve towards the optimal strategy definition in terms of the game theory. This point will be recalled in Section 6, "Concluding Remarks and Future Work".

After our algorithm has been used to extract the individual supply curves, we follow the next schema:

1. A simulation of the pool is run to calculate the profit of all firms.
2. The strategy of one firm is altered.
3. A second simulation is run to calculate the new profit.

5.4.1. Experimental Framework. Tests with an estimated pool from each model studied (GM, CGM and n-KGM models, where n is the number of clusters) have been made. Let us suppose first that we are firm 0, the firm with the lowest market share. We will increase our supply curve by 1% and evaluate the new profits. Then the supply is increased by 5% and the analysis repeated. Finally, a second, identical experiment will be carried out, but from the point of view of firm 2, the biggest company.

The value Perc_error measures the error that CGM, GM and KGM introduce into this process:

$$\text{Perc\_error} = 100 \cdot \frac{|\text{Profit increase(estimated pool)} - \text{Profit increase(actual pool)}|}{\text{Profit increase (actual pool)}}$$

It can be observed that the coevolutionary method presents (see Table 6) the lowest maximum percentage error (CGM, 5.05% of maximum percentage error) of all the methods studied. 1-KGM is the second, with 21.25% of error, after GM with 38.33% and last 2-KGM. These results suggest that clustering (KGM) is not effective. Besides the adjustment to the market points is better when the number of 2 clusters is increased (see Table 14 in Appendix A.8), this does not imply that the resulting pool is nearer to the actual one. This is experimentally stated in Table 6, where 2-KGM method obtains an error of 83.45% as opposed to 21.25% in the 1-KGM method. Therefore, we can conclude that the coevolutionary analysis of markets is a more precise technique than both non coevolutionary analysis and ordinary segmentation with submodel fitting for this application.

6. Concluding Remarks and Future Work

Coevolutionary genetic models are usually used to simulate natural systems with multiple agents of independent behavior, where mathematical models are too complex to be applied. Here, we have experimentally shown that a coevolutionary genetic model is more effective than a regular genetic model or a K-means algorithm plus genetic algorithm based on non-linear regression. On the other hand, it has been stated that multi-objective fitness is also more effective than real value fitness. Multi-objective fitness allows goals to be explored in parallel, avoiding mutual obstruction, and, therefore, the premature convergence to a local optimal.

From the point of view of a firm that manages our estimated pool, this method shows the influence of a slight variation in its strategy over its profits, given a certain market situation. This way, a firm can adjust its strategy and improve its profits.

The following step to improve this algorithm consists in designing a tool capable of generating the optimal pool, given the demand and the production costs of the firms. From the Market Operator's point of view, it is useful to know the theoretical optimal strategies for a set of market points in a perfect oligopolistic situation, which is a non-linear extension to the Cournot problem [20]. This information will serve to estimate the difference between the real profit of the pool and the theoretical maximum profit if competition is perfect, therefore detecting illegal agreements between

Table 6. Summary of percentage error obtained with GM, CGM and KGM.

Firm	% Modif.	Method	$\text{Prof}_{\text{before}}$	$\text{Prof}_{\text{after}}$	Increase	Percentage error
0	1.0	Actual	883.2	892.0	+8.8	0.00
0	1.0	CGM	883.1	891.9	+8.8	**0.00**
0	1.0	GM	881.2	890.1	+8.8	0.79
0	1.0	1-KGM	886.3	895.1	+8.8	0.09
0	1.0	2-KGM	864.2	872.5	+8.4	5.24
0	5.0	Actual	883.2	927.3	+44.1	0.00
0	5.0	CGM	883.2	927.2	+44.1	**0.01**
0	5.0	GM	881.2	925.7	+44.4	0.81
0	5.0	1-KGM	886.3	930.4	+44.0	0.10
0	5.0	2-KGM	864.2	906.0	+41.8	5.26
2	1.0	Actual	23887.3	23899.6	+12.3	0.00
2	1.0	CGM	23929.2	23942.0	+12.8	**4.10**
2	1.0	GM	23593.6	23601.3	+7.7	37.40
2	1.0	1-KGM	23922.9	23932.7	+9.8	20.33
2	1.0	2-KGM	23893.6	23915.5	+21.9	78.05
2	5.0	Actual	23887.3	23944.7	+57.4	0.00
2	5.0	CGM	23989.5	23929.2	+60.3	**5.05**
2	5.0	GM	23593.6	23629.0	+35.4	38.33
2	5.0	1-KGM	23922.9	23968.1	+45.2	21.25
2	5.0	2-KGM	23893.6	23998.9	+105.3	83.45

generators. In addition, it will serve to simulate the pool under large changes in the supply curves of one firm, overpassing the limitations of the analysis in the last section.

