Evolving Subjective Utilities: Prisoner's Dilemma Game Examples

Koichi Moriyama Satoshi Kurihara Masayuki Numao

The Institute of Scientific and Industrial Research, Osaka University
8-1, Mihogaoka, Ibaraki, Osaka, 567-0047, Japan
{koichi, kurihara, numao}@ai.sanken.osaka-u.ac.jp

ABSTRACT

We have proposed the *utility-based Q-learning* concept that supposes an agent internally has an emotional mechanism that derives subjective utilities from objective rewards and the agent uses the utilities as rewards of Q-learning. We have also proposed such an emotional mechanism that facilitates cooperative actions in Prisoner's Dilemma (PD) games.

However, this mechanism has been designed and implemented manually in order to *force* the agents to take cooperative actions in PD games. Since it seems slightly unnatural, this work considers whether such an emotional mechanism exists and where it comes from. We try to *evolve* such mechanisms that facilitate cooperative actions in PD games by conducting simulation experiments with a genetic algorithm, and we investigate the evolved mechanisms from various points of view.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*intelligent agents, multiagent systems*

General Terms

Experimentation

Keywords

Reward structures for learning; Multiagent Learning; Evolution, adaptation; Game theory

1. INTRODUCTION

In this paper, we consider a learning agent that chooses actions seeming appropriate. The most popular learning method for agents is reinforcement learning [14]. In reinforcement learning, an agent is given rewards from the environment according to its actions and the states of the environment, and the agent learns to take actions that maximize the rewards.

If there is only one agent in the world, it can take actions that maximize its own rewards. However, in a multiagent environment consisting of multiple agents, the maximization becomes impossible because of interactions among agents. In this case, cooperative behaviors like yielding are necessary, but reinforcement learning as a reward-maximizer fails to learn such behaviors.

When we consider the most intelligent agents, i.e., humans, they take cooperative behaviors in such cases. For example, Fehr and Schmidt [3] tried to explain human behaviors by introducing "inequity-aversion" assumption into their models. This assumption represented a feeling of aversion to the difference from others and prevented the models from outsmarting others. Rilling et al. [11] reported that, when humans took cooperative behaviors, reward processing areas in their brains were positively activated. They inferred that the cooperative behaviors themselves became a kind of rewards in their brains.

Based on them, we have proposed the *utility-based Q-learning* concept [8]. This concept supposes that the agent internally has an *emotional mechanism* that derives *subjective utilities* from *objective rewards*, and it uses the utilities as rewards of Q-learning [16], a representative method of reinforcement learning. We have also proposed such an emotional mechanism that facilitates cooperative actions in Prisoner's Dilemma (PD) games [8]. PD games [1, 9] are the most famous two-person two-action game in game theory, in which if the two players try to maximize individual payoffs without cooperating with each other, they obtain less payoffs than those they would obtain when cooperating with each other. The mechanism we have proposed derives a larger subjective utility than the payoff each agent is given when the agents take mutual cooperation in PD games. The agents learning by the utilities take cooperation afterward.

However, this utility deriving function has been designed and implemented manually in order to *force* the agents to take cooperative actions in PD games. Since it seems slightly unnatural, this work considers whether such an emotional mechanism exists and where it comes from. We try to *evolve* such mechanisms that facilitate cooperative actions in PD games by conducting simulation experiments with a genetic algorithm. Notice that the objective of this work is not to acquire a strategy itself in PD games, but to know whether such an emotional mechanism evolves.

This paper consists of six sections. Section 2 introduces PD games, Q-learning, the utility-based Q-learning concept, and genetic algorithms, which are used in the following sections. In Section 3, we see the experiment scheme to obtain the emotional mechanisms for PD games by evolution. In Section 4, we investigate the emotional mechanisms we obtained from the experiment. After we check several related works in Section 5, this paper is summarized in Section 6.

Cite as: Evolving Subjective Utilities: Prisoner's Dilemma Game Examples, Koichi Moriyama, Satoshi Kurihara, and Masayuki Numao, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 233-240.

Table 1: Prisoner's Dilemma game

$A \setminus B$	C	D
C	R, R	S, T
D	T, S	P, P

2. BACKGROUNDS

This section introduces PD games, Q-learning, the utility-based Q-learning concept, and genetic algorithms, which are used in the following sections. We use the terms "payoff" and "reward" in the same sense in this paper.

