Designing Decision Making Systems for a Market Selection Game

Hisao Ishibuchi, Chi-Hyon Oh, and Tomoharu Nakashima
Department of Industrial Engineering
Osaka Prefecture University
Gakuen-cho 1-1, Sakai
Osaka 599-8531, JAPAN
{hisaoi, oh, nakashi}@ie.osakafu-u.ac.jp
http://www.ie.osakafu-u.ac.jp/student/ci_lab/ci_lab_e/index.html

Abstract

This paper describes how a decision making system for a market selection game can be automatically designed during the iterative execution of the game. Our market selection game is a non-cooperative repeated game where many players compete with one another at several markets. At each iteration of our game, every player is supposed to simultaneously choose a single market from several ones for maximizing his own profit obtained by selling his product at the selected market. It is assumed in our market selection game that the market price of the product is determined by the demand-supply relation at each market. For example, if many players bring their products to a particular market, the market price at that market becomes low. On the contrary, the market price is high if the total amount of products brought to the market is small. In this manner, the market price at each market is determined by the actions of all players. Each player's profit at each iteration of the game mainly depends on the market price at the selected market. So every player wants to choose a market with a high market price, i.e., a market that is not chosen by many other players. In this paper, we intend to design a decision making system that automatically chooses a single market for a player based on the market prices of all markets at the previous iteration of the game. That is, the inputs to the decision making system are the market prices at the previous iteration, and the output is a single market from which the player is likely to obtain the highest profit at the current iteration of the game. We mainly examine two approaches to the design of decision making systems. One approach is based on a fuzzy reinforcement learning technique called "fuzzy Q-learning". In this approach, knowledge related to the market selection is automatically acquired in the form of fuzzy if-then rules during the iterative execution of the game. The antecedent part of each fuzzy if-then rule is linguistically conditioned by the market prices at the previous iteration (e.g., If the price at Market 1 was *high* and the price at Market 2 was *low*). The consequent part is the expected profit obtained from each market at the current iteration (e.g., then 23\$ from Market 1 and 10\$ from Market 2). The expected profit in the consequent part is automatically adjusted during the iterative execution of the game. In the other approach, the design of a decision making system is handled as a pattern classification problem where a feature vector consists of the market prices at the previous iteration. The class label for the feature vector (i.e., the desired output corresponding to the feature vector) is the market from which the player would obtain the highest profit at the current iteration if he chose that market. That is, the class label is the optimal market for the player at the current iteration (not the actually selected market at the current iteration). It should be noted that the optimal market for the current iteration is known only after all the players made the market selection. In this manner, a single input-output pair (i.e., a single training pattern) is obtained after a single iteration of the game is completed. This means that the available information for the design of the decision making system increases during the iterative execution of the game. By computer simulations, we examine the performance of these two approaches. The main contributions of this paper in comparison with our previous studies are twofold: to propose the handling of the market selection game as a pattern

classification problem and to examine the performance of decision making systems for the market selection game with dynamically changing market conditions.

Key words: Market selection, demand-supply relation, market price, repeated game, decision making, pattern classification, fuzzy rule-based systems, reinforcement learning.

1 Introduction

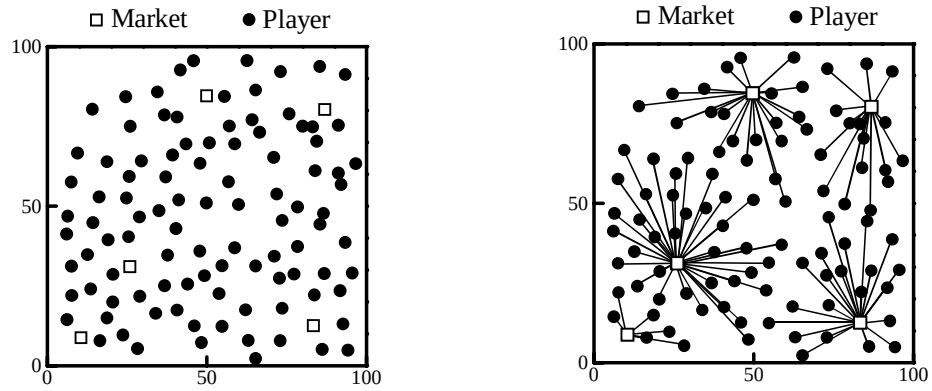
We have already formulated a market selection game as a multi-player non-cooperative repeated game where each of many players (e.g., 100 players) is supposed to iteratively choose a single market from several ones (e.g., five markets) to sell his product [1]. At each iteration of our market selection game, every player simultaneously makes the market selection. The profit of a player at each iteration of the game is defined by the market price at the selected market and the transportation cost to that market. That is, the profit is the difference between the market price and the transportation cost. Whereas the transportation cost from each player to each market is prespecified as a constant, the market price at each market varies depending on the number of players choosing that market at each iteration of the game. If many players choose a particular market, the market price at that market becomes low. On the contrary, the market price is high if only a small number of players choose that market. Thus, for obtaining a high profit, a player has to choose a market that is not chosen by many players. We have examined various strategies for our market selection game such as a random selection strategy, a minimum transportation cost strategy, an optimal strategy for the previous actions, a mimic strategy of the nearest neighbor player, a Q-learning-based strategy, and a fuzzy Q-learning-based strategy [2]. In this paper, we mainly examine two approaches that can automatically design decision making systems for our market selection game in an adaptive manner. One approach is the fuzzy Q-learning-based strategy [1,2]. The other is the handling of the design of a decision making system as a pattern classification problem. The main aim of this paper is to examine the adaptability of these approaches to dynamically changing market conditions. In our computer simulations, the demand-supply relation at each market (i.e., mechanism for determining the market price) and the strategies of other players are changed during the repeated execution of our market selection game. Such situations were not considered in our previous studies [1,2].

