# Using Genetics Based Machine Learning to find Strategies for Product Placement in a dynamic Market

Fent, Thomas

Vienna University of Economics and Business Administration

October 1999

# Using Genetics Based Machine Learning to find Strategies for Product Placement in a dynamic Market

Thomas Fent

Working Paper No. 55
October 1999

October 1999

This piece of research was supported by the Austrian Science Foundation (FWF) under grant SFB#010 ('Adaptive Information Systems and Modelling in Economics and Management Science').

# Using Genetics Based Machine Learning to find Strategies for Product Placement in a dynamic Market

Thomas Fent

October 1999

**Abstract**

In this paper we discuss the necessity of models including complex adaptive systems in order to eliminate the shortcomings of neoclassical models based on equilibrium theory. A simulation model containing artificial adaptive agents is used to explore the dynamics of a market of highly replaceable products. A population consisting of two classes of agents is implemented to observe if methods provided by modern computational intelligence can help finding a meaningful strategy for product placement. During several simulation runs it turned out that the agents using CI-methods outperformed their competitors.

## 1 Introduction

Many methods used in today's business administration are based on microeconomic theory. Therefore, neoclassic approaches assuming that there exists a stable equilibrium play a major role. Such approaches typically assume

1. perfect information about the analysed problem and its structure,
2. diminishing returns, and
3. only perfectly rational individuals.

However, in reality the individuals lack complete information and different participants interpret the same information in a different way.

Many of these models are very elegant from a mathematical point of view. They may be appropriate for describing an agricultural or manufacturing economy, but they are not at all suitable for a market determined by innovation, change, and uncertainty. A typical neoclassical model is based on simple assumptions about the individuals' behaviour. The benefits obtained by such models are the mathematical proofs confirming the results. The shortcomings, however, are that the behaviour of the interacting agents is restricted to the simple assumptions. If the assumptions exclude important aspects of the real world, then the answers delivered by the model are irrelevant. In order to (at least partially) overcome those shortcomings we will make use of computer simulations based on assumptions that are too complex to be included into a neoclassical anlytical model. This gives us the chance to consider aspects of learning and adaption.

## 1.1 Shortcomings of equilibrium-based models

The world in which we live is not static, nor does it converge to a stable-state equilibrium at all. If it were, it would be almost imposible for a new entrepreneur to succeed in a market segement that is already covered by big suppliers with decades of experience. Innovation and growth cannot be explained as internal effects of an equilibrium-based model but just as a result of random exogenous shocks (see Beinhocker, 1997).

Bloomberg News for instance showed that it is possible for a newcomer to outperform established competitors within a few years. What they did was reinventing existing services and providing additional features which were experienced as additional values by the customers. Such changes, which take place in many lines of todays business, usually do not occur within a mathematical framework based on equilibrium theory. Neither can such a framework describe the appearance of deregulation, profound technological change, industry convergence, globalisation, and increasing returns. The latter phenomena for instance occur in internet business or telecommunication services, which ensures that even the second of the above assumptions must be altered.

To choose a certain strategy of any kind, the situation has to be analysed first. Based on some observations one might choose an appropriate strategy and, finally, a decision (an action) is derived as an outcome of the chosen strategy. Whenever an individual takes a decision based on information collected one time step ago, the decision can only be optimal with respect to the environment as it was one time step before. If, in the sequel, the individual sticks to the strategy that was good with respect to the formerly observed situation, the difference between the assumptions and reality becomes bigger and bigger.

## 1.2 Complex adaptive systems

A dynamic market makes it difficult for the participants to maintain their competitiveness and survive in the long-run. Companies with a lot of experience in their main business units might feel motivated to rely on their strategies that have proven to be successful in the past. To survive dramatic shifts of customers' tastes or sudden technological changes, a company must be as good or even better at evolving as the environment. If the company is not capable of performing the required level of evolution, then the market will do it by the entry and exit of firms. Therefore, companies that want to remain successful within a dynamic environment have to keep on observing, analysing, learning, and adopting continously. This process is illustrated in figure 1.

