Agents that Have Desires and Behave Adaptively

Cheng-Yuan Liou*, Chung-Hao Tan, Hwann-Tzong Chen, and Jiun-Hung Chen
Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan, R.O.C.
Tel: 8862-23625336 ext. 515, Fax: 8862-23628167
*correspondence cyliou@csie.ntu.edu.tw

Abstract

An agent which has desires is an autonomous agent which gives substantial attention to its character, and it will behave much like a human being. We implement such agents using an internal ethologically-inspired action selection model. The parameters of this model are adaptively tuned by a stocastic cellular automata learning algorithm. We examine these agents by observing their negotiating behaviors in a multi-agent electronic marketplace.

Keywords: agents, electronic marketplace, stochastic cellular automata, action selection model.

1 Introduction

Autonomous agents have been applied in many areas. An agent is called autonomous if it decides by itself how to relate its sensor data to motor commands in such a way that its goals are attended to successfully [12]. We would expect an autonomous agent to be adaptive, robust, and effective in its dynamic environment. An agent which has desires is an autonomous agent which not only emphasizes the properties that an autonomous agent cares about, but gives much attention to its own character. This kind of agent can be viewed as a semi-creature which has emotions, motivation and belief with regard to the external world. It will behave much like a human being. Our goal is to build such an agent.

Each of our agents has two main parts. One is an internal ethologically-inspired action selection model [3] which makes the agent effective and life-like. The other is a cellular automata which gives the agent learning ability. The architecture of this agent is shown in Figure 1. The Sensor module collects external world information, filters out noises, and sends useful information to the Behavior module. The Behavior module then determines an appropriate action among possible candidates. The Motor Controller module executes the action to react to the external world and the agent itself. The cellular automata assists the agent by tuning the parameters in the action selection model. It uses a reinforcement learning mechanism to learn the feedback from the environment.

We expect that by simulating the behaviors of human beings, an agent can handle more sophisticated problems. We believe this is reasonable because, if an agent can simulate the behaviors of people, then the agent might be able to solve the same problems that people can solve, at least to a significant degree. However, in addition to simulating behaviors, modeling internal characteristics is important. For example, if Bob eats a piece of pizza, he will be more satisfied if he is hungry than if he is satiated. If we want to simulate the above situation, we will have to deal with two things. The first one is that, there must be a way to meansure the degree of hunger(internal state); the second is that, under different internal states, the value of the pizza(belief to external world) may be different. Many behaviors of human beings show such properties. People also have motivation to do something.

Therefore, we must design this agent so that it will possess many personal characteristics. This agent will have mental states like emotions, motivation and beliefs with regard to the external world. It might be happy if it finishes a task on time. It might be impatient if it spends too much time doing the same thing. This agent selects its action according to its internal mental state and external stimuli. For example, if the agent is hungry, there will be more likely to select a feeding action than a sporting action. In the following section, we will discuss the action selection model used by the desired agent. A detailed example to explain the idea behind this agent will be given in section 3. We will develop a mulit-agent electronic marketplace, put the desired agent into the marketplace, and observe its behavior. In section 4, we will further equip the desired agent with a stocastic cellular automata(SCA) learning mechanism so that it can learn from experience.

2 Action Selection in the Desired Agent

As opposed to the traditional 'knowledge-based AI' technique, the 'behavior-based AI' [12] has become a useful approach to building an autonomous agent. Because we want this agent to behave more like a human being, it is reasonable to use an action selection model which is borrowed from ethology. Many researchers have studied this kind of model to imitate the behaviors of animals or have applied it to other applications like robot control. For example, Reynolds [15] modeled the flocking behavior of birds and the schooling behavior of fishes. Blumberg [3] proposed a computational model which sucessfully models an artificial dog with life-like behaviors [4]. Terzopoulos [16] modeled the mental states of fishes using a set of mathematical equations. Tyrrell [18] wrote a comparative comment on some early works and argued that an information free-flow hierarchy is a sensible approach to action selection.

Among these different action selection models, we use Blumberg's model as the core of action selection for the desired agents. We further modify his model by adding branching instructions to the algorithm. This makes action selection effective in critical situations. We will first present Blumberg's model, discuss the aspects of ethology in his model, and then explain our modification.

The computational model presented here originated from Blumberg's ideas [3][4]. Figure 2 shows the architecture of the model. The Internal Variable(IV) is used to model the internal states of the agent. This value can change over time based on growth rates, damping rates, and the value of the relative gain of active behavior. The Releasing Mechanism(RM) is used to sense the external stimuli and output a continuous value which has common currency with the Internal Variable. The Level of Interest is used to model the temporal apects of animal behaviors. When a behavior is active, the value of the Level of Interest of this behavior decreases, which in turn reduces the value of this behavior such that a non-active behavior has a chance to be executed. The Level of Interest prevents the agent from pursuing a single behavior all the time. The Behavior module is used to decide on an appropriate behavior that the agent will execute. Behaviors are organized into groups. Different groups are organized into a hierarchical architecture. Behaviors in the same group are also organized into a loose hierarchical architecture. The upper level behaviors are more general and abstract whereas the lower level behaviors are more specific. Each behavior in the same group mutually inhibits others such that there is only one active behavior at

a time. With higher inhibition gains, the active behavior shows more persistence. With lower gains, the system tends to dither. 'Loose hierarchical' means a losing behavior could recommend to the winning one. Finally, the Behavior module outputs a primary behavior together with a second behavior if there is one.

There are some implications related to ethology [3]. First of all, animals do not dither among multiple activites; they engage in one behavior at a time. Animals behave in a form of time-sharing. Low priority behaviors can be executed while high priority behaviors will not be interrupted arbitrarily. Second, a hierarchical architecture is one of the essential organizing principles of animals. It is easy to scale up and provide a focus of attention to avoid processing irrelevant information. Third, an animal's response to an external stimulus depends both on the strengh of the stimulus and on its internal need so that a continuous value of response and common currency between the internal need and external stimulus are needed. Following are the life-like equations proposed by Blumberg [4]:

- **Internal Variable (IV) Update Equation**

$$iv_{it} = (iv_{i(t-1)} \cdot IVdamp_i) + IVgrowth_i - \sum_k Gain_{ki} \cdot b_{k(t-1)},$$

where iv_{it} is the value of Internal Variable i at time t. $iv_{i(t-1)}$ is its value at the previous time step. $IVdamp_i$ and $IVgrowth_i$ are damping rates and growth rates related to this Internal Variable i. $Gain_{ki}$ is the gain used by Behavior k related to Internal Variable i. $b_{k(t-1)}$ is the value of Behavior k at time $t-1$.