2.1 Prisoner's Dilemma Games

A Prisoner's Dilemma (PD) game [1, 9] is a two-person two-action game and is often shown by a bimatrix, called a payoff matrix (Table 1). Each player has two actions C (cooperation) and D (defection). The players A and B choose actions from rows and columns, respectively. After choosing actions, each player obtains a payoff in Table 1. For example, when A and B choose C and D, respectively, A and B obtain payoffs S and T, respectively. Hereafter, (X, Y) stands for the pair of actions of both players such that X and Y are the actions of A and B, respectively.

PD has the following relations among payoffs:

$$T > R > P > S \quad \text{and} \quad 2R > T + S.$$

Under these relations, each player obtains a larger payoff when he/she chooses D regardless of the opponent's action. As a result, the action pair becomes (D, D) and each player obtains a payoff P. However, it is more desirable for both players to choose the first actions (C, C) and obtain R which is larger than P. Since rational decisions of both players give a worse result, this game is called "dilemma".

2.2 Q-learning

Suppose an agent senses a state $s_t \in \mathcal{S}$ and chooses an action $a_t \in \mathcal{A}(s_t)$ at a discrete time t. $\mathcal{S}$ is a set of possible states in the environment and $\mathcal{A}(s_t)$ is a set of possible actions in the state s_t. After choosing an action, the agent receives a *reward* $r_{t+1} \in \mathbb{R}$ and senses a new state s_{t+1}. Q-learning [16] updates an *action value function* Q by the following rule to make it approach the true *value* under the optimal policy π^*, which is the expected sum of rewards discounted by $\gamma \in [0, 1)$ under π^*, i.e., $E_{\pi^*}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+1+k}\right)$.

$$Q_{t+1}(s, a) = \begin{cases} Q_t(s_t, a_t) + \alpha\, \delta_t & \text{if } (s, a) = (s_t, a_t), \\ Q_t(s, a) & \text{otherwise.} \end{cases}$$

$$\delta_t \equiv r_{t+1} + \gamma \max_{a' \in \mathcal{A}(s_{t+1})} Q_t(s_{t+1}, a') - Q_t(s_t, a_t).$$

$\alpha \in (0, 1]$ is a parameter called the learning rate and δ_t is called TD error that approaches 0 when $Q(s, a)$ approaches the true value of the action a in the state s under π^*. For all s and a, $Q(s, a)$ is proved to converge to the true value $Q^*(s, a)$ under π^* with probability one when (i) the environment has the Markov property, (ii) the agent visits all states and takes all actions infinitely, and (iii) α is decreased properly [16].

If the true value function Q^* is known, the agent can choose an optimal action a^* in a state s from Q^* by

$$a^* = \arg \max_{a'' \in \mathcal{A}(s)} Q^*(s, a'').$$

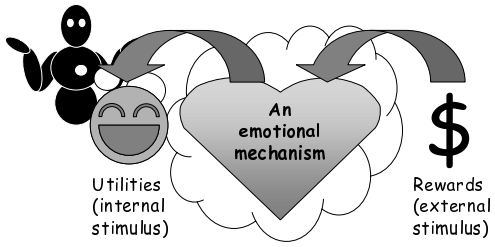


Figure 1: An emotional mechanism

However, if the agent always chooses such actions during learning, Q may converge to a local optimum because the agent may not visit all states. To avoid it, the agent usually uses a stochastic method like ε-greedy [14] to choose actions. ε-greedy method chooses either an action having the maximum Q with probability $1 - \varepsilon$ or a random action with probability ε.

2.3 Utility-based Q-learning

The utility-based Q-learning concept [8] supposes that the agent internally has an emotional mechanism that derives *subjective utilities* from *objective rewards* (Figure 1), and it uses the utilities as rewards of Q-learning. Especially, in one-state Q-learning, we have also proposed such an emotional mechanism, namely a utility deriving function, that facilitates mutual cooperation in PD games [8]. This utility deriving function derives a utility u from a reward r,

$$u \equiv \begin{cases} r + r' (\equiv R + r') & \text{if the actions are } (C, C), \\ r & \text{otherwise,} \end{cases} \quad (1)$$

such that

$$r' \geq \frac{P - (\alpha R + (1 - \alpha) S)}{\alpha} \quad (2)$$

and $\alpha \in (0, 1)$ is constant.

The utility u derived from Equation 1 makes the action value of cooperation larger than (or equal to) that of defection, i.e., $Q(C) \geq Q(D)$, after a single mutual cooperation. It comes theoretically from the number of consecutive mutual cooperation that is necessary to make $Q(C) \geq Q(D)$, when $Q(D)$ is the limit value after infinite mutual defections. See the paper [8] for details.