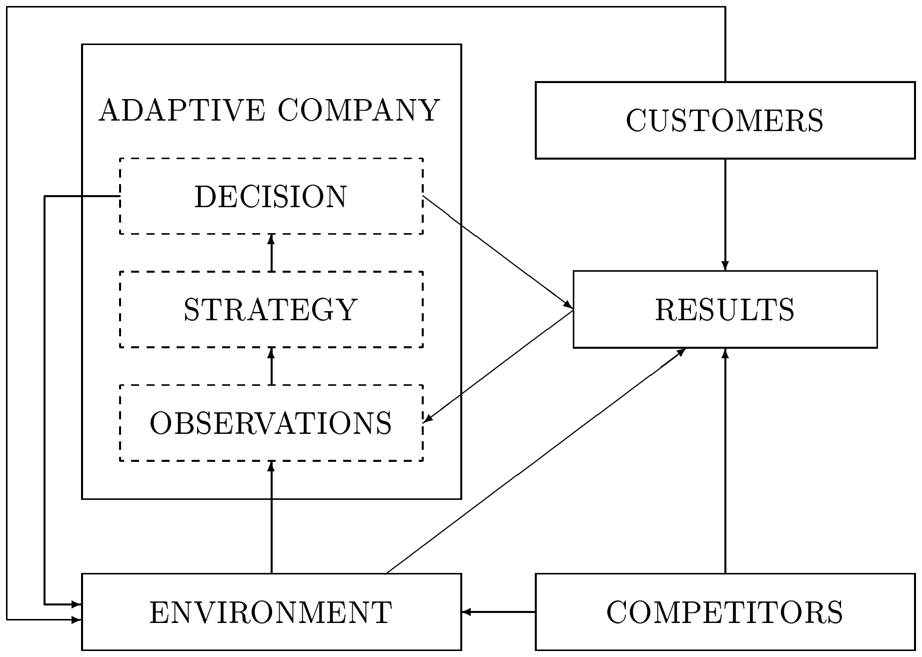

Figure 1: The process of adaption

Moreover, they must be prepared to face a big variety of situations and adopt their behaviour even when the system is in a state that has never been experienced before. Hence, a big bundle of optional strategies might be a valuable tool in a modern economy. A market built up by many individuals of that kind can be seen as a complex adaptive system (CAS). The main characteristics of CASs are:

**open and dynamic:** Only a closed system without external (exogenous) influences can tend to a stable-state equilibrium and persist there. In a CAS the individuals are always aware of unpredictable changes and adapt their behaviour whenever such shifts are encountered.

**interacting agents:** Decisions taken by one agent have an impact on the environment of all the agents and, in turn, they all have to adjust their strategy to the new situation. Thus, when one agent, due to evolution, changes her/his main strategy, this does not only effect the environment but also the evolution of the other agents. This causes the inherent dynamics of a CAS which makes it so unlikely to arrive to a steady-state. A typical example are the huge changes of stock prices. In a typical classical model they can only be explained by external perturbations, while in fact they are just a result of the investors' manners of trade.

**emergence and self-organisation:** There is no central planner deciding what and when to happen, but there are many intelligent individuals taking their own decisions. Only the whole bundle of actions and reactions can determine the behaviour of the system.

Practical examples of CASs are cities (not located in a dictatorial country), ecosystems, the internet, economies, and firms with a fractal structure.

Another reason why microeconomics cannot always describe an economy is the shortcoming of human reasoning. In many situations humans are just overtaxed when they have to analyse all the information they have, transform it to knowledge, think of all the connections that might exist, and finally, take the right decision taking into consideration several conflicting objectives. Therefore, most individuals use their experience. They compare the current situation with things that happend in the past, and guess which of their rules created previously fits best to the new problem. This even qualifies such models to explain why two individuals facing the same pattern of information behave differently due to their different experience. In a classical microeconomic model such effects are very difficult to give reason for.

Human beings often use induction instead of deduction, and that's exactly what agents in CASs are doing. Both, the real-world individuals and the agents in CASs try to recognize patterns and develop and apply inductive rules of thumb. This even works in case of incomplete or changing information. Most of the time not even all the available information is actully taken into consideration. Real-world individuals have different reasons to change their behaviour, such as mistakes, curiosity, or external perturbations. Typical artificial agents use variation, elimination, and imitation to update their rule-base.