2 Formulation of Our Market Selection Game

In this section, we illustrate our market selection game formulated in our previous studies [1,2]. In Fig. 1, we show an example of our market selection game, which is used in computer simulations of this paper. As shown in Fig. 1, many players and several markets are involved in our market selection game. We denote the number of players by n ($n = 100$ in Fig. 1). Each player is indexed by i where $i = 1, 2, \cdots, n$. The number of markets is denoted by m ($m = 5$ in Fig. 1). Each market is indexed by j where $j = 1, 2, \cdots, m$. Our market selection game is iterated for a prespecified number of iterations. Let us denote the total number of iterations by T ($T = 1000$ in our computer simulations). Each iteration is indexed by t (i.e., $t = 1, 2, \cdots, T$). We assume that each player has a single product to be sold at each iteration.

The action of each player at each iteration is to select a single market where his product is sold. An example of such actions of all players is illustrated in Fig. 1 (b). Let us denote the action of the i-th player at the t-th iteration of our game by x_{ij}^t where $i = 1, 2, \cdots, n$, $j = 1, 2, \cdots, m$, $t = 1, 2, \cdots, T$, and

$$x_{ij}^t = \begin{cases} 1, & \text{if the } i\text{-th player chooses the } j\text{-th market at the } t\text{-th iteration,} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$



(a) Locations of 100 players and 5 markets. (b) Example of the actions of 100 players.

Figure 1: Example of our market selection game, which is used in computer simulations of this paper.

Because every player is supposed to choose a single market from the given m markets for selling his product at each iteration of the game, the following relation holds:

$$\sum_{j=1}^{m} x_{ij}^t = 1 \quad \text{for } i = 1, 2, \cdots, n; \ t = 1, 2, \cdots, T. \tag{2}$$

We assume that all players simultaneously perform the market selection at each iteration of our game. Thus no player knows the current actions of the other players when he chooses a market. This means that no player knows the optimal market selection for the current iteration of the game.

We assume that the market price of the product is determined by the demand-supply relation at each market. For example, if many players bring their products to a particular market, the market price at that market becomes low. On the contrary, the market price is high if the total amount of products brought to the market is small. In this manner, the market price at each market is determined by the actions of all players. The total amount of products that are sold in the j-th market at the t-th iteration is calculated from (1) as follows:

$$X_j^t = \sum_{i=1}^{n} x_{ij}^t \quad \text{for } j = 1, 2, \cdots, m; \ t = 1, 2, \cdots, T. \tag{3}$$

We assume that the market price of the j-th market at the t-th iteration is determined by the following linear demand-supply relation:

$$p_j^t = a_j - b_j \cdot X_j^t \quad \text{for } j = 1, 2, \cdots, m; \ t = 1, 2, \cdots, T, \tag{4}$$

where a_j and b_j are positive constants that specify the demand-supply relation in the j-th market. An example of the demand-supply relation is shown in Fig. 2 where the market price is determined by the following relation:

$$p_j^t = 100 - 3 \cdot X_j^t. \tag{5}$$

This relation is used in our computer simulations.

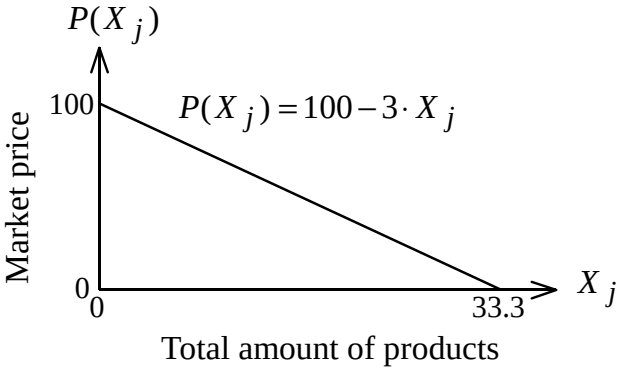


Figure 2: Illustration of the demand-supply relation.

It is assumed that that the transportation cost c_{ij} of the product from the i-th player to the j-th market depends on the distance between the player and the market. Let us denote the distance between the i-th player and the j-th market by d_{ij}. We assume that the transportation cost c_{ij} is given as follows:

$$c_{ij} = c \cdot d_{ij} \quad \text{for } j = 1, 2, \cdots, m; \ t = 1, 2, \cdots, T, \tag{6}$$

where c is the transportation cost for the unit distance. In our computer simulations, we specify the value of c as $c = 1$. In this case, the transportation cost c_{ij} is given as $c_{ij} = d_{ij}$. Since the locations of players and markets are fixed during the iterative execution of the game, the transportation cost c_{ij} is handled as a prespecified constant.

Let us denote the profit of the i-th player at the t-th iteration by r_i^t. We define the profit r_i^t as follows when the i-th player chooses the j-th market for selling his product (i.e., when $x_{ij}^t = 1$):

$$r_i^t = p_j^t - c_{ij} \quad \text{for } i = 1, 2, \cdots, n; \ t = 1, 2, \cdots, T. \tag{7}$$

It should be noted that the profit r_i^t of the i-th player depends on the actions of the other players through the market price p_j^t (see (3) and (4)). The aim of each player in our game is to maximize his own total profit r_i over T iterations:

$$r_i = \sum_{t=1}^{T} r_i^t \quad \text{for } i = 1, 2, \cdots, n. \tag{8}$$

The point in our market selection game is to choose a market that is not chosen by many other players. That is, when the choice of a player is different from many other players, that player obtains a high profit. On the contrary, if a player chooses the same market as many other players, his profit is not high. This is the main characteristic feature of our market selection game. The deterioration of the profit by the same action of many players can be observed in some real- world situations. For example, if many drivers choose the same highway, they will not be able to drive fast due to heavy traffic jams. When only a small number of drivers choose another highway, they may enjoy a comfortable drive.