2.4 Genetic Algorithm

A genetic algorithm (GA) [5, 7] is an algorithm that simulates the evolution of creatures. GA uses many entities called *genomes* consisting of multiple *genes*. Each genome represents a solution of the target problem. GA finds a genome that maximizes a *fitness function*, which indicates goodness of a genome for the target problem, by iterating *selection*, *crossover*, and *mutation*. GA can be applied to many problems of which we can define fitness functions.

Simple GA first defines genomes and a fitness function from solution candidates and the objective of the problem, respectively. After that, GA executes the following procedure iteratively. One iteration is called one *generation*. Finally, a genome with the maximum fitness shows the best solution satisfying the objective of the problem.

1. Generate N genomes in a set named "current" randomly and prepare an empty set named "new".

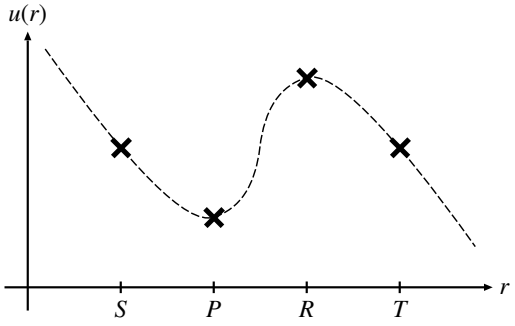


Figure 2: Assumption of a utility deriving function

2. Calculate the fitness of each genome in the current set.
3. According to the fitness, select two genomes from the current set (selection), copy them, and apply the following procedure to the copied genomes to create new genomes. After that, the new genomes are added to the new set.
   - Exchange several genes between the two genomes with probability p_c (crossover).
   - Modify each gene in each genome with probability p_m (mutation).
4. Until the size of the new set becomes N, repeat 3.
5. If the termination condition is not satisfied, clear out the current set and move all the contents of the new set to the current set. Back to 2.

3. EVOLVING UTILITIES

Although *the* utility-based Q-learning with the utility deriving function (Equation 1) was shown to bring out mutual cooperation [8], this function has been designed and implemented manually in order to *force* the agents to take cooperative actions in PD games. Since it seems slightly unnatural, this work considers whether such a utility deriving function, i.e., an emotional mechanism, exists and where it comes from. Humans, who may show mutual cooperation, appeared as the result of evolution. Therefore, we try to obtain such an emotional mechanism by evolutionary calculation using GA.

This work especially investigates how utility deriving functions evolve by GA when N agents having these functions play PD games in a round-robin fashion.

3.1 Genome Definition

In order to apply GA to evolving utility deriving functions, first we have to define genomes. Suppose $u(r)$ shows a utility deriving function that derives a utility u when the reward is r. If the payoffs $T > R > P > S$ in Table 1 become $u(R) > u(T)$, $u(S) > u(P)$, and $u(R) > u(P)$, it is obvious that the action C becomes superior to D. Since this function $u(r)$ can be depicted as like Figure 2, we assume that $u(r)$ is a cubic function, i.e.,

$$u \equiv u(r) \equiv ar^3 + br^2 + cr + d, \tag{3}$$

and the coefficients a, b, c, d are evolved by real-valued GA. Note that we do not assume that the cubic function itself promotes cooperation. We simply choose it because it may be the simplest function that can represent the assumption of Figure 2[1].

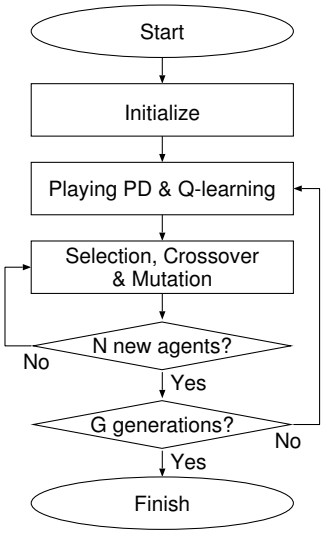


Figure 3: Flowchart of a run of the experiment

3.2 Experiment Scheme

Let N and G be the number of agents in a generation and the total number of generations, respectively. Figure 3 shows the overview of a run of the experiment. This is based on the simple GA shown in Section 2.4.

Initialize: Two empty sets named "current" and "new" are prepared. N Q-learning agents each of which has its own u represented by a genome (Equation 3) are constructed and put into the current set. Each gene, i.e., the coefficients a, b, c, d of Equation 3, is a random real value in a finite interval. The action value functions Q of the agents are initialized.