Two examples of methods which can be used to simulate such an adaptive company are *classifier systems* (CS) and *genetic algorithms* (GA), which will be described in the sections 1.3 and 2, respectively. CSs have the ability to pursue several paths simultanously, which is a prerequisite to prosper in a CAS. On the other hand, GAs enable our artificial agents to discard unsuccessful rules, recombine the successful ones, and make some random changes to create rules that might be helpful in completely new situations. Brenner (1996) claimed that evolutionary algorithms cannot descibe the change of an individuals behaviour due to dissatisfaction about the achieved results. Nevertheless, in this work it will be shown that a synthesis of CSs and GAs can do that.

In section 2 we formulate a model of product placement in a market with several competing vendors supplying highly replaceable products. The needs of the consumers cannot be observed directly but only the sales of all the suppliers. Therefore, the process of adaption has to be capable of detecting indirect connections. Finally, in section 3 we summarize the results obtained by our simulations, draw some conclusions, and outline a couple of promising possible extensions of the present work.

## 1.3 Classifier systems

To model a connection between input and output signals consisting of vectors of integer (or binary) entries we use classifier systems (CS). CS were first introduced by (Holland, 1976) as a tool for pattern recognition. They can be seen as a vehicle to use GAs in studies of machine learning (Holland, 1995).

### The rule base

The main part of a CS is the rule base consisting of the condition part and the action part (see Fig. 2). The conditions within one particular row plus the action in the same line represent a rule, which can also be called a classifier. The conditions may contain integer (binary) entries plus the so called **don't care** symbols #. Thus, in the most simple implementation only the three symbols 0, 1, and # are permitted. On the other hand, the entries of the message list and the action part are restricted to integer (binary) values. The message list represents the information the individual receives, already encoded in a way appropriate for further computation. First, the information in the message list has to be compared with the conditions. Whenever there exists a message that is equal to a condition, except those bits where a # occurs, then the condition is considered to be fulfilled. A whole rule is fulfilled, when all it's conditions are fulfilled. Thus in the example in figure 2 the first, the third, and the fourth rule are fulfilled.

| message list | rule base | | |
|---|---|---|---|
| | condition part | | action part |
| | cond. 1 | cond. 2 | |
| 1 0 1 | 1 0 1 | 0 # 1 | 0 1 1 |
| 0 1 1 | # 1 0 | 1 0 # | 1 1 0 |
| 1 0 0 | 1 # # | 1 # 1 | 0 0 0 |
| 0 0 1 | # 0 # | 0 0 # | 1 1 1 |
| 1 1 1 | # 0 1 | 1 1 0 | 0 1 0 |

Figure 2: The message list and rule base of a classifier system

### Choosing an action

Now all the fulfilled rules become candidates to post an action. If, like in the previous example, more than one rule is fulfilled, one of them has to be selected randomly. Usually the strength of the rule, wich depends on the success of this rule in the past, and the strictivity are used to weigth the rules. The strictivity is a measure of the frequency of the don't care symbols. A very general rule, i.e. a rule containing many #, certainly has a higher chance to get selected, because it will be fulfilled more often. To compensate this, it is neccesary to favour the more specific rules. Moreover, we can assume that a more specific rule might yield a better solution to a particular situation. Another reason to favour those rules containing only a few #.

The chosen action may be posted directly to the environment, or it may be used as an internal message, and brought back to the message list. Thus, it will be considered as an input in the next time step. In figure 3 the core part of the classifier system, which has already been shown in detail in figure 2, is placed into a dashed box. The arrow pointing from the right edge of the rule base to the message list illustrates the stream of internal messages. To distinguish between internal messages and output signals, one bit of the action part has to be reserved to determine the type of the signal.