3 Various Strategies for Our Market Selection Game

We have already examined the performance of various strategies for our market selection game in our previous study [2]. In this section, we briefly describe those strategies.

3.1 Random Strategy

The simplest strategy for our market selection game is a random strategy where a player randomly selects a market from the given m markets. The random strategy can be obtained by specifying the market selection probability $\Pr(x_{ij}^t = 1)$ as

$$\Pr(x_{ij}^t = 1) = 1/m, \ j = 1, 2, \cdots, m. \tag{9}$$

That is, each of the m markets is randomly selected with the same probability. This strategy takes no factors into account when it chooses a market. Thus we are not likely to obtain a high profit from this strategy. We use the random strategy for calculating a base line profit, which are compared with profits obtained by other strategies.

3.2 Minimum Transportation Cost Strategy

Another simple strategy is a minimum transportation cost strategy where a player chooses its nearest market for minimizing the transportation cost. This strategy is very effective when the transportation cost c for the unit distance is large. If we assign the minimum transportation cost strategy to all players in the market selection game in Fig. 1 (a), every player always chooses its nearest market as in Fig. 1 (b). The effectiveness of this strategy depends on not only the unit transportation cost but also the locations of the player adopting this strategy and the given markets.

3.3 Optimal Strategy for Previous Actions

Since every player simultaneously performs the market selection at each iteration of our game, no player knows the current actions of the other players before he chooses a market. This means that no player knows the optimal market selection for the current iteration of the game. Every player, however, can calculate the optimal market for the previous actions of the other players. This strategy, which is referred to as an optimal strategy for the previous actions, is actually optimal only when the other players choose exactly the same markets as in the previous iteration. Of course, the optimal strategy for the previous actions is not always optimal for the current actions because the other players usually change their choices. At the first iteration of our game, a player adopting this strategy chooses the nearest market with the minimum transportation cost because there is no information about the previous actions of the other players. While this strategy is optimal when all the other players adopt the minimum transportation cost strategy, it does not work well when many other players adopt the random strategy (i.e., when many players changes their actions). As we will show by computer simulations later, the performance of the optimal strategy for the precious actions is terribly poor when many players adopt this strategy.

3.4 Mimic Strategy of Nearest Neighbor Player

The optimal strategy for the previous actions requires the information about the demand-supply relations of all the m markets (i.e., a_j and b_j for all the m markets) and the actions of the other players at the previous iteration of the game. When such information is not available, we need simpler strategies that do not require a lot of information. One of such simple strategies is a mimic strategy of the nearest neighbor player where a player simply mimics the previous action of the nearest neighbor player. At the first iteration of the game, a player adopting this strategy randomly chooses a market because there is no available information. Since a player has almost the same transportation cost to each market as its nearest neighbor player, this strategy is promising when its nearest neighbor player is using a good strategy. The profit of a player adopting this strategy totally depends on the performance of its nearest neighbor player.

3.5 Q-learning-based Strategy

The above four strategies do not use any information about the actual profit obtained from each market during the previous execution of the game. In order to choose a market based on the profit that has already been obtained from each market, we can use Q-learning [3], which is a well-known reinforcement learning scheme. In a Q-learning-based strategy, a player stores and updates a Q-value for each market during the execution of our repeated game. The Q-value can be viewed as the expected profit obtained from that market. Let Q_{ij}^t be the Q-value of the i-th player for the j-th market at the t-th iteration of our game. The Q-value for the selected market is modified after the t-th iteration as

$$Q_{ij}^{t+1} = \begin{cases} (1-\alpha) \cdot Q_{ij}^t + \alpha \cdot r_i^t, & \text{if } x_{ij}^t = 1, \\ Q_{ij}^t, & \text{otherwise}, \end{cases} \tag{10}$$

where α is a positive learning rate. In our computer simulations, α is specified as $\alpha = 0.9$, and the initial value of each Q-value is specified as $Q_{ij}^1 = 100$. As shown in (10), only the Q-value for the selected market is updated. That is, the update of Q-values is based on the actually obtained profit at each iteration of our game.

For obtaining a high profit, it is a natural idea to select the market with the highest expected profit (i.e., highest Q-value) at each iteration of our game. On the other hand, it is also possible that we may obtain a high profit from a rarely selected market with a low Q-value in dynamically changing market conditions. That is, a profitable market after the change of the market conditions may be hidden by a low Q-value caused by the low profitability of the market before the change. For searching for such a hidden market, it is required to select not only the market with the highest Q-value but also the other markets. In the Q-learning-based strategy, the market selection is probabilistically performed based on the Q-value for each market. The selection probability $\Pr(x_{ij}^t = 1)$ of each market is defined by the roulette wheel selection with the linear scaling as follows:

$$\Pr(x_{ij}^t = 1) = \frac{Q_{ij}^t - \min\{Q_{ij}^t\}}{\sum_{j=1}^{m} (Q_{ij}^t - \min\{Q_{ij}^t\})}, \text{ for } i = 1, 2, \cdots, n, j = 1, 2, \cdots, m, t = 1, 2, \cdots, T, \tag{11}$$

where $\min\{Q_{ij}^t\} = \min\{Q_{ij}^t | j = 1, 2, \cdots, m\}$.