Playing PD & Q-learning: N agents play iterative PD games in a round-robin fashion. Each play consists of M games. After each game, each agent executes utility-based Q-learning with the utility u derived from Equation 3.

Let r_{ij}^g be the sum of payoffs an agent i obtains in a play with an agent j at the generation g. After playing with all other agents, the agent i calculates the (raw) fitness $f_i^g = \sum_j r_{ij}^g$. Notice that the fitness is the sum of payoffs, not the sum of utilities.

Selection, Crossover & Mutation: Two agents are chosen from the current set by roulette wheel selection according to the *scaled* fitness that is calculated from f_i^g with a linear scaling method [5] with the coefficient ξ. That avoids premature convergence to extraordinary individuals at the beginning and emphasizes the differences among genomes at the end.

[1]Naturally, there are other functions that can represent the assumption. We need to investigate the cases of using other functions.

The genomes of the selected agents are copied. The copied genomes are crossed, with probability p_c, by the uniform crossover that swaps genes of the genomes with probability 0.5. Then, each of the two genomes is mutated by adding a real value from a Gaussian distribution $N(\mu, \sigma)$ to each gene with probability p_m.

After that, two agents having the two new genomes are constructed and added to the new set. The action value functions Q of the new agents are initialized.

***N* new agents?** Until the size of the new set becomes N, "Selection, Crossover & Mutation" is repeated.

***G* generations?** Until evolving G generations, the current set is cleared out and all agents in the new set are moved to the current set. The whole process except "initialize" is repeated.

We use the following parameters in the experiment: the number of agents $N = 100$, the number of generations $G = 10000$, the number of games in a play $M = 1000$, the coefficient of the linear scaling method $\xi = 1.2$, the probability of crossover $p_c = 0.9$, and that of mutation $p_m = 0.01$. The interval for each gene is $[-10, 10]$ and the Gaussian distribution for mutation is $N(0, 1)$. We do not employ the elite strategy that copies the best genome into the next generation automatically.

The payoffs of PD game are identical with those used in the Axelrod's tournament [1], i.e., $(T, R, P, S) = (5, 3, 1, 0)$.

The number of state of Q-learning is set to one. This means each agent does not remember the action sequences at all and thus each game becomes identical with a one-shot PD game for the agents. The parameters of Q-learning are identical with those in the previous paper [8], i.e., the learning rate $\alpha = 0.25$, the discount factor $\gamma = 0.5$, ε in ε-greedy method is 0.05, and the initial values of the action value function Q are all zero. Since $M \times (\varepsilon/2)^2 = 0.625$, both agents take an action that has less Q simultaneously around 0.625 times in a play.

GAlib[2] 2.4.7 is used as the implementation of real-valued GA.

4. RESULTS

The experiment described in Section 3.2 was conducted 100 runs. In this section, we investigate the results of the experiment.

4.1 Did mutual cooperation occur?

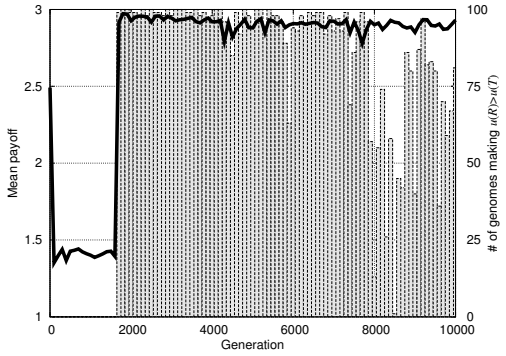
At the end of each run, the average payoff each agent obtained per game became more than 2.7 in 83 out of 100 runs. This means that mutual cooperation occurred because $(T + S)/2 = 2.5$. However, in the remaining 17 runs each agent obtained around 1.4 in the mean, which showed mutual defection was continuing.