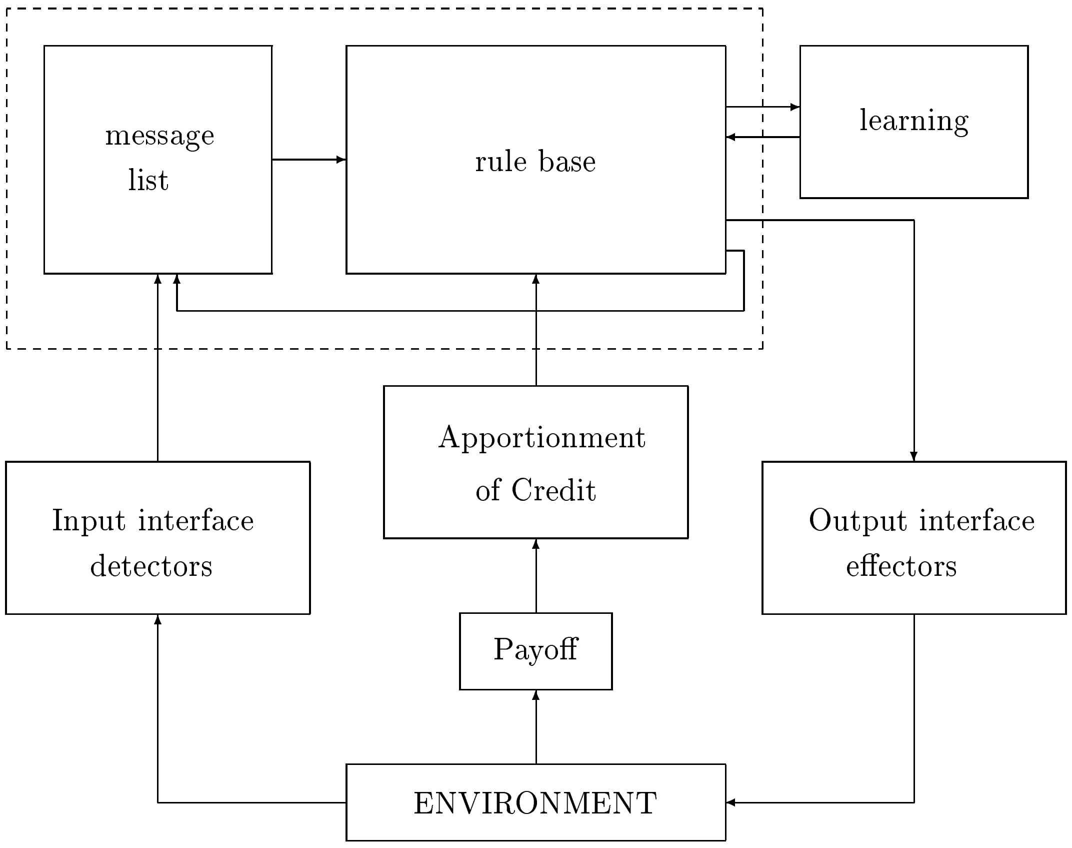

Figure 3: A classifier system

Now, how can the classifier system get connected to the environment. In general it is assumed that there exists an input interface, which translates the information available in the environment into signals that can be interpreted by the classifier systems. Thus, the signals generated by the input interface must be vectors with a fixed length, containing only integer (binary) entries. These signals are collected in the message list, and proceeded as described in the above paragraphs.

In case the action is posted to the environment, it has to be translated by the output interface. For instance, if a CS is used to play chess, the output interface translates each possible vector into a particular move. Whatever kind the chosen actions are, it influences the environment, and may yield a good or bad situation for the individual represented by the CS.

### Apportionment of credit

Now we need a mechanism to evaluate all the situations that can occur in the environment. This may be a problem sometimes. If again we think about a CS trained to play chess there are three possible outcomes, a win, a loss, and a draw. Although it seems to make sense to rate them with 1, -1, and 0 - or to use any equivalent scale - there arises the problem, that only at the end of the game the result is known. If the value of the result of the game gets assigned to all the rules activated during the game, it may happen that extremely good rules get assigned a bad value and vice versa. Moreover, even when there is a possibility to evaluate the state at every time step, in a CS with internal messages there is still the problem of assigning meaningful values to those rules that caused internal messages. A possible solution is provided by the bucket brigade algorithm (see Goldberg, 1989, p. 255 ff).

### Learning better rules

A CS creates new classifiers (rules) by running a genetic algorithm - or any other suitable learning algorithm - on the present population of classifiers. To avoid an extremely volatile behaviour of the system, the incoming messages have to be processesed through the classifier system several times before the genetic algorithm may be invoked. In the following the variable $r_p$ is used to denote the number of repetions, and it will be assigned the values 5, 10, and 20.

## 2 A model of product placement

To launch a new product the marketing department has to decide about the kind of customer attributes they would like to meet. In a heterogenous market with different customers' tastes and several competitors, taking the decision about product placement is rather complex.