3.6 Fuzzy Q-learning-based Strategy

The Q-learning-based strategy only uses the actually obtained profits during the previous iterations of our repeted game. Other information is not utilized in the market selection by the Q-learning-based strategy. Since the profit directly depends on the market prices, they seem to be very important information in the market selection. In an extended version of the Q-learning, which is called "fuzzy Q-learning", continuous states and/or continuous actions can be handled [4,5]. In our market selection game, the market price of each market in the previous iteration is used as a continuous state variable in a fuzzy Q-learning-based strategy [1,2]. The estimation of the expected profit from each market (i.e., Q-value for each market) is conditioned by the market prices of all markets in the previous iteration. In our computer simulations, the market price of each market is partitioned into two linguistic values "*low*" and "*high*" in Fig. 3. These two linguistic values are used as antecedent fuzzy sets of fuzzy if-then rules of the following type:

$$\text{Rule } R_s\text{: If } p_1^{t-1} \text{ is } A_{s1} \text{ and } \cdots \text{ and } p_m^{t-1} \text{ is } A_{sm} \text{ then } Q_{i1}^t = q_{si1}^t \text{ and } \cdots \text{ and } Q_{im}^t = q_{sim}^t, \; s = 1, 2, \cdots, N, \tag{12}$$

where R_s is the label of the s-th fuzzy if-then rule, s is a rule index, p_j^{t-1} is the market price of the j-th market at the $(t-1)$-th iteration, A_{sj} is an antecedent fuzzy set, Q_{ij}^t is a Q-value, q_{sij}^t is a consequent

real number, and N is the number of fuzzy if-then rules. When we have two linguistic values ("*low*" and "*high*") as antecedent fuzzy sets for the market price of each of the m markets, the number of fuzzy if-then rules for each player is $N = 2^m$. In our computer simulations with five markets (i.e., $m = 5$), the number of fuzzy if-then rules for each player is $N = 2^5 = 32$.

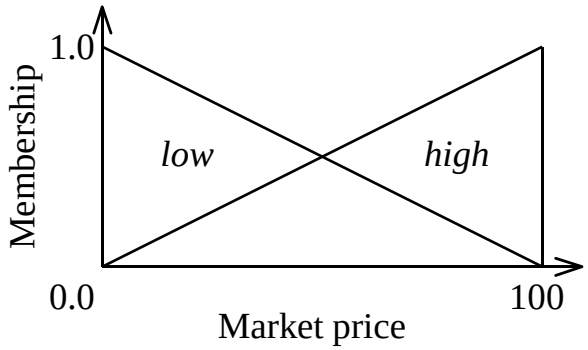


Figure 3: Membership functions of "*low*" and "*high*".

The Q-value of each player for each market at the t-th iteration is calculated by a fuzzy reasoning method from the fuzzy if-then rules in (12). Let us define the compatibility grade of the previous market prices $\mathbf{p}^{t-1} = (p_1^{t-1}, p_2^{t-1}, \cdots, p_m^{t-1})$ with the fuzzy if-then rule R_s by the product operation as

$$\mu_s(\mathbf{p}^{t-1}) = A_{s1}(p_1^{t-1}) \times A_{s2}(p_2^{t-1}) \times \cdots \times A_{sm}(p_m^{t-1}), \tag{13}$$

where $A_{sj}(\cdot)$ is the membership function of the antecedent fuzzy set A_{sj}. The Q-value of the i-th player for the j-th market at the t-th iteration is calculated by a fuzzy reasoning method as follows:

$$Q_{ij}^t = \frac{\sum_{s=1}^{N} \mu_s(\mathbf{p}^{t-1}) \cdot q_{sij}^t}{\sum_{s=1}^{N} \mu_s(\mathbf{p}^{t-1})} \quad \text{for } i = 1, 2, \cdots, n;\ j = 1, 2, \cdots, m;\ t = 1, 2, \cdots, T. \tag{14}$$

The consequent q_{sij}^t of each fuzzy if-then rule is updated after the market selection (i.e., when the actual profit is obtained) in a similar manner to the Q-learning-based strategy as

$$q_{sij}^{t+1} = \begin{cases} (1 - \alpha \cdot \mu_s^*(\mathbf{p}^{t-1})) \cdot q_{sij}^t + \alpha \cdot \mu_s^*(\mathbf{p}^{t-1}) \cdot r_{ij}^t, & \text{if } x_{ij}^t = 1, \\ q_{sij}^t, & \text{otherwise}, \end{cases} \tag{15}$$

where

$$\mu_s^*(\mathbf{p}^{t-1}) = \frac{\mu_s(\mathbf{p}^{t-1})}{\sum_{s=1}^{N} \mu_s(\mathbf{p}^{t-1})}. \tag{16}$$

As in the Q-learning-based strategy, the rule adjustment in (15) is performed only for the selected market. The amount of adjustment in (15) for each consequent value of a fuzzy if-then rule is proportional to the compatibility grade of the rule with the previous market prices.

The market selection is performed based on the calculated Q-values in (14). The selection probability of each market is defined by (11) in the same manner as in the Q-learning-based strategy.

4 Handling as Pattern Classification Problem

In this section, we propose the handling of our market selection game as a pattern classification problem. That is, the design of a decision making system for our game is handled as the design of a pattern classification system.

4.1 Data Generation

A pattern classification problem can be described as dividing a pattern space into several disjoint subspaces using labeled training patterns. Thus the reformulation of our market selection game as a pattern classification problem involves the definition of a pattern space and the collection of labeled training patterns. We define the pattern space of our pattern classification problem by the previous market prices $\mathbf{p}^{t-1} = (p_1^{t-1}, p_2^{t-1}, \cdots, p_m^{t-1})$ as in the fuzzy Q-learning-based strategy where the input space is defined by $\mathbf{p}^{t-1}$. Thus the pattern space of our pattern classification problem is the m-dimensional continuous space $\mathcal{R}^m$.