Figure 4 shows the mean payoffs per game at each generation in a certain run. The function u of the best agent that obtained the highest payoff at the end of this run became

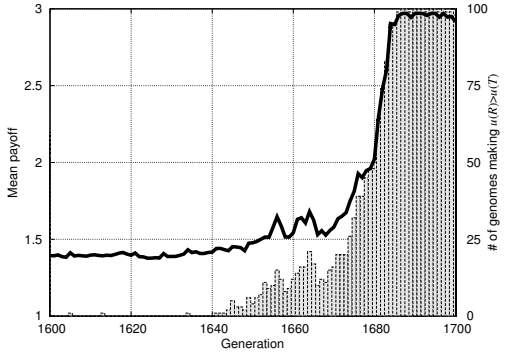
$$u(r) = -1.18073r^3 + 7.81789r^2 - 4.44985r - 10$$

and the agent obtained 2.95169 per game. There were eight agents that had same u, and they obtained more than 2.9 in the mean. This u is shown by a solid line in Figure 5. It goes against the assumption of Figure 2 because $u(R) \equiv u(3) = 15.13175$ is slightly less than $u(T) \equiv u(5) = 15.60675$. It is also obvious that $u(S) \equiv u(0) < u(1) \equiv u(P)$. Thus, this u also gives the agent a PD game.

[2] `http://lancet.mit.edu/ga/`



(a) Overall view (plotting every 100 generations)



(b) Closeup at the phase transition (plotting every generation)

Figure 4: Mean payoffs per game (line with the left scale) and the number of genomes making $u(R) > u(T)$ (bars with the right scale) in a certain run

We can see in Figure 4(b) a kind of phase transition where the average payoffs rose suddenly in a short time of a few dozen games. Although it depended on runs when the phase transition occurred, the phase transition period was a few dozen games in most runs. This is discussed in Section 4.3.

It is clear that the first coefficient a in Equation 3 should be negative so as to realize the assumption of Figure 2. Although the coefficients of most genomes at the end of the 83 successful runs were negative, those of 16 out of 17 failed runs were positive. In the remaining one run, the mean payoffs per game around the end were fluctuating. It might be the beginning of the phase transition.

4.2 What property did u have?

In addition to u of the best agent at the end of a certain run, Figure 5 also shows u of the best agent at the 1700th generation of this run, which was just after the phase transition, by a dotted line. We can see that, at the 1700th generation, $u(R)$ was obviously larger than $u(T)$ as the assumption of Figure 2, but at the end of the run, as we saw before, $u(R)$ became less than $u(T)$.

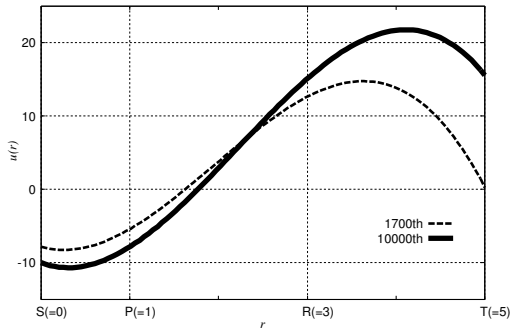


Figure 5: Utility deriving functions of agents obtaining the highest per-game payoff at the 1700th generation and at the 10000th generation in the run of Figure 4, respectively

Hence, let us investigate the relation between $u(R)$ and $u(T)$ in all generations. Figure 4 also shows the number of genomes that made $u(R) > u(T)$ by bars. Let us call such genomes "R-preferring". We can see that, most genomes were R-preferring until around the 8000th generation, but the number fluctuated after that. We have not found the reason why that fluctuation happens. We can only say about the run that, if the best genome was R-preferring at the start of decreasing, it was replaced by a non-R-preferring genome soon. On the other hand, interestingly, when the number was increasing, the best non-R-preferring genome continued to stay for a while and R-preferring genomes were prevailing in the middle level.

We investigated all 83 runs showing the phase transition and found the following two cases:

1. Most of all genomes were R-preferring at least once, like the example shown before (Figure 4), and
2. Only a small number of genomes were R-preferring, like an example shown in Figure 6.

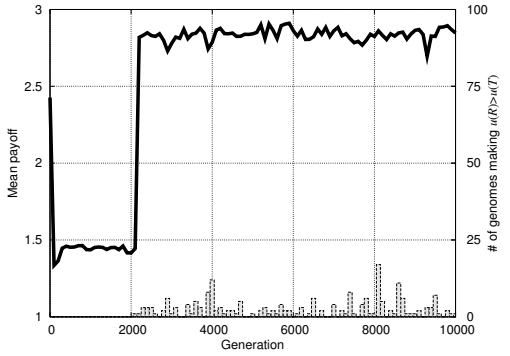
In order to eliminate the effect of random initial values, we ignore first 20 generations and analyze the remaining 9980 generations of the result of each run. It was 35 out of 83 runs that all genomes were R-preferring at least once and it was 19 out of 83 runs that R-preferring genomes were in the minority in all generations. It was only five out of 83 runs that the best genome were R-preferring at the end of the runs. Hence, in at least 30 runs, the best genome changed from an R-preferring one to a non-R-preferring one.