- **Level of Interest (LI) Update Equation**

$$li_{it} = Clamp(((li_{i(t-1)} \cdot LIdamp_i) + LIgrowth_i - (b_{i(t-1)} \cdot bRate_i)), 0, 1),$$

where li_{it} is the value of the Level of Interest related to Behavior i at time t. $LIdamp_i$ and $LIgrowth_i$ are damping rates and growth rates related to this Level of Interest i. $bRate_i$ is the boredom rate for Behavior i. $Clamp()$ clamps the value to the range 0 to 1.

- **Releasing Mechanism (RM) Update Equation**

$$rm_{it} = Clamp(TemporalFilter(t, rm_{i(t-1)}, Stimulus_{it}), min_i, max_i)$$

$$Stimulus_{it} = Find(s_{it}, dMin_i, dMax_i) \cdot Filter(s_{it}) \cdot Weight(s_{it}, Opt_i),$$

where rm_{it} is the value of Releasing Mechanism i at time t. s_{it} is the object in which RM i is interested. $dMin_i$ and $dMax_i$ are minimum and maximum distances for search. $Find()$ returns 1 if s_{it} is found within $dMin_i$ to $dMax_i$; otherwise, it returns 0. $Filter()$ returns 1 or 0 if s_{it} satisfies some conditions. $Weight()$ weights s_{it} according to a weighting function and the distance to an optimal value Opt_i. $TemporalFilter()$ filters the rm_{it} over some period t. $Clamp()$ clamps rm_{it} to the range min_i to max_i.

- **Behavior (B) Update Equation**

$$b_{it} = Max\Big[\Big(li_{it} \cdot Combine(\sum_k rm_{ki}, \sum_j iv_{jt}) - \sum_m ig_{mi} \cdot v_{mt}\Big), 0\Big],$$

where $Combine()$ is a function used to combine the values of the Releasing Mechanisms and Internal Variables. ig_{mi} (ig>1) is the Inhibitory Gain that Behavior m applies against Behavior i. v_{mt} is the value of Behavior m, where m ranges over all other Behaviors in the same Behavior Group.

Note that the parameters $iv_{i(0)}$, $IVdamp_i$, $IVgrowth_i$, $Gain_{ki}$, $b_{k(0)}$, $li_{i(0)}$, $LIdamp_i$, $LIgrowth_i$, $bRate_i$, $rm_{i(0)}$, min_i, max_i, $s_{i(t)}$, $dMin_i$, $dMax_i$, and ig_{mi} should be given in advance. They are initial parameters.

The execution sequence of this algorithm is: (1)update the Internal Variable; (2)update the Releasing Mechanism; (3)update the Level of Interest; (4)update the Behavior; (5)if more than one behavior in the same behavior group has a non-zero value, repeate (4) by post-inhibiting values.

We add branching instructions to the original action selection model (see Figure 3). From our experience in building this agent, we know that there are situations where the agent has to execute a particular action without consideration. For example, a negotiating action is forced to terminate if the deadline has expired. If we use the original algorithm without any modification, we have to carefully tune the parameters of our model to cover such a particular action. This will be diffcult in such a large scale system. Therefore, adding branching instructions in the right situations might be a good choice. This simplifies decision-making in crutial situations without losing the merit of the original model. This can be viewed as pre-programmed conditional action selection.

The execution sequence of the modified algorithm is: (1)update IV; (2)update RM; (3)update LI; (4)if the agent and the environment satify some conditions, (4a)a specified action will be active; otherwise,(4b)update B until only one behavior is active.

3 An Electronic Marketplace Experiment

Many electronic marketplaces have been implemented where people can buy or sell goods. Many of these systems use a 'conditions matching' strategy to find an appropriate product. A Buyer/Seller gives the server a set of condition rules, like a price, properties, or a deadline for the deal. The server searches the answer from its database or makes queries to other servers, and then lists matching answers for the users. BargainFinder [2], Jango [8], and Buy&Sell Online [6] are examples.

Agent-mediated negotiation is a new research approach to electronic commerce (e.q., ActionBot [1], Kasbah [7], and Tete-a-Tete [17]). In this kind of system, agents automate the process of determining prices. The benefit in using this technique is that people could save time and money in real-world negotiation, such as at a stock market or flea market, or in price haggling. Chavez's Kasbah [7] is an interesting experiment. He proposed an agent marketplace for automating the process of negotiating prices. A user in this marketplace can create a particular type of agent. This agent will automatically negotiate with other agents in the same marketplace using a simple and flexible strategy. A buying/selling agent will ask/offer different prices over time.

In this section, we will present an agent-based electronic marketplace and the desired agents for negotiation. These agents will negotiate with each other in a human-like way. To our knowledge, few systems have been implemented in this way. The experiment reveals many life-like negotiating phenomena.

Market Architecture

Our electronic marketplace is a client-server architecture, shown in Figure 4. Client users on-line operate their agents through a web-based user interface. All agents are situated on the server-side. The server chooses two waiting agents to negotiate and notifies remote users of the results.

The operating procedure is as follows: A user first opens an account on the server. The next step is to create a personalized agent by specifing a particular negotiating strategy (shown in Figure 5). He can

assign a task to his agent, that is, specify what kind of product he wants to buy/sell and the best price and so on. The agent will be free to the marketplace and will negotiate with other agents. It will raise or lower prices over time according to its strategy. It can perform other kinds of actions like threatening and dickering to help it get a good price. It will continue to negotiate with another agent until both agents agree on the same price.

The current version of the desired agent in the electronic marketplace experiment has seven Behaviors: 'raise my price', 'lower my price', 'do nothing', 'dicker', 'threaten', 'leave', and 'make a deal'. Also, it has two Internal Variables: the desire for price and the desire for time. The former means how much the agent cares about price. With a high value of desire for price, the agent will spend as little money as possible. On the other hand, the later means how much the agent cares about time taken in negotiation. With a high value of desire for time, the agent wants to make a deal as quickly as possible. There are five Releasing Mechanisms: 'the difference in the prices that the two agents offer', 'the difference in price between the current price an agent offers/asks and the optimal price it expects', 'the residual time before deadline', 'the dicker signal', and 'the threaten signal which detects the relative actions that the other agent performs'.