After each iteration of our market section game, we have a pattern vector $\mathbf{p}^{t-1} = (p_1^{t-1}, p_2^{t-1}, \cdots, p_m^{t-1})$. The point in the handing of our game as a pattern classification problem is how to define the class label corresponding to the pattern vector $\mathbf{p}^{t-1} = (p_1^{t-1}, p_2^{t-1}, \cdots, p_m^{t-1})$. In our game, the market selection can be described as choosing a single market after the previous iteration is completed. If our aim is to imitate a particular player, the class label should be the actually selected market by that player at the current iteration. In this case, a pair of the market prices $\mathbf{p}^{t-1} = (p_1^{t-1}, p_2^{t-1}, \cdots, p_m^{t-1})$ at the $(t-1)$-th iteration and the actually selected market at the t-th iteration is obtained as training data when the t-th iteration of our game is completed. Since our aim is not to imitate a particular player but to design a decision making system that hopefully works better than the player, we use as the class label the optimal market from which the player would obtain the maximum profit at the t-th iteration of our game. It should be noted that we can easily calculate the optimal market for the player after each iteration of the game (of course, it is impossible to identify the optimal market for any player before the current iteration of our game is completed). Thus the class label corresponding to the market prices $\mathbf{p}^{t-1} = (p_1^{t-1}, p_2^{t-1}, \cdots, p_m^{t-1})$ at the $(t-1)$-th iteration is the optimal market for the player at the t-th iteration. We obtain such an input-output pair after each iteration of our game is completed. Since we have m markets, our problem is an m-class pattern classification problem with the m-dimensional pattern space .

The first input-output pair is obtained after the second iteration is completed (in the first two iterations, the nearest market is selected). Thus when we design a decision making system for the third iteration, we have just a single input-output pair. In general, $(t-2)$ input-output pairs are available for the design of a decision making system for the t-th iteration of our game. It should be noted that different players have different training data because the optimal market is not the same for all players.

Let us illustrate the generation procedure of training data for Player A in Fig. 4. For illustration purpose, we iterated our market selection game by assigning the random strategy to all the 100 players in Fig. 4. Simulation results for the first five iterations are shown in Table 1. From this table, we have the following four input-output pairs as training data, which are available when we design a decision making system for the sixth iteration of our game.

After the 2nd iteration: (40, 31, 25, 37, 67); Market 2.
After the 3rd iteration: (52, 58, 37, 34, 19); Market 2.
After the 4th iteration: (40, 55, 37, 52, 16); Market 5.
After the 5th iteration: (49, 52, 37, 10, 52); Market 2.

Table 1: Simulation results of the first five iterations for Player 1.

No. of iterrations	Market Prices					Player A's Selection	Optimal Market
	Market 1	Market 2	Market 3	Market 4	Market 5		
1	40	31	25	37	67	Market 1	Market 5*
2	52	58	37	34	19	Market 5	Market 2
3	40	55	37	52	16	Market 4	Market 2
4	49	52	37	10	52	Market 2	Market 5
5	61	58	31	28	22	Market 1	Market 2

*The optimal market at the first iteration is not used as a part of training data.

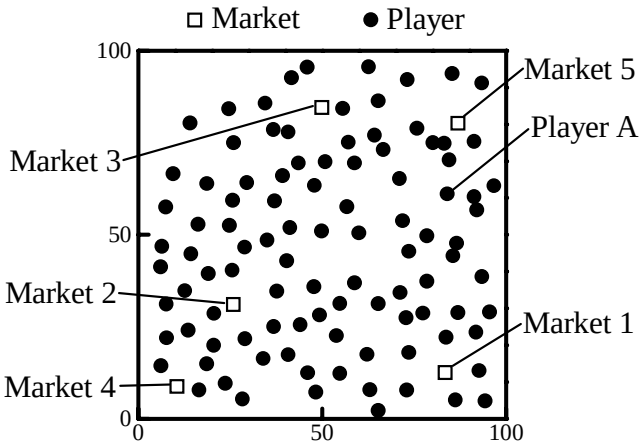


Figure 4: Location of Player A.

4.2 Pattern Classification Technique

For simplicity of notation, let us denote the $(t-2)$ input-output pairs available for the t-th iteration of our game as $(\mathbf{x}_k; C_k) = (x_{k1}, x_{k2}, \cdots, x_{km}; C_k)$, $k = 1, 2, \cdots, t-2$ where x_{kj} is the market price of the j-th market at the k-th iteration (i.e., $x_{kj} = p_j^k$) and C_k is the optimal market at the $(k+1)$-th iteration. Our task is to choose a single market for the t-th iteration of our game based on those training data $(\mathbf{x}_k; C_k)$, $k = 1, 2, \cdots, t-2$ and the previous market prices $\mathbf{p}^{t-1} = (p_1^{t-1}, p_2^{t-1}, \cdots, p_m^{t-1})$. This task is rephrased as designing a pattern classification system from the training data $(\mathbf{x}_k; C_k)$, $k = 1, 2, \cdots, t-2$, for classifying the new input pattern $\mathbf{p}^{t-1} = (p_1^{t-1}, p_2^{t-1}, \cdots, p_m^{t-1})$.