4.3 What happened at the phase transition?

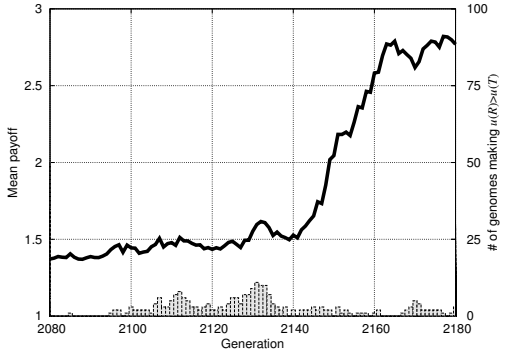
In Section 4.2, we investigated the number of R-preferring genomes in *all* generations. In this section, let us focus on the phase transition period and investigate the number of R-preferring genomes and generations in this period.

First the period is to be defined. Based on the results like Figures 4 and 6, we define the start and the end of the period as the generations when the average payoff first exceeded 1.6 and 2.7, respectively.

Under this definition, Figure 7 shows the length of the period. Y-axis shows the number of generations in log scale and x-axis shows each run sorted by the number of generations. From this figure, we can see that, in most runs, the



(a) Overall view (plotting every 100 generations)



(b) Closeup at the phase transition (plotting every generation)

Figure 6: A certain run in which only a small number of genomes made $u(R) > u(T)$

phase transition period was less than 100 generations, and the length seems to follow an exponential function.

Figure 8 shows the average numbers of R-preferring genomes during the period. In each run, the total number of R-preferring genomes during the period was divided by the number of generations of the period. X-axis shows each run sorted by the average number of genomes. We can see that this graph consists of two linear lines bending around the 37th, where the number of R-preferring genomes is around 10. Eighteen out of the aforementioned 19 runs of the "minority case", in which R-preferring genomes were in the minority in all generations, were left side of the bending point. On the other hand, 31 out of the 35 runs of the "majority case", in which all genomes were R-preferring at least once, were right side of the point. Therefore, the left side and the right side may correspond to the minority case and the majority case, respectively. It suggests that about a half of the phase transition is similar to the minority case. This is interesting because it disagrees with the assumption of Figure 2.

Figure 9 shows the relation between the number of R-preferring genomes in the phase transition period and the duration of the period. We can see the variance became smaller as the number of genomes increased.

More analyses are necessary to know the reason why mutual cooperation occurred even when R-preferring genomes were in the minority. Perhaps, since the condition $u(R) >$

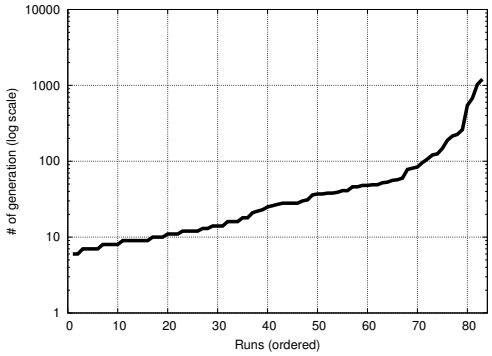


Figure 7: Number of generations needed for phase transition. x: each run (sorted by the number of generations), y: number of generations (log scale)

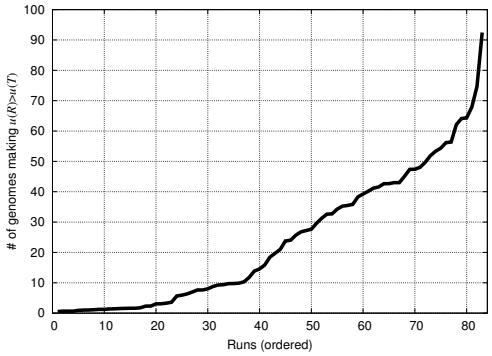


Figure 8: Average number of genomes that made $u(R) > u(T)$ in the phase transition period. x: each run (sorted by the average number), y: average number of genomes

$u(T)$ is not a necessary one but a sufficient one, genomes in such runs may have evolved so as to satisfy only (unknown) necessary conditions.

4.4 Relation between u and Formula 2

Formula 2 appeared in Section 2.3 is a kind of necessary condition to facilitate mutual cooperation in PD games. Hence, based on the discussion in Section 4.3, here we investigate the relation between the result of this work and Formula 2.