Appendix A: Experiments

This Appendix includes numerical results for experiments made with the semi-synthetic problem from Section 5 to select the best genetic parameters. These results have been moved here to improve the readability of the paper. General framework and specific error functions are described in Section A.1.

Section A.2 shows results for multi-objective fitness compared to scalar fitness using CGM. Results of comparing proposed crossover operators for CGM are included in Section A.3. Different fitness function results for GM are shown in Section A.4. Section A.5 retrieves results for crossover operators with GM. Studies of the crossover and mutation probability are presented in Sections A.6 and A.7 respectively and finally a summary table is included in Section A.8.

A.1. Framework and Error Measurement

All tests were made using genetic parameters from Table 5. Different fitness functions and crossover operators will be analyzed in each of the following sections.

In order to provide complete information about the results obtained, two error functions have been used to measure the distance between the estimated curves and the original supply curves:

- Error_1 measures the mean percentage error of estimated quantity for all firms in each market.
- Error_2 measures the mean percentage error of estimated energy share in each market for all firms.

A.2. Mono-Objective vs. Multi-Objective Fitness, Coevolutionary Model

Tests made with multi-objective fitness, scalar fitness and a coevol-1-random crossover operator (see results in Section A.3) using CGM are in Table 7. Results

Table 7. Comparison of errors using multi-objective and scalar fitness for CGM.

Fitness	Pop	Iter	$Error_1$	σ_{error_1}	$Error_2$	σ_{error_2}
Multi-objective fitness	100	200	**2.427**	**0.872**	**0.411**	**0.157**
Scalar fitness	100	200	253.896	126.628	22.695	6.821
Multi-objective fitness	200	200	**2.189**	**0.965**	**0.400**	**0.213**
Scalar fitness	200	200	240.562	99.271	20.013	6.778
Multi-objective fitness	300	200	**1.835**	**0.691**	**0.342**	**0.131**
Scalar fitness	300	200	261.687	64.373	19.510	4.912
Multi-objective fitness	400	200	**1.842**	**0.663**	**0.364**	**0.149**
Scalar fitness	400	200	269.891	72.581	22.769	4.962
Multi-objective fitness	500	200	**1.631**	**0.748**	**0.332**	**0.176**
Scalar fitness	500	200	246.641	96.271	20.926	5.624
Multi-objective fitness	1000	200	**1.118**	**0.523**	**0.197**	**0.088**
Scalar fitness	1000	200	255.751	62.520	21.408	4.483
Multi-objective fitness	500	500	**1.327**	**0.782**	**0.270**	**0.181**
Scalar fitness	500	500	215.066	112.140	17.916	6.920
Multi-objective fitness	1000	500	**1.168**	**0.640**	**0.253**	**0.139**
Scalar fitness	1000	500	223.648	126.607	19.170	8.001

are the average of 10 runs with each set of parameters: type of fitness (Fitness), number of individuals by population (Pop) and number of iterations (Iter).

Multi-objective fitness obtains the best solution with both quality criteria (absolute error –$Error_1$– and distribution of market shares –$Error_2$–). It seems that multi-objective fitness converges in parallel to a global minimum in its two components (variance of unitary profits –$Fitness_b$– and square error –$Fitness_a$–) while scalar fitness, which is a weighted sum of both, is dragged by the easiest convergence component, causing convergence to a local minimum (see Fig. 12).

A.3. Selection of the Crossover Operation, Coevolutionary Model

Numerical results for the two proposed crossover operators, coevol-1-random and coevol-all, and multi-objective fitness are in Table 8. It shows average results of 10 runs with each set of parameters: type of crossover operator (Crossover), number of individuals by population (Pop) and number of iterations (Iter).

Although the margin between both variants is small, coevol-1-random is better than coevol-all in all tests.

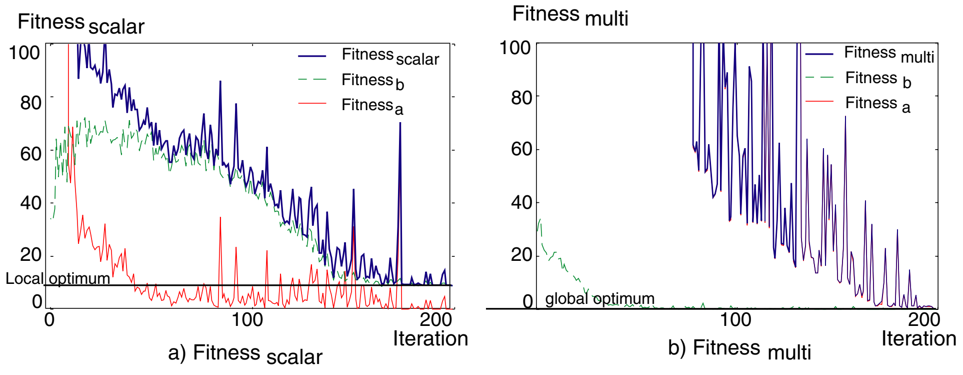

Figure 12. Comparison of Scalar (a) vs. Multi-objective (b) Fitness Progression.