One of the most commonly used pattern classification techniques is the nearest neighbor classification where an input pattern is classified as the class of its nearest neighbor training pattern. In this paper, we use the nearest neighbor classification for our market selection game. The distance between the new input pattern $\mathbf{p}^{t-1} = (p_1^{t-1}, p_2^{t-1}, \cdots, p_m^{t-1})$ and each of the training patterns $\mathbf{x}_k = (x_{k1}, x_{k2}, \cdots, x_{km})$, $k = 1, 2, \cdots, t-2$ is measured as

$$Distance(\mathbf{p}^{t-1}, \mathbf{x}_k) = \sqrt{(p_1^{t-1} - x_{k1})^2 + (p_2^{t-1} - x_{k2})^2 + \cdots + (p_m^{t-1} - x_{km})^2}. \tag{17}$$

The new input pattern $\mathbf{p}^{t-1}$ is classified as the class C_{k^*} of the nearest neighbor training pattern $\mathbf{x}_{k^*}$ with the minimum distance. In the context of our market selection game, the above nearest neighbor classification for the market selection at the t-th iteration can be described as the following three procedures:

1. Find the k^*-th iteration at which the market prices are the most similar to those at the $(t-1)$th iteration.
2. Find the optimal market C_{k^*} at the (k^*+1)-th iteration from which the player would have obtained the highest profit.
3. Choose the market C_{k^*} for the t-th iteration.

4.3 Adaptation of Classification System

If the market conditions (e.g., demand-supply relations of some markets and strategies of other players) change during the iterative execution of our market selection game, the importance of information about old iterations is less than that of new information. For handling such difference in the importance of available information, we modify the distance measure in (17) so that new patterns are more likely to be selected as the nearest neighbor training pattern than old patterns as

$$ModifiedDistance(\mathbf{p}^{t-1}, \mathbf{x}_k) = \beta^{(t-2-k)} \times Distance(\mathbf{p}^{t-1}, \mathbf{x}_k), \tag{18}$$

where β is a positive constant ($\beta \leq 1$). In the modified distance measure in (18), $\beta^{(t-2-k)}$ can be viewed as a penalty factor introduced for increasing the distance between the input pattern $\mathbf{p}^{t-1}$ and old patterns $\mathbf{x}_k$'s. When $\beta = 1$, the modified distance measure in (18) is actually the same as the distance measure in (17). If we think that old information is much less important than new information, we may assign a relatively large value (i.e., 1.1) to β. On the contrary, if we think that old information is valuable as much as new information, β should be almost the same as 1.0. In our computer simulations, we specify β as $\beta = 1.0$ in the case of the fixed market conditions, and 1.1 when the market conditions are variable.

5 Computer Simulations

In this section, we examine the performance of the fuzzy Q-learning-based strategy and the nearest neighbor classification strategy through computer simulations on the market selection game in Fig. 1. For evaluating the performance of these two strategies, we also examine the other strategies described in Section 3.

5.1 Performance of Each Strategy

First we examine the performance of each strategy at the fixed market condition. In this subsection, we assume that all the 100 players use the same single strategy. In computer simulations, our game was repeated 1000 times (i.e., $t = 1, 2, \cdots, 1000$). For calculating the average profit per each iteration and each player, such a computer simulation was performed 100 times for each strategy because some strategies involve stochastic nature based on randomization procedures. Average profit obtained by each strategy is summarized in Table 2. From this table, we can see that good results were obtained by the Q-learning-based strategy and the fuzzy Q-leaning-based strategy. We can also see that the optimal strategy and the nearest neighbor classification strategies were the worst among the seven strategies. In our computer simulations, the roulette wheel selection with the linear scaling in (11) was used for the first 100 iterations of our game in the Q-learning-based strategy and the fuzzy Q-learning-based strategy. After the 100th iteration, the market with the maximum Q-value (i.e., $\max\{Q_{ij}^t | j = 1, 2, \cdots, 5\}$) was always selected at each iteration of our game.

Table 2: Average profit by strategy when all the players use the same strategy.

Strategy	Average profit
Random selection strategy	-15.2
Minimum transportation cost strategy	8.0
Optimal strategy for the previous actions	-52.8
Mimic strategy of the nearest neighbor player	-17.7
Q-learning-based strategy	16.1
Fuzzy Q-learning-based strategy	13.1
Nearest neighbor classification	-84.6

5.2 Competition between Strategies

In this subsection, we examine the competition between two or more strategies. First we examine the situation where one strategy is adopted by half of the players (i.e., 50 players). The other 50 players adopt another strategy. All combinations of two strategies were examined in the same manner as in the previous sections. That is, the competition between each pair of strategies was examined by 100 independent trials of the repeated game with 1000 iterations. In each trial, 100 players were randomly divided into two groups of 50 players. Simulation results are summarized in Table 3. From this table, we can see that good results were obtained by the minimum transportation cost strategy, the Q-learning-based strategy and the fuzzy Q-learning-based strategy.

Table 3: Simulation results of the competition between two strategies.

Strategy	Strategy of the other 50 players						
	Random	Cost	Optimal	Mimic	Q	Fuzzy Q	Nearest
Random	—	-15.3	-15.3	-15.2	-15.2	-15.2	-15.2
Cost	12.8	—	17.4	9.1	17.1	16.8	6.2
Optimal	-20.4	-18.4	—	2.4	-41.6	7.9	-20.0
Mimic	-16.4	-2.4	5.8	—	4.5	2.3	-2.5
Q	16.2	16.9	11.8	16.6	—	16.2	14.0
Fuzzy Q	12.5	13.5	15.3	13.2	13.3	—	12.0
Nearest	-17.2	-2.3	-16.3	-22.3	-40.4	-27.3	—

Next we examine the competition between two strategies in a different situation where a single player uses one strategy and the other 99 players use another strategy. All combinations of two strategies were examined. The performance of a strategy adopted by a single player was examined by 100 independent trials of the repeated game with 1000 iterations. Among those 100 trials, each of the 100 players was selected just once as the player to use the different strategy from the other 99 players. Simulation results are summarized in Table 4. From this table, we can see that the nearest neighbor classification strategy worked very well.