Experiment Result

An experiment result is shown in Figure 6. There are ten charts, divided into three rows. In each chart, the horizontal axis is the time step. The vertical axis is the value of each parameter which has been normalized from 0 to 1. The four charts from left to right in the first row indicate the IV curves, RM curves, LI curves, and Behavior curves of Agent A, respectively. The charts in the third row indicate those for Agent B. The left chart in the second row indicates the fluctuation of price; the right chart indicates the degree of impatience. The number below 'Deal Price' indicates the price at which these two agent make a deal; the number below 'Tick' indicates the time that this negotiation process takes. In the IV chart, curve **M** is the desire for money, and curve **T** is the desire for time. In the Behavior chart, the curves **u, s, k, w,** and **d** indicate, respectively, the 'raise my price', 'do nothing', ' lower my price', and 'make a deal' behaviors. The multiple curves in the RM and LI charts indicate different parameters, but we will not interpret them in order to reduce the length of this paper.

We can explain Figure 6 from at least two viewpoints. From the viewpoint of internal variables, for Agent A, the value of **T** is always much higher than the value of **M**. This indicates that Agent A wants to save time more than money. It will buy/sell the goods as soon as possible, without considering the price it offers. On the other hand, Agent B pays much attention to saving money, but as time passes, it cares about time as much as money.

From the point of view of 'Behaviors', we can easily find that at first, Agent A(Buyer) quickly raises the price it offers, because of its initial interest in saving money. As the interest in price increases, it stops raising the price and chooses to 'do nothing'. Although it performs these actions, it is patient and feels that there is still a lot of time for negotiation. But as time goes on, it dickers; it wants to save both time and money. After dickering, it becomes more impatient and wants to leave. However, the action it chooses at this time is 'do nothing'. 'Wait, then make a decision' might be a good strategy here. For Agent B(Seller), because of its initial interest in saving money, it chooses to 'dicker' in the early stage of the negotiation process. But as time passes, it lowers the price it offers. At last, these two agents agree on a price and make a deal.

Before ending this section, we will note three important phenomena in the behaviors of agents. First of all, we can find that in each time step there is only one active behavior. The second is that the intension to do one behavior decreases gradually. The third is that one behavior will last only for a short period until another behavior wins in its competition with all the others. These life-like phenomena are also found in ethology and generally agreed on by ethologists [3].

4 Stochastic Cellular Automata for Adaptive Behaviors

Blumberg's model is a hand-built, hierarchical network [12]. Developers have to design and adjust parameters by hand. To perform this tricky task, we need a learning mechanism to give the agent a learning ability. Furthermore, it is important to adjust the agent itself in order make it adaptive to changes in the environments. Blumberg proposed a learning algorithm [5] and used it in his action selection model. We, however, use a stochastic cellular automata(SCA) learning algorithm to give the learning ability to the desired agent.

The cellular automata(CA) is a discrete time/space universe and operates using a set of simple local rules. Much work has been done on the theoretical bases of CA. The results show that the cellular automata supports the basic operations of information transmission, storage, and modification in the vicinity of a phase transition [9]. It has been proved that the cellular automata and Turing machine have the same computational power [11]. By building appropriate rules, CA can display many kinds of complex behaviors. The difference between SCA and CA is that SCA updates itself by means of nonuniform local rules.

Definition of Stocastic Cellular Automata

Let SCA = (U,X,Y,Q,N,γ,F,G) be a stochastic cellular automata, where

(i) U: the discrete cellular space,

(ii) X: the set of *input* symbols,

(iii) Y: the set of *output* symbols,

(iv) Q: the set of *state* symbols,

(v) N: the *neighborhood relations* of one cell in U,

(vi) γ: the *neighborhood state configuration* function

$$\gamma : Q \to \otimes_N Q,$$

(vii) F: the *stochastic next state mapping* function

$$F : X \times \otimes_N Q \to Q$$

$$\forall x \in X, \quad \forall n \in N, \quad \sum_{q \in Q} f(x,n,q) = 1, and$$

(viii) G: the *stochastic output mapping* function

$$G : \otimes_N Q \to Y$$

$$\forall n \in N, \quad \sum_{y \in Y} g(n,y) = 1.$$

The state-transition diagram of SCA is shown in Figure 7. x_i and y_i were six-dimentional vectors in all our experiments. x_i, y_i, and q_i are, respectively, the input, output, and state at time step i.

Reinforcement Learning Method

We used the following learning algorithm to train the parameters $iv_{i(t)}$, $IVdamp_i$, and $IVgrowth_i$ of the Internal Variable 'desire for time' and 'desire for money' in the experiment described in section 3. All the other parameters were fixed. This is because training all the parameters was inefficient. We chose these key parameters to enable the agent to learn feedback from the environment.

Following is a reinforcement learning method of the stochastic cellular automata proposed by Lee et al. [10]. It has been demostated by means of a pole balancing experiment [14].

We have two learning strategies. Both learning strategies will converge with the desired behavior of minimizing the penalty:

(i) **only punish** $g(n, y)$ **and** $f(x, n, q)$ **:**

If at time t, the automaton is at state $s_t = s^1$ and the output is $y_t = y^1$, then

- when the reinforcement signal $r_t = 1$ (penalize)

  $$P[y^1 \mid s^1, t+1] = \lambda P[y^1 \mid s^1, t],$$

  where $0 < \lambda < 1$,

  and to preserve normalization, for all other output $y \neq y^1$,

  $$P[y \mid s^1, t+1] = \lambda P[y \mid s^1, t] + \frac{1-\lambda}{n_y - 1},$$

  where n_y is the cardinality of the set of ouput symbols;

- when $r_t = 0$ (not penalize)

  do nothing!