Customers usually choose that product which best fits their desire, as indicated for instance by (Kotler et al., 1996, p. 7) *"They therefore want to choose products that provide the most satisfaction for their money."* To make sure that one particular customer buys, a producer could decide to customize his/her offer according to the wishes of that customer. However, this might lead to a product that no one else would like to buy. Certainly this is not a very favourable situation for a supplier, except in case this one customer has such a great purchasing power that indeed designing a product for one particular individual can still yield a good profit. Some practical examples for this situation are custom-made suits, paper-making machines, or power stations.

To avoid dependence on one customer the producer could decide to place her/his product such that the distance to most customers' requirement profiles is as small as possible. Again, this might not always be the optimal strategy. If all the competitors already try to launch such a mass product that represents the average of all the customers' wishes then offering another average product might not lead to great success.

Thus, for deciding what kind of product to supply one has to be aware of the customers' desires and the competitors' products as well. An interesting example of a simulation capable of analysing a market with very different agents can be found in (Polani and Uthmann, 1999). However, in this paper we will analyze how adaptive agents, who use classifier systems to take the decision and learn by using genetic algorithms, would place their products in a dynamic and heterogenous market.

In our model we assume that there might be $m$ customers and $n$ suppliers in a market of highly replaceable products[1]. The products are assumed to have only two different attributes (this assumption is made to facilitate visualization) and each attribute can take 10 different values. Thus, the consumers and the vendors both have 100 alternatives. A typical situation with $m = n = 5$ (i.e. there are 5 customers who can choose one out of 5 different products) is illustrated in figure 4 , where the o-symbols denote the suppliers and the x-symbols the customers.

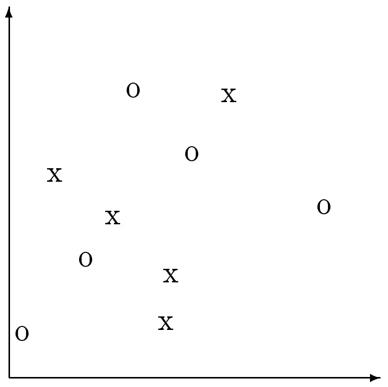

Figure 4: A typical market situation

## The market mechanism

After all the customers have declared their wishes and all the suppliers have made their offers, the customers choose those products with the smallest

[1]In most markest it holds that $m \gg n$.

euclidian distance between the ideal product and the actual offers. In case of two or more products with the same distance one of them is chosen randomly. If we denote $\vec{x^i}$ the $i-$th customers wish and $\vec{y^j}$ the $j-$th vendors offer, then the decision $a^i$ of the $i-$th customer might be

$$a^i = \arg \min_{j \in \{1 \dots n\}} \left\{ \|\vec{x^i} - \vec{y^j}\|_2 \right\}. \tag{1}$$

To keep the model simple we make the following assumptions:

1. All the customers have the same purchasing power.
2. All the producers have the same internal cost structur.
3. Whenever a product is chosen the vendor receives a fixed profit $p$.
4. Each product in the set $\{1, \dots, 100\}^2$ causes the same costs.
5. All the producers are capable to manufacture each possible product in $\{1, \dots, 100\}^2$.

Thus, $\pi^j$ the profit of supplier $j$ within one particular time period equals $p$ times the number of customers who decided to buy the product offered by the supplier $j$, i.e. $\pi^j = p\#\{i|a^i = j\}$.

## The buying agents

At the initial state of our simulation the preferences of the customers are placed randomly somewhere in the twodimensional set. Then, their movement follows a random-walk with small steps. Both components may be increased or decreased by one. Thus, the demand side of the market is not completely static, but changes slightly. If the changes were too big, it would not make sense for the suppliers to build decision rules based on their experience.

In mathematical terms we can say that the demand side is represented by a $2 \times m$ matrix

$$D = \begin{pmatrix} x_1^1 & x_1^2 & \dots & x_1^m \\ x_2^1 & x_2^2 & \dots & x_2^m \end{pmatrix}.$$

The initial state $D_0$ is a $2 \times m$ matrix with all the components uniformly distributed on the set $\{0, \dots, 100\}$. The state $D_t$ at any time $t$ can be given as

$$D_t = D_{t-1} + \Delta D_t, \tag{2}$$

where $\Delta D_t$ is a $2 \times m$ matrix with all its components uniformly distributed on the set $\{-1, 0, 1\}$.

