The Emergence of Cooperation in a Society of Autonomous Agents — The Prisoner's Dilemma Game Under the Disclosure of Contract Histories —

Akira Ito and Hiroyuki Yano
Kansai Advanced research Center
Communications Research Laboratory
588-2, Iwaoka, Nishiku, Kobe, 651-24, Japan
{ai, yano}@crl.go.jp

Abstract

The emergence of cooperation in a society of autonomous agents is investigated. Each agent is made to repetitively engage in a deal equivalent to the "Prisoner's Dilemma" game, each time changing the other party of the deal. The conditions of the deal are that the contract histories of all the agents are disclosed to the public. Several deal strategies are evaluated, and their behaviors are investigated by matching them under various conditions. Next the social evolution of deal strategies is investigated using genetic algorithm techniques. Each agent can bear a child according to the profit he gets through the deal. The child inherits the deal strategy of the parent, but the random mutation is introduced to the inheritance of strategies. It is shown that the robust and cooperative strategies emerges through the evolution starting from a simple "Tit for Tat" algorithm.

Introduction

In a society of autonomous agents, each agent seeks its own benefit in interacting with other agents. How can cooperation ever be possible in such a society of agents? The problem has been investigated extensively by Axelrod (Axelrod and Hamilton 1981; Axelrod 1984; Axelrod and Dion 1988) and other researchers (Genesereth, Ginsberg and Rosenschein 1986) using the Prisoner's Dilemma game. The Prisoner's Dilemma game is a non zero-sum game between two players. In the game, each participant plays either C (Cooperate) or D (Defect), according which points are earned as listed in Table 1.

Table 1 Payoff matrix of the Prisoner's Dilemma game

A \ B	C	D
C	A:3 B:3	A:0 B:5
D	A:5 B:0	A:1 B:1

The main features of the game are:
1) Whatever the hand of the other player is, it is more profitable for a player to defect.
2) The point earned when both defect, or the average point earned when one defect whereas the other cooperates, is lower than that for both cooperate.

Axelrod showed us that:
1) Cooperation emerges if the expectation exists for both to play repetitively in the future (Iterated Prisoner's Dilemma game).
2) The best strategy for a single round game is to defect, and no cooperation can emerge in such a situation.
3) "Tit for Tat (TFT)" strategy, which mimics the opponent's latest hand, is effective for the Iterated Prisoner's Dilemma game.

However, we are not always interacting with the same people. On the contrary, we are forced to interact with new people everyday. Our society collapses if the best strategy is always to defect for such encounters. Fortunately, most people can cooperate with new people. As we know, reputations or confidences are regarded as essential or vitally important not only in private life but also in business. This suggests that the social sanction mechanism by disclosing information works effectively for the emergence of cooperations. The same mechanism may also work for the Prisoner's Dilemma game. With the above considerations in mind, the interaction of people is modeled as a kind of "deal" equivalent to the Prisoner's Dilemma game. The condition of the game is that the contract histories of all the agents are recorded and disclosed to the public.

First, our model of a deal society is explained briefly (Ito and Yano 1995). An agent moves around randomly in a two-dimensional space (lattice) of $N \times N$. That is, an agent moves to one of the neighboring sites with probability r respectively, or stays at the same place with probability $(1 - 4r)$. Agents occupying the same location can make a deal. The deal is equivalent to the Prisoner's Dilemma game, with the payoff matrix listed in Table 2. Note that the deal fee of 2.5 is subtracted from the corresponding payoff in Table 1. In the deal, each agent decides either to cooperate or to defect, applying his deal strategy algorithm to the contract history of the opponent. All the contract histories, (i.e., timestamps, the players and the hands played)

are made open to the public, and are accessible by any agent.

Table 2 Deal profit/loss matrix

A \ B	C	D
C	A:0.5 B:0.5	A:-2.5 B:2.5
D	A:2.5 B:-2.5	A: -1.5 B: -1.5

An agent has an initial asset of A_0, and the payoffs in Table 2 are added each time it makes a deal. The agent whose asset becomes less than zero is bankrupted, and is deleted from the system. On the other hand, the agents whose asset becomes larger than $2A_0$ bears a child and imparts the amount A_0 to the newborn child. The child inherits the deal strategy of his parent, but does not inherit the contract history.