If at time t, the automaton is at state $s_t = s^1$, its previous state is s^0, and the input is $x_t = x^1$, then

- when reinforcement signal $r_t = 1$ (penalize)

  $$P[s^1 \mid x^1, s^0, t+1] = \lambda P[s^1 \mid x^1, s^0, t],$$

  and to preserve normalization, for all other states $s \neq s^1$,

  $$P[s \mid x^1, s^0, t+1] = \lambda P[s \mid x^1, s^0, t]$$

  $$+\frac{1-\lambda}{n_s - 1},$$

  where n_s is the cardinality of the set of state symbols;

- when $r_t = 0$ (not penalize)

  do nothing!

(ii) **reward and punish $g(n, y)$ and $f(x, n, q)$:**

When $r_t = 1$, we solve the above four equations to punish $g(n, y)$ and $f(x, n, q)$; when $r_t = 0$, we can reward them by

$$P[y^1 \mid s^1, t+1] = \lambda P[y^1 \mid s^1, t] + (1 - \lambda)$$

$$P[s^1 \mid x^1, s^0, t+1] = \lambda P[s^1 \mid x^1, s^0, t] + (1 - \lambda).$$

Note that $P[A|B]$ is the conditional probability.

Implementation

We modified the implementation of the pole balancing learning experiment [14] to suit our requirements. The meaning of each variable in our algorithm is explained as follows. X is the set of Internal Variables. S is the set of coarsely quantized states of X in phase space. Y is the set of bounded output values of S. Each value in Y will be added to a corresponding value in X in the next run. Our algorithm, first, sets X and lets an agent negotiate with others based on X. Second, it generates a punishment/reward signal which depends on whether the goal is reached or not. Third, it calculates S by looking up the nearest two neighbors around each cell [14]. Next, it calculates Y by looking up in the CA rule table. Then, it modifies the CA rule table according to a punishment/reward signal. Last, it updates X. Figure 8 showes a flowchart of our algorithm.

Experiment Result

In the two experiments described below, we used 'whether two chosen agents will make a deal or not' as the objective function. According to this function, we could decide when to punish or reward the agents. The first experiment was performed as follows. We chose two agents and let them negotiate with each other. At the beginning, Agent A could not make a deal with Agent B within fifty time steps. But we found that after learning, Agent A could make a deal with Agent B. Note that learning could be processed either online or offline. Figure 9 displays an offline example.

In the second experiment, one agent could not make a deal with any of the four agents in a group. After learning, this agent could make a deal with every one in this group. This experiment shows that this agent had memory in its CA brain. The results are shown in Figure 10, Figure 11, and Figure 12. Note that in these figures, Agent A is the agent which learned. The results shown in (a), (b), and (c) were generated by untrained Agent A negotiating with the other three agents. Those in (d), (e), and (f) were generated by trained Agent A negotiating with the same three agents. The training in this experiment was performed offline.

In this paper, we have presented an implementention of a desired agent. The system includes two main parts: an ethologically-inspired action selection model and a learning mechanism based on a stochastic cellular automata. We also performed three experiments. One was for an agent-based electronic marketplace, and the others tested the learning mechanism. All of these three experiments demostrated the capability of our agents.

References

[1] ActionBot, at URL: http://auction.eecs.umich.edu/

[2] BargainFinder, at URL: http://bf.cstar.ac.com/bf

[3] Blumberg B.M., 'Action-Selection in Hamsterdam: Lesson from Ethology', in *From Animal to Animat, Proceedings of the Third International Conference on the Simulation of Adaptive Behavior*, MIT Press, Cambridge, MA, 1994.

[4] Blumberg B.M., and Galyean T.A., 'Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments', in *Proceedings of SIGGRAPH 95*, 1995.

[5] Blumberg B.M., Todd P.M., and Maes P., 'No Bad Dogs: Ethological Lessons for Learning in Hamsterdam', in *From Animal to Animat, Proceedings of the Fourth International Conference on the Simulation of Adaptive Behavior*, MIT Press, Cambridge, MA, 1996.

[6] Buy&Sell Online, URL: http://www.buysell.com/

[7] Chavez A., Dreilinger D., Guttman R., and Maes P., 'A Real-Life Experiment in Creating an Agent Marketplace', in *Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Mulit-Agent Technology (PAAM'97)*, London, UK, Apr. 1997.

[8] Jango, at URL: http://www.jango.com/

[9] Langton C.G., 'Computation at the Edge of Chaos: Phase Transitions and Emergent Computation', *Physica D*, 42, pp.12-37, 1990.

[10] Lee Y.C., Qian S., Jones R.D., Barnes C.W., Flake G.W., O'Rourke M.K., Lee K., Chen H.H., Sun G.Z., Zhang Y.Q., Chen D., and Giles C.L., 'Adaptive Stochastic Cellular Automata: Theory', *Physica D*, 45, pp.150-180, 1990.

[11] Smith III A.R., 'Simple computation-universal cellular spaces' *J. ACM*, 18, pp. 339-353, 1971.

[12] Maes P., 'Modeling Adaptive Autonomous Agents', *Artificial Life Journal*, Vol.1, No.1&2, MIT Press, Cambridge, MA, 1994.

[13] Maes P., 'Artificial Life Meets Entertainment: Lifelike Autonomous Agents', *Communication of the ACM*, Vol.38, No.2, pp.108-114, 1995.

[14] Qian S., Lee Y.C., Jones R.D., Barnes C.W., Flake G.W., O'Rourke M.K., Lee K., Chen H.H., Sun G.Z., Zhang Y.Q., Chen D., and Giles C.L., 'Adaptive Stochastic Cellular Automata: Application', *Physica D*, 45, pp.181-188, 1990.

[15] Reynolds, C.W., 'Flocks, herds, and schools: A distributed behavioral model', in *Proceedings of SIGGRAPH 87*, pp.25-34, 1997.

[16] Terzopoulos D., Tu X., and Grzeszczuk R., 'Artificial Fishes with Autonomous Locomotion, Perception, Behavior, and Learning in a Simulated Physical World', *Artificial Life IV*, MIT Press, Cambridge, MA, pp.17-27, 1994.

[17] Tete-a-Tete, at URL: http://ecommerce.media.mit.edi/Tete-a-Tete/

[18] Tyrrell T., 'The Use of Hierarchies for Action Selection', in *From Animal to Animat, Proceedings of the Second International Conference on the Simulation of Adaptive Behavior*, MIT Press, Cambridge, MA, 1993.