### The selling agents

Like before we use a uniform random distribution to define the initial offers of all the suppliers, and collect the data in the $2 \times n$ matrix

$$S = \begin{pmatrix} y_1^1 & y_1^2 & \cdots & y_1^n \\ y_2^1 & y_2^2 & \cdots & y_2^n \end{pmatrix}.$$

We will observe two classes of agents at the supply side of our artificial market.

### First class agents

The first class of selling agents are using classifier systems like described in chapter 1. We use CSs with a condition part containing at least three and at most five conditions. In the following we use the variable $n_{cond}$ to refer to the number of conditions. The incoming messages are the data about the offered products of all the sellers in the last period plus an additional bit which contains $\pi^j$, the recent success of the offers. Thus, $M_t$ the list of incoming messages at time $t$ becomes

$$M_t = \begin{pmatrix} y_{1,t-1}^1 & y_{2,t-1}^1 & \pi_{t-1}^1 \\ \vdots & \vdots & \vdots \\ y_{1,t-1}^n & y_{2,t-1}^n & \pi_{t-1}^n \end{pmatrix}. \qquad (3)$$

Certainly it does not make sense to make a decision based on the information about only one competitor. Therefore, in this model the rules contain several condition parts and, hence, only those rules are fulfilled that are activated by several suppliers.

### The genetic algorithm

Genetic algorithms (GA) basically have been created to optimise technical systems. Later on, it turned out that they might also be of interest in modeling human behaviour. This is due to the strong analogy between the genetic operators and human learning by trying, experiencing, and imitating. This makes them a powerful tool to simulate social interaction. The main contents of GAs are

- selection,
- replication,
- recombination, and
- mutation.

Certainly, it does make a big difference for the individual if the evolution of the system as a whole is caused by biological like selection (i.e. elimination of unsuccessful individuals) or by learning. In opposite to GAs, general learning processes are usually described by

- variation,
- satisfaction, and
- imitation.

A nice distinction between learning and evolution is given for instance by (Brenner, 1998). However, here the GAs are used to force evolution and improvement within the individuals rule-bases rather than within the population of agents itself. The share of those rules that performed well in the past increases, while shares of rules that led to an outcome below the average decreases. Thus, only rules can be eliminated, but not the individual as a whole. Therefore the changes in the population of agents takes place due to learning effects rather than selection among individuals.

In general a GA works on a population of strings with a fixed length. Also a particular fitness value has to be assigned to each string of the population. The strings can be binary (i.e. only containing 0 and 1), integer, or real valued. Nevertheless, the whole population of strings must be of the same type.

In our model we use the GA to update the rule bases of the first class agents. Therefore, the population contains row vectors of length $3n_c$ because we consider two attributes of the products plus the profit in the last period of $n_{cond}$ different suppliers.

Depending on the kind of data to be used the genetic operators differ sligthly. In the following we will have a closer look to the three main steps of genetic algorithms. A more comprehensive description can be found in (Goldberg, 1989) or (Holland et al., 1997).

## Selection

The selection operator has a very high influence on the dynamics of the population. It is used to determine which individuals' offsprings may occur in the next generation, and which get discarded. In the present model we used a ranking procedure. First, the rules have to be ordered according to their fitness values. Then, those rules belonging to the best 80% are selected, and the others discarded. Finally, those rules belonging to the best 20% are written into the list a second time. This increases the chance of the very successful rules to remain in the rule base of the next time step.

## Crossover

After selecting the rules we produce offsprings by either copying the rows of the present rule-base into the new one, or by combining two rules. First we build pairs of rules randomly. After that with a probability of $\chi = 0.5$ we create new rules by combining the strings, otherwise both strings remain.

### Mutation

At the beginning of the iteration process it is very important to avoid striving to a local optimum. Therefore, a mutation operator is used to place random numbers somewhere into the population. This happens with a probability of $\mu$, which we assigned the values 0, 0.001, and 0.002. In order to control the strictivity of the rule-base we use another mutation operator, which only writes don't care symbols (#) into the condition part of the rules. This is done with a probability of $d_p$.