Our questions in this paper are the followings:
1) What strategy prospers best (makes the largest offsprings) in this artificial deal society.
2) If cooperative strategies can survive at all, how cooperation can emerge socially, and what is the condition for its evolution?

Deal Strategy Algorithms

Deal strategy is an algorithm which calculates what hand to play next using the contract history of the opponent. As the history of deals will be known to other agents, the strategy should take into account not only the expected profit of the current deal, but also the effect the play of this deal might have on future deals.

First, strategies are classified as cooperative or uncooperative. A cooperative strategy always plays C in a society of cooperative agents. A strategy that is not cooperative is an uncooperative strategy. Note that even a cooperative strategy may play D against a cooperative strategy, if uncooperative strategies co-existed in the system. Cooperative strategies are further classified into retaliative and nonresistant strategies. A retaliative strategy tries to defeat uncooperative strategies by playing D against them. The strategies we are mainly interested in are retaliative ones.

The following are a few basic strategies.

1. Nonresistance (CCC): Always cooperate.

2. Tit for Tat (TFT): Mimic the opponent's play in his latest game. Cooperate, if the opponent's history is empty.

3. Random (RAN): Cooperate/defect with probability 1/2, irrespective of the opponent.

4. Exploiter (DDD): Always defect.

CCC and TFT are grouped as cooperative strategies. TFT is a retaliative strategy, but it does not work well as one. For, on finding that the opponent defected in the latest deal, it cannot guess why the opponent defected. The possible reason for defections are:
a) to exploit the others,
b) to defend himself anticipating the opponent's defection,
c) to punish an uncooperative behavior of the opponent.

An agent that retaliates without considering the deep intention of the opponent will, at the next deal, get retaliations from other agents, eventually leading to a storm of retaliations.

As a first step for improvement, we try to avoid the case b). If it is found that both agents defected in the opponent's latest deal, then we regard the opponent's defection as a self-defense, and forgive the opponent's behavior. Hence, we go one record back in the opponent's history, and examine the opponent's hand in the next latest deal. Thus the revised version of Tit for Tat is obtained.

5. Revised Tit for Tat (TT3): If both agents defected in the opponent's latest game, then go one recored back in the opponent's history, and apply this algorithm again. Otherwise, mimic the latest hand of the opponent. Cooperate if the opponent's history is exhausted.

Next we try to incorporate the case c). Suppose our opponent had found, in his latest deal, that his opponent had played D in the previous deal. This does not automatically mean that our opponent has the right to retaliate at once. As just explained, this would cause a storm of retaliations. The strategy we adopted is to think what hand we ourselves would have played if we were in our opponent's place in the deal. If our opponent played D in the situation where we would have played C, then we are allowed to retaliate on the opponent. In this way, a strategy with a reflective ability is obtained.

6. Reflective (REF): In the record of the opponent in the latest game, if both the agents defected, or if the opponent defected and this algorithm also suggests D in the opponent's position, then go one recored back in the opponent's history and apply this algorithm again. Otherwise mimic the opponent's latest hand. Cooperate if the opponent's history is exhausted.

The above strategy is too tolerant. Even if the opponent played C to those who was sure to play C, it does not necessarily mean that the opponent is worth cooperating with. The check of comradeness (if the opponent is cooperative or not) can be made a little more rigorously. The next latest record must be consulted, if both agents cooperated in the opponent's latest record. This gives a revised version of the reflective strategy.

7. Revised Reflective (RF1): In the record of the opponent in the latest game, if both the agents defected, or if both the agents cooperated, or if the opponent defected and this algorithm also suggests D in the opponent's position, then go one recored back in the opponent's history and apply this algorithm again. Otherwise mimic the opponent's latest hand. Cooperate if the opponent's history is exhausted.

The strategies described so far examine only the latest record, and decide his next hand. Strategies are also possible that examine all the opponent's records

in deciding the comradeness of the opponent.

8. Self Centered (SLC): If in all the opponent's records, there exists a case where the opponent played D and this algorithm suggests C, then defect. Otherwise cooperate.