Table 8. Crossover methods comparison for a coevolutionary model.

Crossover	Pop	Iter	$Error_1$	σ_{Error_1}	$Error_2$	σ_{Error_2}
coevol-all	500	200	2.310	0.997	0.383	0.182
coevol-1-random	500	200	**1.631**	**0.748**	**0.332**	**0.176**
coevol-all	500	500	1.744	0.755	0.356	0.167
coevol-1-random	500	500	**1.327**	**0.782**	**0.270**	**0.181**
coevol-all	1000	200	1.778	0.526	0.282	0.073
coevol-1-random	1000	200	**1.118**	**0.523**	**0.197**	**0.088**
coevol-all	1000	500	1.241	0.715	0.246	0.159
coevol-1-random	1000	500	**1.168**	**0.640**	**0.253**	**0.139**

A.4. Fitness Function, Non Coevolutionary Model

Both GM fitness functions, "intra-i" fitness and "extra-i," fitness have been tested with both GM crossover operators, evol-1-random and evol-all (see Section A.5).

We propose an alternative to regular fitness (intra-i fitness) that consists in making each firm comprising an individual compete with firms from another individual selected at random from the population. Table 9 contains average results of 10 runs with each set of parameters: type of fitness (Fitness), type of crossover operator, number of individuals by population (Pop) and number of iterations (Iter).

Table 9. Intra-i fitness vs. extra-i fitness for GM.

Fitness	Pop	Iter	$Error_1$	σ_{error_1}	$Error_2$	σ_{error_2}
		Evol-all crossover				
intra-i	500	200	12.742	4.032	2.245	0.885
extra-i	500	200	7.844	1.744	0.793	0.333
intra-i	500	500	5.708	1.880	1.252	0.443
extra-i	500	500	4.495	1.959	0.718	0.252
intra-i	1000	200	11.181	2.660	1.899	0.503
extra-i	1000	200	7.230	1.276	0.720	0.346
intra-i	1000	500	4.822	1.163	1.003	0.237
extra-i	1000	500	4.223	1.858	0.607	0.220
		Evol-1-random crossover				
intra-i	500	200	35.972	17.795	4.131	2.317
extra-i	500	200	27.063	15.330	1.463	0.495
intra-i	500	500	4.268	1.746	0.899	0.410
extra-i	500	500	5.266	3.894	0.962	0.613
intra-i	1000	200	14.577	5.724	1.888	0.690
extra-i	1000	200	17.239	9.279	1.208	0.654
intra-i	1000	500	3.949	1.331	0.867	0.370
extra-i	1000	500	3.875	1.871	0.750	0.340

In conclusion, we can affirm that the method of competition of external fitness (extra-i) is better in almost all the cases studied. This difference is attenuated as the number of individuals and iterations grows.

A.5. Crossover Operator, Non Coevolutionary Model

We propose two GM crossover operators, one similar to coevol-1-random, but crossing whole firms and not prototype strategies. Let us define a CGM scenario with p populations, i individuals by population and crossover probability of c, so the expected next generation number of individuals produced by the crossing operator will be:

$$\text{Crossed CGM individuals/population} = \frac{2ci}{100+c}$$

$$\text{Total crossed CGM individuals} = \frac{2cip}{100+c}$$

The population of the equivalent GM has i individuals (like a CGM population), but each GM individual will be made up of p firms (a firm is a set of strategies, such as a CGM individual). According to this, if the crossing between two GM individuals composed by m population segments is implemented as the crossing of all the corresponding firms comprising it (evol-all), the number of firms after crossing will be the same number as CGM individuals:

$$\text{Crossed GM firms/GM individual} = \frac{2ci}{100+c}$$

$$\text{Total crossed GM firms} = \frac{2cip}{100+c}$$

Thus, we propose a new GM crossover consisting of the random selection of only one firm for the crossing (evol-1-random), instead of crossing all the firms. The results of Table 10 are the average of 10 runs for each set of parameters: type of crossover (Crossover), number of individuals by population (Pop) and number of iterations (Iter).