Furthermore we examine the competition among the seven strategies. In computer simulations, we specified the number of players adopting each strategy as follows:

Random strategy: 15 players,
Minimum transportation cost strategy: 15 players,

Table 4: Performance of each strategy when it was adopted by a single player.

Strategy	Strategy of the other 50 players						
	Random	Cost	Optimal	Mimic	Q	Fuzzy Q	Nearest
Random	—	-15.1	-15.2	-15.2	-15.2	-15.2	-15.1
Cost	18.1	—	21.9	13.3	17.6	18.4	-7.1
Optimal	15.2	31.8	—	17.8	19.3	17.7	2.6
Mimic	-15.2	6.1	64.2	—	15.1	13.1	10.0
Q	15.3	30.0	21.3	19.5	—	16.1	22.9
Fuzzy Q	12.2	29.2	62.9	17.3	14.2	—	22.9
Nearest	15.3	31.8	64.4	19.6	18.9	17.0	—

Optimal strategy for the previous actions: 14 players,
Mimic strategy of the nearest neighbor player: 14 players,
Q-learning-based strategy: 14 players,
Fuzzy Q-learning-based strategy: 14 players,
Nearest neighbor classification strategy: 14 players.

In the same manner as in the previous computer simulations, the performance of each strategy was calculated over 100 independent trials where players adopting each strategy were randomly selected. Simulation results are summarized in Table 5. From the comparison between Table 2 and Table 5, we can see that the performance of almost all strategies was improved by competing with many strategies.

Table 5: Simulation results of the competition among the seven strategies.

Strategy	Average profit
Random selection strategy	-15.2
Minimum transportation cost strategy	17.7
Optimal strategy for the previous actions	16.4
Mimic strategy of the nearest neighbor player	8.4
Q-learning-based strategy	16.1
Fuzzy Q-learning-based strategy	12.9
Nearest neighbor classification	15.2

From the simulation results in Table 2 - Table 5, we can see that the performance of each strategy strongly depends on situations. For example, while the optimal strategy for the previous actions and the nearest neighbor classification strategy worked very well in Table 4 and Table 5, their performance in the other tables was not good.

5.3 Adaptation to Dynamically Changing Conditions

In this subsection, we demonstrate how the fuzzy Q-learning-based strategy and the nearest neighbor classification strategy can adapt to changes of market conditions. First we examine the change of the demand-supply relation in each market. In the above computer simulations, we used the demand-supply relation in Fig. 2 (i.e., $p_j^t = 100 - 3 \cdot X_j^t$) for all the five markets. In this subsection, we change this demand-supply relation after the 500th iteration of our market selection game as follows:

Market 1: $p_1^t = 100 - 2 \cdot X_1^t$,
Market 2: $p_2^t = 100 - 4 \cdot X_2^t$,
Market 3: $p_3^t = 100 - 4 \cdot X_3^t$,
Market 4: $p_4^t = 100 - 2 \cdot X_4^t$,
Market 5: $p_5^t = 100 - 4 \cdot X_5^t$,

The performance of each strategy was evaluated by 100 independent trials of the repeated game with 1000 iterations where the seven strategies were competed as in the previous subsection (i.e., as in Table 5). In Fig. 5 - Fig. 7, we show the average performance at each iteration obtained by each strategy. From these figure, we can see that the above change of the demand-supply relations did not have a large effect on the performance of each strategy.

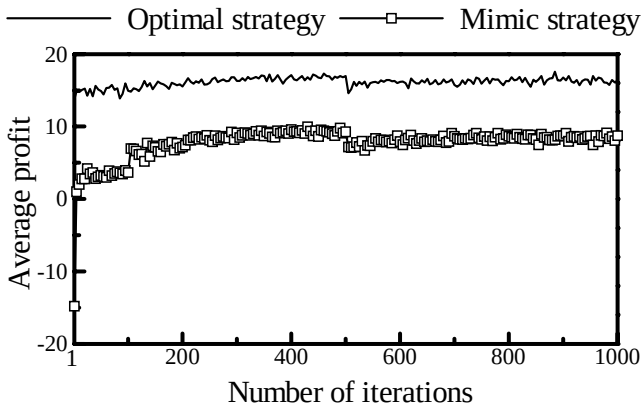


Figure 5: Performance of the optimal strategy and the mimic strategy.

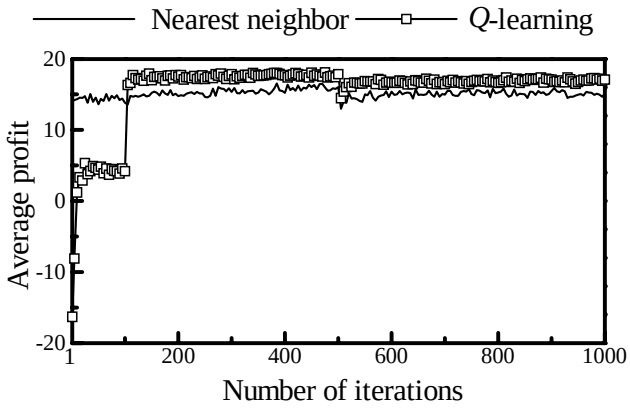


Figure 6: Performance of the nearest neighbor classification strategy and the Q-learning-based strategy.