### Second class agents

The second class of selling agents are the simpler ones. They just make small random movements like the buying agents in the previous section. The purpose of these agents is just to find out if the agents using classifier systems are indeed capable to find intelligent strategies, i.e. to outperform the second-class agents.

If we have $n^1$, the number of first-class agents, and $n^2 = n - n^1$, the number of second-class agents, then the $2 \times n^2$ matrix $\Delta S^2$ with all its components uniformly distributed on $-1, 0, 1$ determines the movements of the second-class agents. The decisions of the agents in class $i$ are collected in the $2 \times n^i$ matrix $S^i$, which leads us to

$$S_t = \left(S_t^1, S_t^2\right) = \left(S_t^1, S_{t-1}^2 + \Delta S_t^2\right). \tag{4}$$

This in turn is the transpose of the first two columns of the matrix $M$ in equation (3) for the next time step $t+1$.

## 3 Simulation Results and conclusions

A typical result is shown in figure 5. We used setups with $m = 50$ customers, $n_1 = 5$ first class selling agents, and $n_2 = 5$ second class selling agents. Assuming $p = 1$, i.e. each sale is worth one monetary unit, the total sales in the market add to $P = m\,p = 50$.

The plot in the first row on the left-hand side compares the average profits of the first- and second class agents. At the begining of the simulation the market shares differ only slightly, but in the sequel the share of the intelligent agents increases. Finally, after 24 generations, the first class agents occupy the whole market, and their average profit becomes $P/n_1 = 10$. This shows that the learning process induced by the GA succeeded in producing useful rules. Moreover, we can conclude, that a decision need not to be based on all the information available. In the present simulation illustrated in figure 5 the CSs contain 3 conditions. Thus, the decisions taken by the first class agents are based only on information about 3 different suppliers.

On the right-hand side we see the frequency of don't care symbols (#) in the CSs of the first class agents. At the initial state only about 40% of

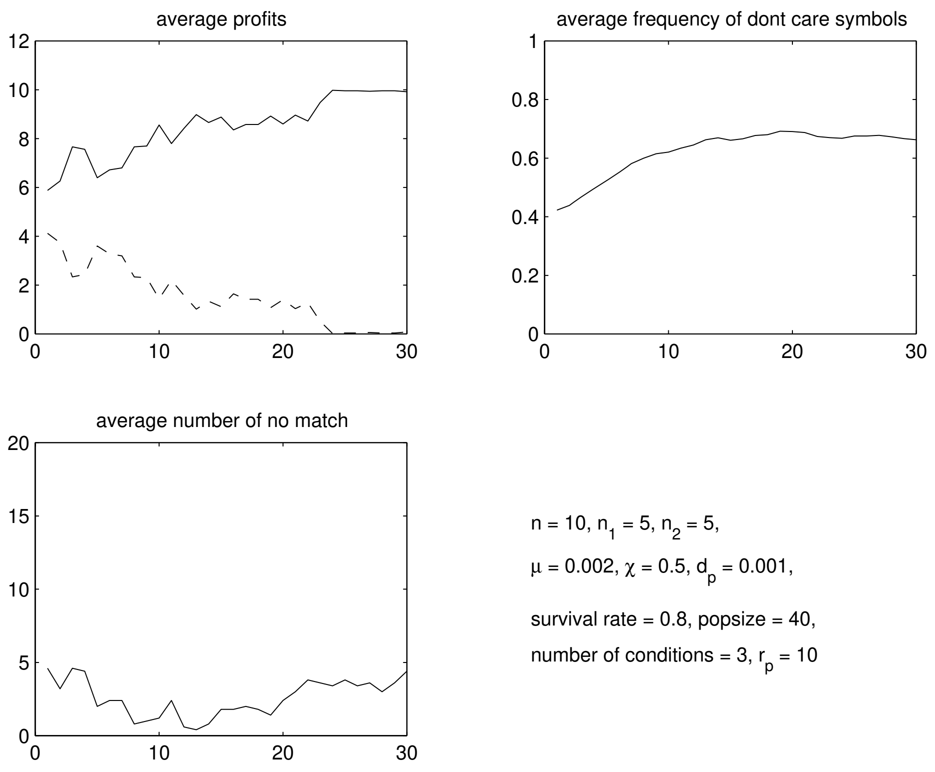

Figure 5: simulation results

the entries in the condition parts were #, but later on their share increased significantly until it stabilized slightly above 60 %. Thus, we can conclude that the very general rules were the more successful ones, and thus survived the selection process.