Assigning the learning rate $\alpha = 0.25$ of the experiment to Formula 2 gives the following formula:

$$r' \geq 4P - R - 3S.$$

From this formula, if $4P - R - 3S > 0$, i.e., $R < 4P - 3S$, r' should be larger than 0. This means that, in order to maintain mutual cooperation in one-state Q-learning, it is necessary to make the payoff larger. On the other hand, if $R \geq 4P - 3S$, the mutual cooperation can be maintained by the original payoff. Hence, we check whether the utilities u calculated by Equation 3 satisfied

$$u(R) \geq 4u(P) - 3u(S) \quad (4)$$

so as to know whether the mutual cooperation could be maintained only by the evolved utilities.

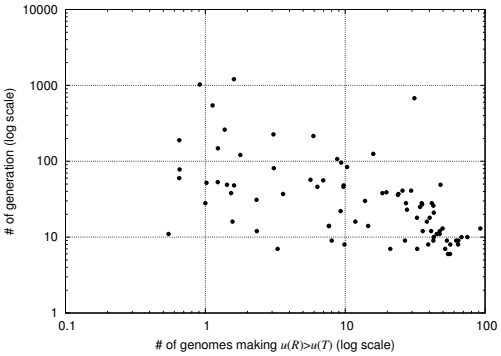


Figure 9: Relation between the number of genomes making $u(R) > u(T)$ in the phase transition period and the duration of the period. x: average number of genomes (log scale), y: number of generations (log scale)

The result shows that most of all genomes in most of all runs satisfied Formula 4 regardless of the phase transition. It is because, as $u(T)$ became much larger before the phase transition, $u(R)$ also became much larger than $u(P)$ and $u(S)$[3]. It may be a reason why mutual cooperation occurred even when R-preferring genomes were in the minority.

5. RELATED WORKS

There are several existing works in which agents learn strategies in PD games. Sandholm and Crites [12] conducted various experiments in which Q-learning agents played PD games and investigated whether or not mutual cooperation occurred. They reported that mutual cooperation did not occur when both agents did not take their past actions into account and that the parameters determining exploration rates had a major impact on the result when both agents used the past actions. Stimpson et al. [13] investigated what patterns the *satisficing strategy* produced in PD games. This strategy consisted of a parameter named *aspiration*, a simple behavior rule, and a simple update rule for aspiration. Under the behavior rule, the agent (i) continued to take the current action if it gave a larger payoff than aspiration, or (ii) took the other action otherwise. This work is interesting because it tried to derive a satisfactory result instead of an optimal result.

There are a lot of works that tried to obtain strategies of PD games by evolutionary algorithms. Here we see only a few of them. Axelrod [1] held two computer competitions of PD games, in which all of the attending programs played iterative PD games in a round-robin fashion. In both competitions, a strategy named Tit for Tat (TFT) became the best one according to the mean payoffs. After the competitions, he investigated what strategies of PD games evolved by GA [2]. First he specified that each locus of a gene indicated the sequence of past three actions and each gene indicated the next action. After that, he calculated some representative

[3]This may depend largely on the fact that u was a continuous function in which $u(T)$ and $u(R)$ were (somewhat) dependent. We also have to investigate the cases where u is a discrete function, in which they are completely independent.

strategies statistically from all of the attending programs of the second competition. He reported that, by playing PD games with these representative strategies, GA evolved strategies like TFT. Also, he reported that if all genomes in the genome set played PD games in a round-robin fashion, mutual cooperation occurred after growth and decay of defection. Fogel [4] also reported that cooperation occurred in PD games if genomes were defined as finite state automata. Vega-Redondo [15] introduced multiple sets of genomes. In each set, genomes played PD games with each other and evolved according to payoffs, and, simultaneously, each set was also under selection pressure according to the sum of payoffs of member genomes. Under this setting, he showed mutual cooperation was evolved even in one-shot PD games. Obviously, the selection of the sets themselves facilitated the cooperation.

Several works used both learning and evolution. Hingston and Kendall [6] introduced a kind of "switch" gene into the traditional genome which represented (fixed) next actions depending on the past action sequences. The switch gene determined whether the agent learned the opponent's action model and changed its actions according to the model. They investigated how the genomes evolved by mutations and what strategy was spread. The result showed that the learning genomes did not increase so much. Quek et al. [10] proposed memetic learning and tried to obtain strategies of PD games. Memetic learning combined learning and evolution as an optimizer in an individual and a communicator between individuals, respectively. They reported that by combining learning and evolution, both compensated each other and, as a result, memetic learning could derive good strategies.