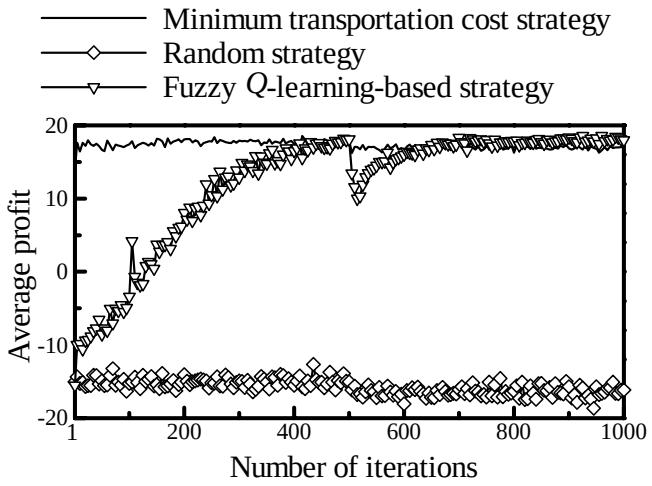


Figure 7: Performance of the minimum transportation cost strategy, the random strategy and the fuzzy Q-learning-based strategy.

Next we examine the change of strategies of other players. In computer simulations, we first assigned the minimum transportation cost strategy to 99 players. After the 500th iteration, we assigned the optimal strategy for the previous actions to those 99 players. Throughout 1000 iterations of our game, the fuzzy Q-learning-based strategy was used for the other single player. Such a computer simulation was performed 100 times so that each of the 100 players was selected as the fuzzy Q-learning player just once. In this manner, the performance of the fuzzy Q-learning-based strategy was evaluated. We also evaluated the performance of the nearest neighbor classification strategy in the same manner. For comparison, we also examined the performance of the Q-learning-based strategy. Simulation results are summarized in Fig. 8 - Fig. 10. From these figures, we can see that the fuzzy Q-learning-based strategy and the nearest neighbor classification strategy could adapt to the change of strategies of other players.

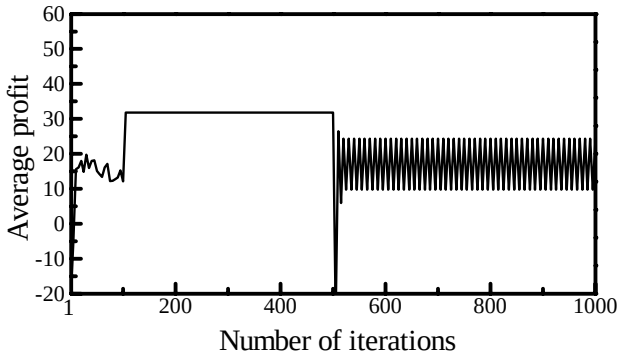


Figure 8: Performance of the Q-learning strategy.

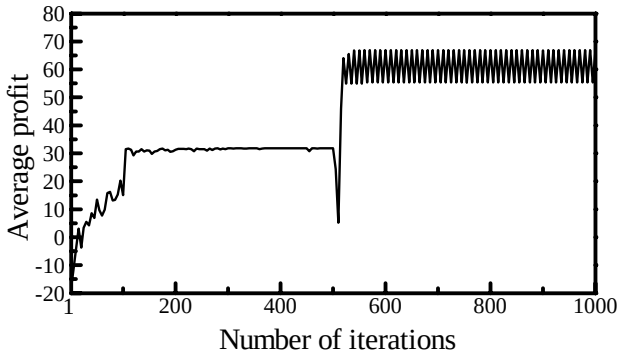


Figure 9: Performance of the fuzzy Q-learning-based strategy.

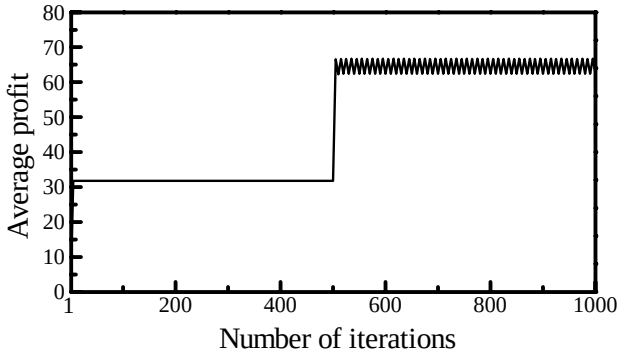


Figure 10: Performance of the nearest neighbor classification strategy.

6 Conclusion

In this paper, we showed how the design of a decision making system for a market selection game can be handled as a pattern classification problem. The design of the decision making system was viewed as the design of a pattern classification system. In this context, we proposed a nearest neighbor classification strategy for the market selection game. Through computer simulations, we examined the performance of various strategies for the market selection game. Simulation results indicated that the performance of each strategy strongly depended on situations. Strategies, which worked very well on some situations, did not work well on other situations. Some adaptable strategies such as the fuzzy Q-learning-based strategy and the nearest neighbor classification strategy could quickly respond to changes of market conditions.

Acknowledgment

This work was partially supported by Foundation for Fusion of Science & Technology.

References

[1] H. Ishibuchi, T. Nakashima, H. Miyamoto, and C. H. Oh, "Fuzzy Q-learning for a multi-player non-cooperative repeated game," *Proc. of 6th IEEE International Conference on Fuzzy Systems* (Barcelona, Spain) pp. 1573-1579, 1997.

[2] H. Ishibuchi, C.-H. Oh, and T. Nakashima, "Competition between strategies for a market selection game," *Proc. of Complex Systems' 98* (Sydney, Australia) , pp. 272-281, 1998. Also in *Complexity International*, vol. 6, 1999 (On line Journal; http://www.csu.edu.au/ci/).

[3] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279-292, 1992.

[4] P. Y. Glorennec, "Fuzzy Q-Learning and Dynamical Fuzzy Q-Learning," *Proc. of 3rd IEEE International Conference on Fuzzy Systems* (Orlando, USA) pp. 474-479, 1994.

[5] L. Jouffe and P. Y. Glorennec, "Comparison between connectionist and fuzzy Q-learning," *Proc. of 4th International Conference on Soft Computing* (Iizuka, Japan), pp. 557-600, 1996.