The graph in the second row shows how often it happend that some of the first class agents had no rule fulfilled. In that case their decision had to be taken randomly. This curve always remains between 0 and 5. This means that at most 16% of the agents using CSs did not receive an appropriate input and, thus, had to guess what to do.

In total we ran 650 experiments. In table 1 all those parameters are listed, whose value was fixed in all the simulations.

In table 2 all those parameters are listed, which got assigned different values. In some of the experiments the general mutation operator also could produce #, thus leading to the expressions $f(\mu)$ in the table.

In all the experiments the intelligent agents (i.e. those using classifier systems rather than random walk) outperformed their competitors. In further research we would like to examine what happens if different types of intelligent agents compete in the market. First of all we think about different parameter settings for the kind of agents described in the present paper. Moreover, one might also think about implementing completely different strategies. For instance, it may yield interesting results to let one agent

| | | |
|---|---|---|
| $\chi$ | crossover probability | 0.5 |
| $m$ | number customers | 50 |
| $n$ | number of suppliers | 10 |
| $n_1$ | number of $1^{st}$ class agents | 5 |
| $n_2$ | number of $2^{nd}$ class agents | 5 |
| $P$ | cumulated profit | 50 |
| $p$ | profit obtained when selling to one customer | 1 |
| $s_{rate}$ | survival rate used by the selection operator | 0.8 |
| $T$ | number of iterations | 30 |

Table 1: fixed parameters

| | | |
|---|---|---|
| $\alpha$ | factor of fitness updates | 0.1, 0.2 |
| $d_p$ | probability of # used by the mutation operator | $0, 0.001, f(\mu), 0.001 + f(\mu)$ |
| $\mu$ | mutation probability | 0, 0.001, 0.002 |
| $n_{cond}$ | number of conditions in the CSs | 3, 4, 5 |
| $p_{size}$ | size of the populations of rules in the CSs | 40, 80, 160 |
| $r_p$ | number of repetitions before invoking the GA | 5, 10, 20 |

Table 2: variable parameters

just imitate the decision of the most successful supplier in the previous time step. Another interesting extension could be achieved by having some superior agents who can even observe the customers' wishes directly. Up to now we assumed, that this information is not available. Thus, the suppliers just could guess the consumers' preferences by observing their buying habits.

## References

Beinhocker, E. D. (1997). Strategy at the edge fo chaos. *The McKinsey Quarterly*, pages 24–39.

Brenner, T. (1996). Learning in a repeated decision process: A variation-imitation-decision model. Papers on Economics & Evolution #9603, Max-Planck-Institute for Research into Economic Systems.

Brenner, T. (1998). Can evolutionary algorithms describe learning processes? *Journal of Evolutionary Economics*, 8:271–283.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Company.

Holland, J. H. (1976). Adaption. In *Progress in Theoretical Biology IV.* ed. Rosen, R. F. New York: Academic Press.

Holland, J. H. (1995). *Adaption in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* The MIT Press, Bradford Books edition, Ann Arbor. MI.

Holland, J. H., Holyoak, K. J., Nisbeth, R. E., and Thagard, P. R. (1997). *Induction, Processes of Inference, Learning, and Discovery.* The MIT Press, Cambridge, Massachusetts, London, England.

Kotler, P., Armstrong, G., Saunders, J., and Wrong, V. (1996). *Priciples of Marketing, The European Edition.* Prentice Hall Europe, Campus 400, Maylands Avenue, Hemel Hampstead, Herfordshire, HP2 7EZ.

Polani, D. and Uthmann, T. (1999). Dmarks: Eine verteilte umgebung für agentenbasierte simulationen von marktszenarien. In Hohmann, G., editor, *Simulationstechnik, 13. Syposium in Weimar*, volume 3 of *Frontiers in Simulation*, pages 391–394. SCS - The Society for Computer Simulation International in cooperation with ASIM - Arbeitsgemeinschaft Simulation.