Table 10 shows that evol-1-random crossover improves with the population size (Pop) and the number of iterations (Iter). Evol-1-random is better than evol-all from 500 iterations. This demonstrates that the number of crossings is not totally decisive in the problem that we are raising, since the errors ($Error_1$, and $Error_2$) are not improved with a new evolutionary crossover

Table 10. Crossover methods comparison for evolutionary model.

Crossover	Pop	Iter	$Error_1$	σ_{error_1}	$Error_3$	σ_{error_3}
evol-all	500	200	12.742	4.032	2.245	0.885
evol-1-random	500	200	35.972	17.795	4.131	2.317
evol-all	1000	200	11.181	2.660	1.899	0.503
evol-1-random	1000	200	14.577	5.724	1.888	0.690
evol-all	500	500	5.708	1.880	1.252	0.443
evol-1-random	500	500	4.268	1.746	0.899	0.410
evol-all	1000	500	4.822	1.163	1.003	0.237
evol-1-random	1000	500	3.949	1.331	0.867	0.370

operator with respect to the equivalent coevolutionary model.

A.6. *Crossover Probability Study*

All the previous studies were carried out using a crossover probability calculated with the expression $100(1 - \frac{iter}{NTotalIter})$, with which the best practical results were obtained. To confirm this statement, the results were tabulated for several tests with a constant crossover probability (between 1.0% and 100%) in the CGM model with multi-objective fitness, using the crossover operator "coevol-1-random" (see Table 11).

Observe that as the crossover probability increases, the value of $Error_1$ worsens, and the minimum crossover probability of 10% is obtained. This indicates that the population diversity remains high until the end of CGM learning, when the crossover probability is still high, which favors high dispersion. For this reason, we decided on the use of a descending crossover probability. The minimum errors, $Error_1$ as well as $Error_2$, obtained with constant crossover probability, do not improve the results obtained with the descending probability of 1.631% (see Table 12).

Table 11. Errors from applying a constant crossover probability of 10–100% to the CGM model combined with the multi-objective fitness and crossover operator "coevol-1-random".

%Crossover probability	$Error_1$	σ_{Error_1}	$Error_3$	σ_{error_3}
Descending Probability	**1.631**	**0.748**	**0.332**	**0.176**
1	5.340	2.957	0.595	0.257
5	3.419	0.950	0.611	0.232
10	*3.010*	*1.344*	0.554	0.204
20	3.580	2.350	*0.530*	*0.223*
30	3.374	1.501	0.594	0.292
40	3.730	1.250	0.652	0.220
50	4.341	1.569	0.770	0.309
60	4.604	1.900	0.806	0.284
70	6.418	3.029	1.043	0.460
80	6.160	2.867	0.997	0.413
90	6.099	2.659	0.971	0.406
100	6.511	2.556	1.100	0.477

Table 12. Comparison of the best results obtained with a constant crossover probability as opposed to a descending crossover probability with a linear function

Crossover probability	$Error_1$	σ_{Error_1}	$Error_3$	σ_{error_3}
Const. 10%	3.010	1.344	0.554	0.204
Const. 20%	3.580	2.350	0.530	0.223
Descending Probability	**1.631**	**0.748**	**0.332**	**0.176**

A.7. *Mutation Probability Study*

Uniform arithmetic crossover operators already include a random factor, which is why the possibility of not using a mutation operator was assessed. To experimentally contrast this decision, testing was carried out with positive mutation probabilities from 1.0 to 10%. Table 13 contains the results obtained with a mutation operator applied with positive probabilities (% Mut.) and without it (probability 0%), both combined with the CGM model with multi-objective fitness and crossover operator "coevol-1-random".

In view of the results obtained, we can say that the errors increase as the mutation probability increases, which is why it was decided not to use this operator.

A.8. *Summary*

As shown in Table 14, it can be concluded that the coevolutionary model using multi-objective fitness is

Table 13. Comparison of errors obtained without mutation (0%) vs. positive mutation probability with CGM and multi-objective fitness.

% Mut.	$Error_1$	σ_{Error_1}	$Error_3$	σ_{Error_3}
0.0	**1.631**	**0.748**	**0.332**	**0.176**
0.5	*1.949*	*0.834*	*0.373*	*0.148*
1.0	2.323	1.286	0.420	0.183
5.0	3.024	0.587	0.488	0.083
10.0	3.132	0.330	0.450	0.073

Table 14. Comparison of errors obtained with CGM, GM and KGM.