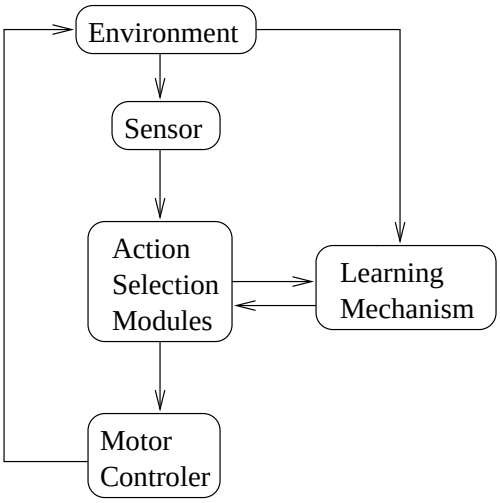

Figure 1: The architecture of the desired agent.

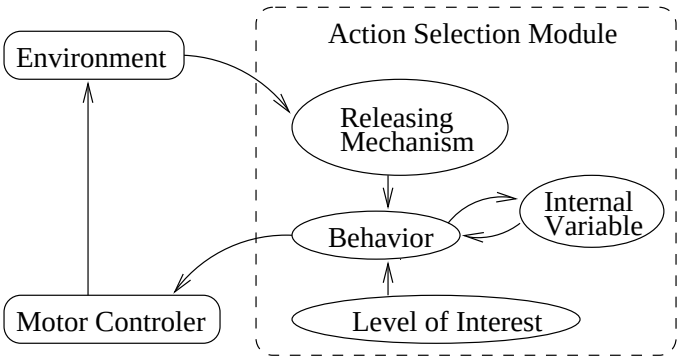

Figure 2: The architecture of Blumberg's model.

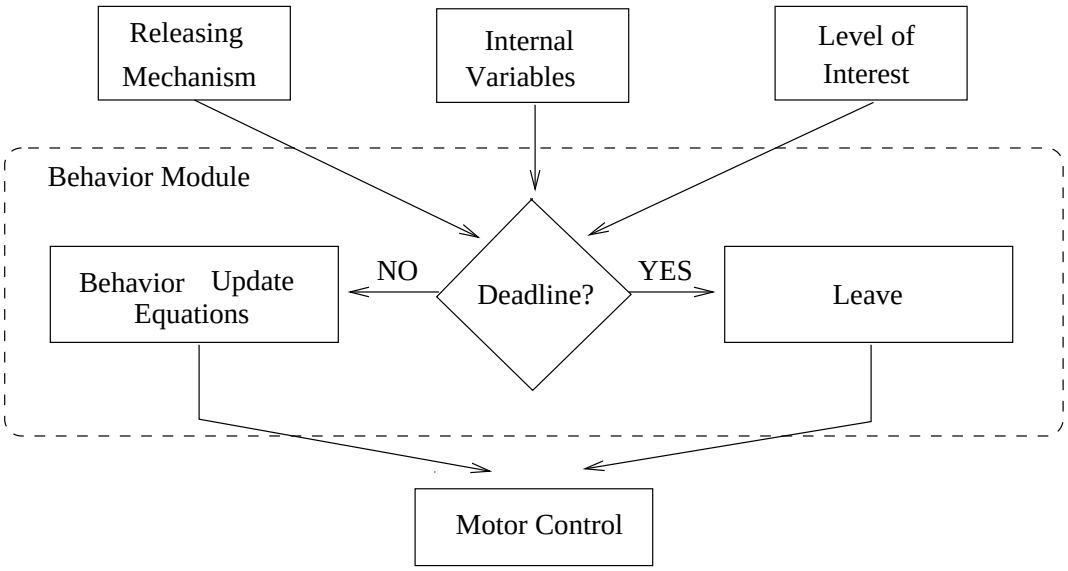

Figure 3: An example of modification of Blumberg's model.

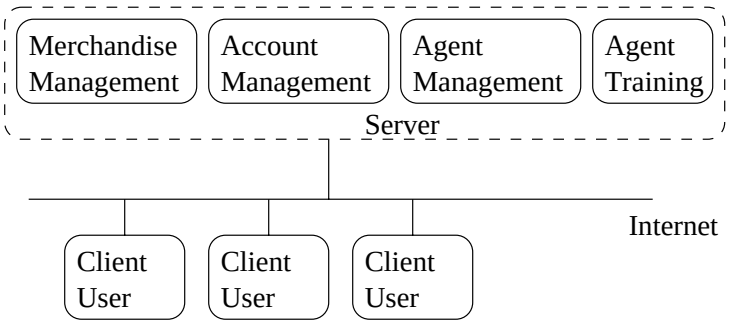

Figure 4: The electronic marketplace.