All of those works investigated strategies themselves for PD games. However, the work of this paper is not to derive strategies themselves for PD games, but to investigate whether an emotional mechanism that derives subjective utilities can evolve according to objective payoffs. This is a major difference between this work and those works.

6. CONCLUSION

The utility-based Q-learning concept supposes an agent internally has an emotional mechanism deriving subjective utilities from objective payoffs and it uses the utilities as rewards of Q-learning. In this work, we tried to obtain such an emotional mechanism by evolutionary computation using genetic algorithm (GA). Especially, we tried to evolve such mechanisms that facilitated cooperative actions in a game named Prisoner's Dilemma (PD).

First we assumed that the mechanism could be represented by a cubic function and used real-valued GA to evolve its coefficients. We succeeded in obtaining mutual cooperation in 83 out of 100 runs. Although, in at least one generation, all genomes in 35 out of 83 runs were R-preferring, i.e., $u(R) > u(T)$, there were only five runs in which the best genome was R-preferring at the end. Even more surprisingly, R-preferring genomes were in the minority in all generations of 19 runs.

It was also shown that mutual cooperation followed a phase transition. In most runs, the phase transition period was less than 100 generations. The average numbers of R-preferring genomes at the phase transition suggested that about a half of the phase transition might occur even when R-preferring genomes were in the minority. We also investigated the relation between the evolved u and Formula 2, which was originated in the previous paper [8], so as to know whether the evolved u could maintain the mutual cooperation by themselves. The result showed that most of all genomes in most of all runs satisfied the formula regardless of the phase transition. It may be a reason why mutual cooperation occurred even in the minority case.

Until now, this work is in a preliminary phase of showing results in a certain setting. We have to discuss the reason why such the results emerge and also have to investigate the effect of settings on the results. Especially, we have to clarify the property of the phase transition rigorously. Also, we should analyze the dynamics of the evolution process to elucidate the results of this paper.

7. REFERENCES

[1] R. Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.

[2] R. Axelrod. The Evolution of Strategies in the Iterated Prisoner's Dilemma. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, pages 32–41. Pitman / Morgan Kaufmann, London / Los Altos, CA, 1987.

[3] E. Fehr and K. M. Schmidt. A Theory of Fairness, Competition, and Cooperation. *Quarterly Journal of Economics*, 114:817–868, 1999.

[4] D. B. Fogel. Evolving behaviors in the iterated prisoner's dilemma. *Evolutionary Computation*, 1:77–97, 1993.

[5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[6] P. Hingston and G. Kendall. Learning versus Evolution in Iterated Prisoner's Dilemma. In *Proc. 2004 Congress on Evolutionary Computation, CEC'04*, pages 364–372, Portland, Oregon, U.S.A., 2004.

[7] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

[8] K. Moriyama. Utility based Q-learning to facilitate cooperation in Prisoner's Dilemma games. *Web Intelligence and Agent Systems*, 7(3):233–242, 2009.

[9] W. Poundstone. *Prisoner's Dilemma*. Doubleday, New York, 1992.

[10] H.-Y. Quek, K. C. Tan, C.-K. Goh, and H. A. Abbass. Evolution and Incremental Learning in the Iterated Prisoner's Dilemma. *IEEE Transactions on Evolutionary Computation*, 13:303–320, 2009.

[11] J. K. Rilling, D. A. Gutman, T. R. Zeh, G. Pagnoni, G. S. Berns, and C. D. Kilts. A Neural Basis for Social Cooperation. *Neuron*, 35:395–405, 2002.

[12] T. W. Sandholm and R. H. Crites. Multiagent reinforcement learning in the Iterated Prisoner's Dilemma. *BioSystems*, 37:147–166, 1996.

[13] J. L. Stimpson, M. A. Goodrich, and L. C. Walters. Satisficing and Learning Cooperation in the Prisoner's Dilemma. In *Proc. 17th International Joint Conferences on Artificial Intelligence, IJCAI-01*, pages 535–540, Seattle, Washington, U.S.A., 2001.

[14] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[15] F. Vega-Redondo. Long-run cooperation in the one-shot Prisoner's Dilemma: A hierarchic evolutionary approach. *BioSystems*, 37:39–47, 1996.

[16] C. J. C. H. Watkins and P. Dayan. Technical Note: Q-learning. *Machine Learning*, 8:279–292, 1992.