Model	Cross./fitness	Pop	Iter	$Error_1$	σ_{error_1}	$Error_2$	σ_{error_2}
1-KGM	–	500	200	3.716	0.000	0.423	0.000
2-KGM	–	500	200	3.580	0.000	0.515	0.000
GM	All/extra-i fitness	500	200	7.844	1.744	0.793	0.333
CGM	1-random/multi-objective fitness	500	200	**1.631**	**0.748**	**0.332**	**0.176**
GM	1-random/intra-i fitness	500	500	4.268	1.746	0.899	0.410
CGM	1-random/multi-objective fitness	500	500	**1.327**	**0.782**	**0.270**	**0.181**
GM	All/extra-i fitness	1000	200	7.230	1.276	0.720	0.346
CGM	1-random/multi-objective fitness	1000	200	**1.118**	**0.523**	**0.197**	**0.088**
GM	1-random/extra-i fitness	1000	500	3.875	1.871	0.750	0.340
CGM	1-random/multi-objective fitness	1000	500	**1.168**	**0.640**	**0.253**	**0.139**

more effective than the evolutionary equivalent model, and also better than the equivalent KGM.

Acknowledgments

This research has been partly funded by the "Fundación Banco Herrero 2002 Research Grant" and by "Ministerio de Ciencia y Tecnología," under the research proyect TIC2002-04036-C05-05.

References

1. Ministerio de Industria y Energía, "Protocolo para el establecimiento de una nueva regulación del sistema eléctrico nacional," Technical report, Ministerio de Industria y Energía, 1996.
2. J. Arifovic, "Genetic algorithm learning and the cobweb model," *Journal of Economic Dynamics and Control*, vol. 18, pp. 3–28, 1994.
3. T.C. Price, "Using co-evolutionary programming to simulate strategic behaviour in markets," *Journal of Evolutionary Economics*, vol. 7, pp. 219–234, 1997.
4. R. Franke, "Coevolution and stable adjustments in the cobweb model," *Journal of Evolutionary Economics*, vol. 8, pp. 383–406, 1998.
5. H. Dawid, *Adaptative Learning by Genetic Algorithms*, Springer-Verlag: Berlin, 1999.
6. W.W. Leontief, "Verzögerte angebotsanpassung und partielles gleinchgewicht," *Zeitschrift für Nationalökonomie*, vol. 5, pp. 670–676, 1934.
7. R.E. Marks, "Breeding hybrid strategies: Optimal behaviour for oligopolists," *Journal of Evolutionary Economics*, vol. 2, pp. 17–38, 1992.
8. R.E. Marks and L.G. Cooper, "The complexity of competitive marketing strategies," *Complexity International*, vol. 6, 1999.
9. W.D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization technique," *Artificial Life II*, pp. 41–47, 1991.
10. M.A. Potterm, K.A. De Jong, and J.J. Grefenstette, "A coevolutionary approach to learning sequential decision rules," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, edited by Larry Eshelman, Morgan Kaufmann: San Francisco, CA, 1995, pp. 366–372.
11. J. Paredis, "Coevolutionary computation," *Artificial Life*, vol. 2, no. 4, pp. 355–375, 1995.
12. C.D. Rosin and R.K. Belew, "New methods for competitive coevolution," *Evolutionary Computation*, vol. 5, no. 1, pp. 1–29, 1997.
13. W.M. Spears and K.A. De Jong, "On the virtues of parameterized uniform crossover," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, edited by R. Belew and L. Booker, Morgan Kaufman: San Mateo, CA, 1991, pp. 230–236.
14. G. Syswerda, "Uniform crossover in genetic algorithms," in *Proceedings of the Third International Conference on Genetic Algorithms*, edited by R. Belew and L. Booker, Morgan Kaufman: San Mateo, CA, 1989, pp. 2–9.
15. Y.J. Lai and C.L. Hwang, *Fuzzy Multiple Objective Decision-Making: Methods and Applications*, Springer-Verlag: New York, 1996.
16. D.A. Veldhuizen and G.B. Lamont, "Multiobjective evolutionary algorithms: Analyzing the state-of-the-art," *Evolutionary Computation*, vol. 8, no. 2, pp. 125–147, 2000.
17. C.M. Fonseca and P.J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, 1995.
18. C.M. Fonseca and P.J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms—Part II: A application example," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 28, no. 1, pp. 38–47, 1998.
19. J. Horn, N. Nafpliotis, and D.E. Goldberg, "A Niched Pareto genetic algorithm for multiobjective optimization," in *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, IEEE Service Center: Piscataway, NJ, vol. 1, 1994, pp. 82–87.
20. A. Cournot, *Recherches sur les Principies Mathématiques de la Théorie des Richesses*, Hachette, Paris, 1838.