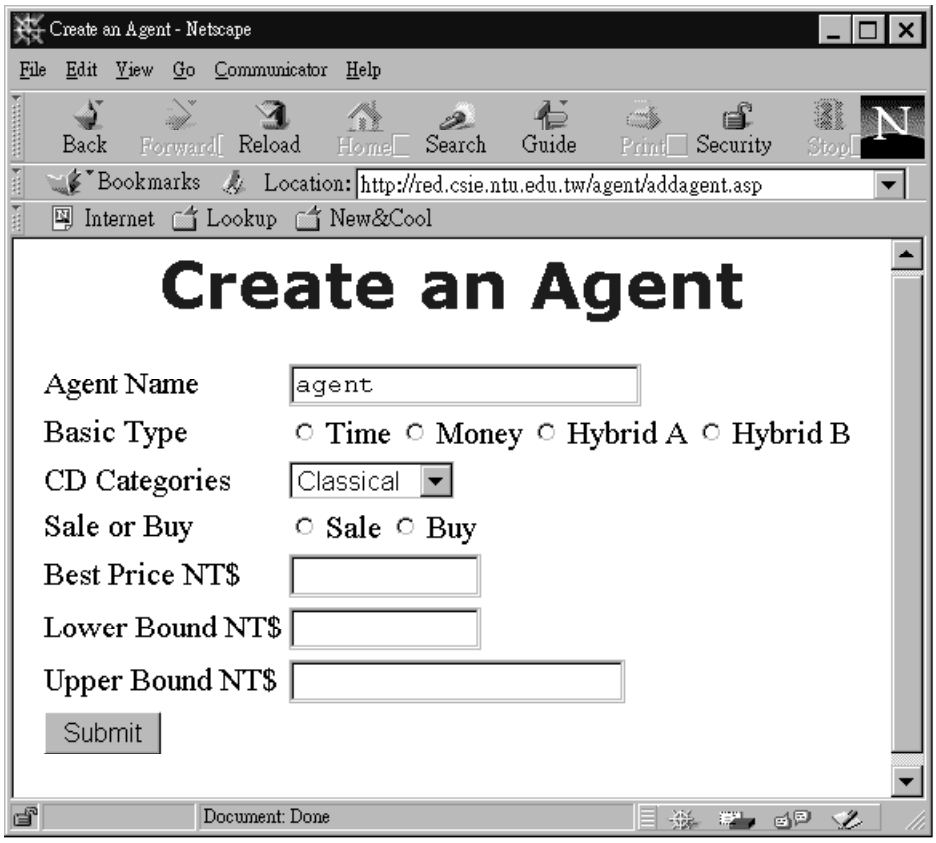

Figure 5: A web-based user interface.

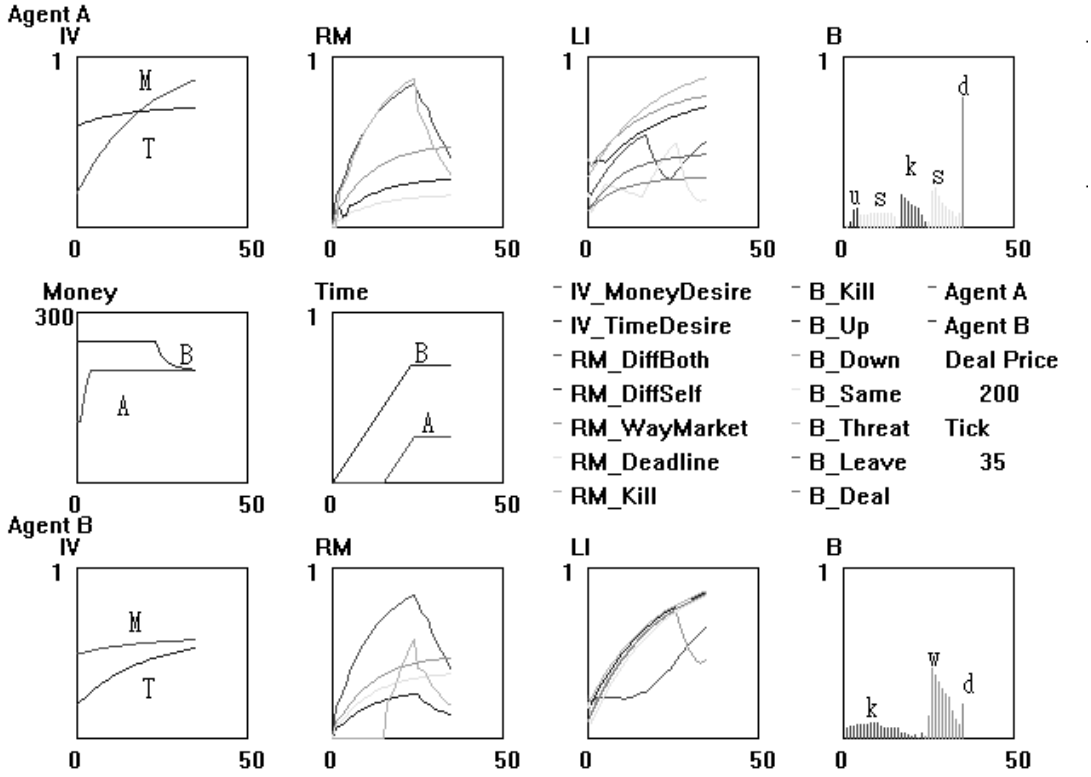

Figure 6: Experimental results of negotiation between two agents.

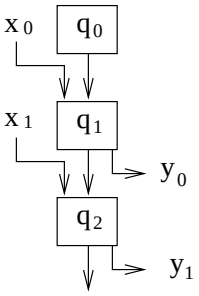

Figure 7: The state transition diagram of SCA.

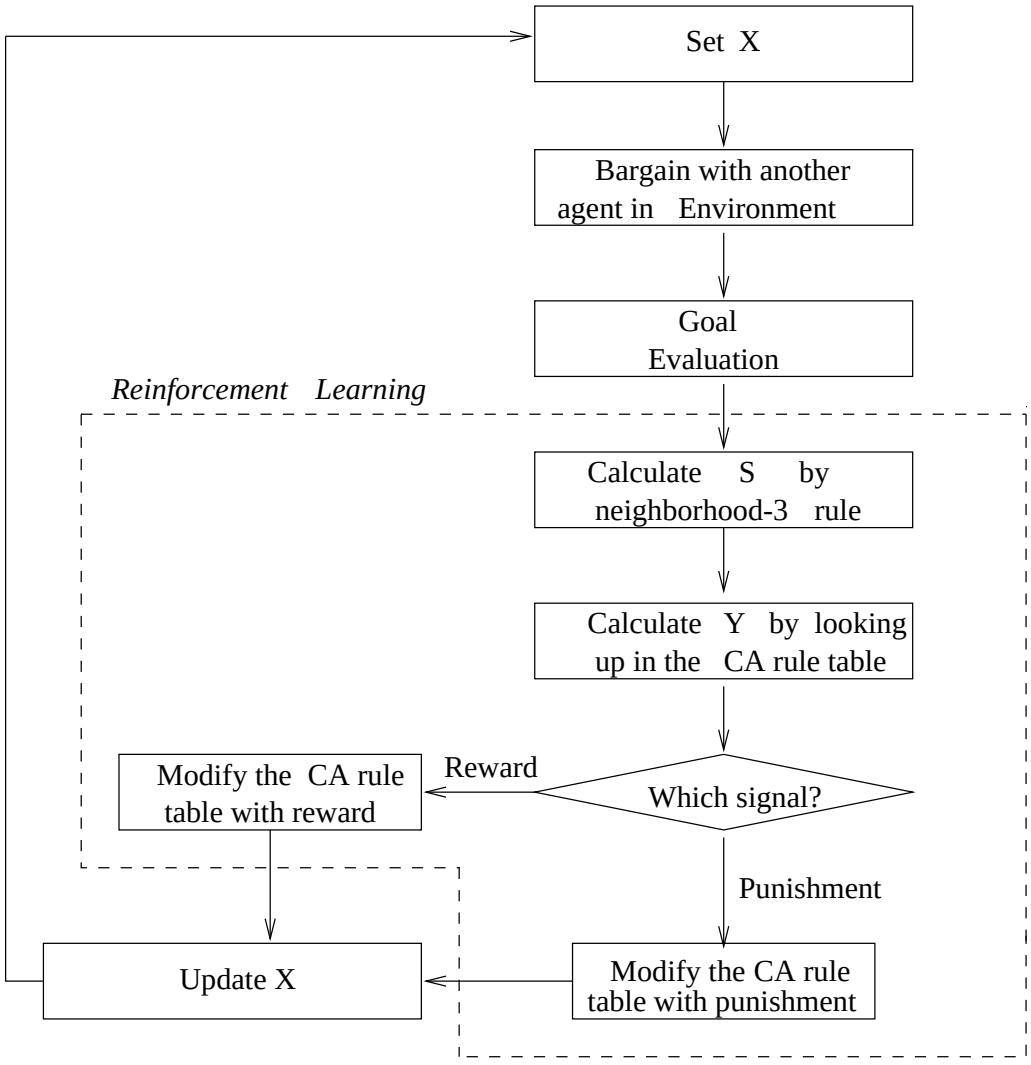

Figure 8: The flowchart of the learning algorithm.

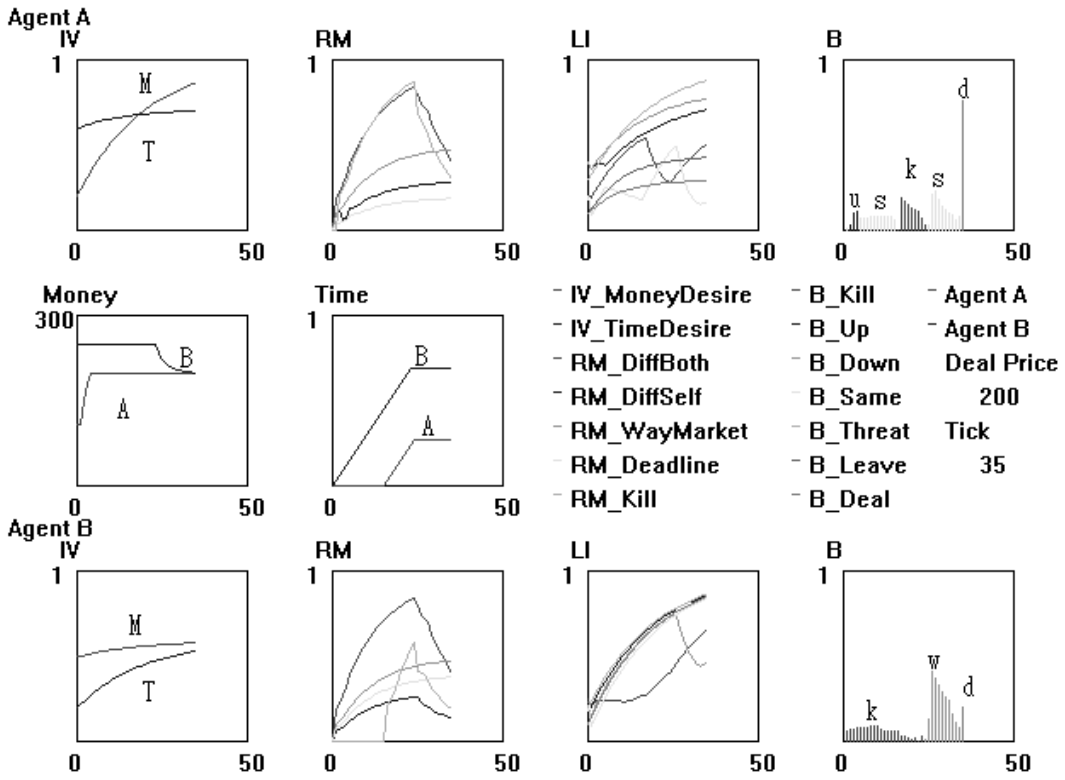

Figure 9: The first set of experimental results : In (a), Agent A is untrained. In (b), Agent A is trained.

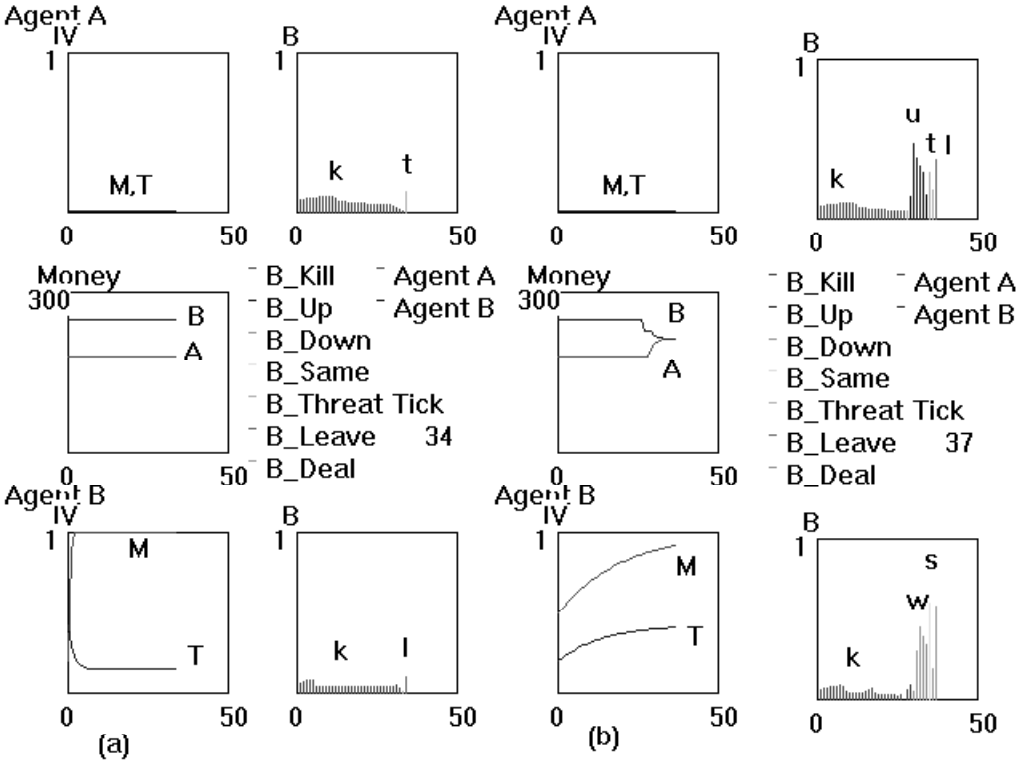

Figure 10: The second set of experimental results, part 1.

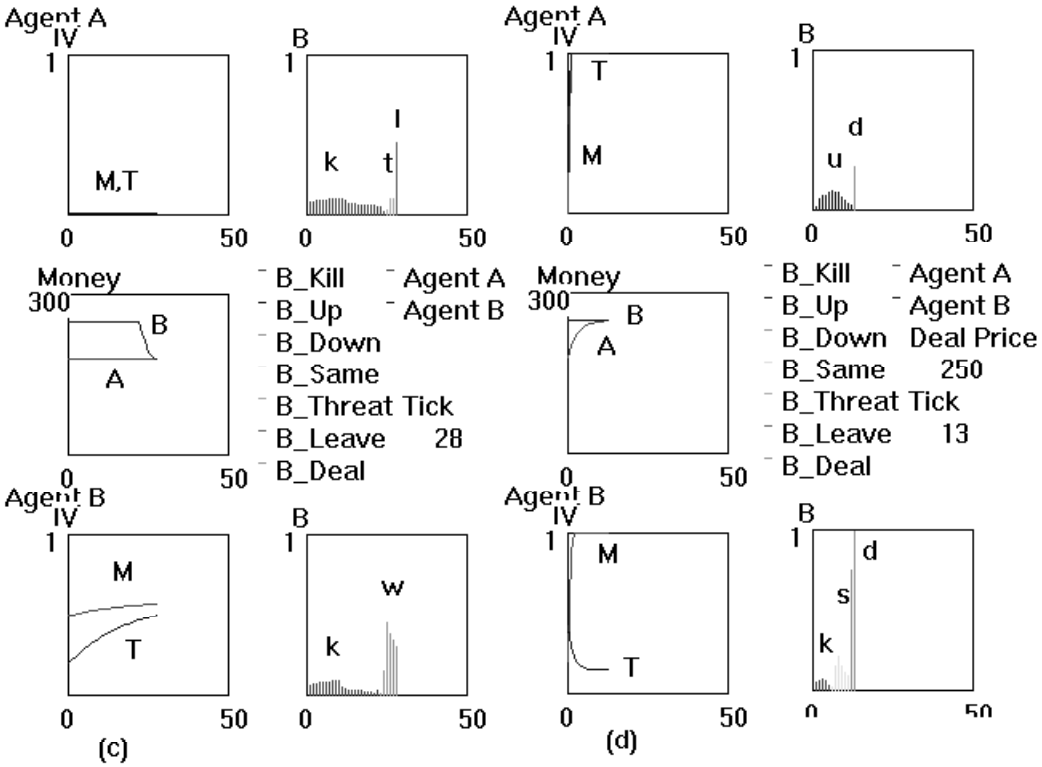

Figure 11: The second set of experimental results, part 2.

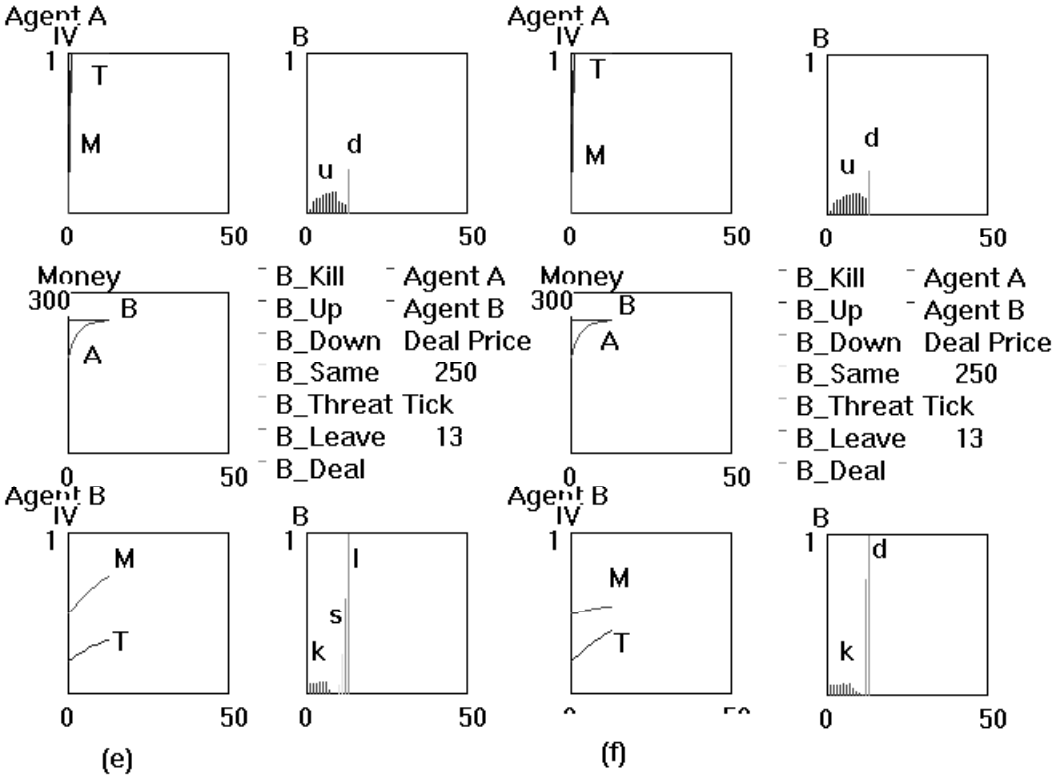

Figure 12: The second set of experimental results, part 3