9. Selfish (SLF): If in all the opponent's records, there exists a case where the opponent played the hand other than the way this algorithm suggests, then defect. Otherwise cooperate.

SLC tries to exclude all the strategies less conscientious than it is. SLF, on the other hand, tries to exclude all the strategies that differs even a little from it. Note that each of the reflective strategies (REF, RF1, SLC and SLF) uses its own algorithm recursively to check the opponent's comradeness, and can correctly avoid playing D against the same strategy.

Simulation Results

The behavior of the strategies introduced in the previous section are investigated in computer simulations. The parameters adopted are as follows:

Size of the two-dimensional space: $N = 6$.
Probability of an agent's movement: $r = 1/5$.
Initial asset: $A_0 = 20$.

One-to-one Matches

First, we examined the relative strength of various strategies by matching each strategy one-to-one. Matches between cooperative strategies lead to both cooperating, so our interest focuses on matches between cooperative and uncooperative strategies. The qualitative results are given in Table 3, where "win" means that the cooperative strategy wins, "defeated" means it is defeated, and "mut. (mutual) decay" means both the strategies decay in the battle. Note that even if an uncooperative strategy wins at first, it is eventually destined to decay after the exhaustion of the victims.

Table 3 One-to-one matches between various strategies

	DDD	RAN
CCC	defeated	defeated
TFT	mut. decay	mut. decay
TT3	win	mut. decay/win
REF	win	win
RF1	win	win
SLC	win	win
SLF	win	win

Competition Among RAN and Two Retaliative Strategies

To examine how two retaliative strategies cooperate under the existence of uncooperative strategy, RAN and two retaliative strategies are matched together. In all the cases, RAN is destroyed by the retaliative strategies, but the difference of behaviors between two retaliative strategies which is made clear during the battle against RAN causes the following:

1) SLC tries to destroy all the strategies that plays less conscientious (i.e., plays D where SLC might have played C), and SLF tries to destroy all the strategies that do not play as SLF does.

2) TFT, TT3, REF and RF1 do not have an effective countermeasure for this attack and are destroyed by SLC or SLF.

3) TFT, TT3, REF and RF1 are mutually tolerant and can coexist peacefully.

Competition among All the Strategies

When all the strategies introduced in the previous section are made to compete together, uncooperative strategies (DDD and RAN) are quickly destroyed. Then exclusive SLF fights against all the remaining strategies and is destroyed. The strategies TFT, TT3, REF and RF1 can coexist, but not with SLC. The strategy CCC can coexist with both.

Eventually the system goes to one of these three states:

(1) SLC wins and destroys all the strategies except CCC.

(2) The group TFT, TT3, REF and RF1 wins and destroys SLC.

(3) Uncooperative strategies are destroyed at the early stage. CCC grows, and SLC and TFT/TT3/REF/RF1 group make a truce and prosper together.

An example of the case (2) is given in Fig. 1.

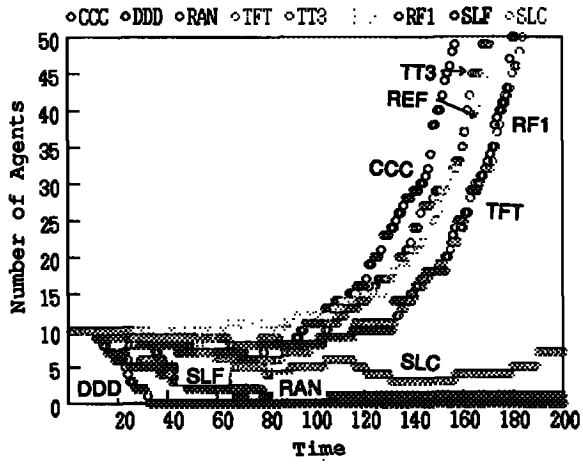


Fig.1 The competition of all the strategies (2) TFT, TT3, REF and RF1 group wins

The results of the simulation are summarized as follows:

(1) TFT does not work well in our system, for it causes the storm of retaliations.

(2) A reflective strategy avoids fighting between comrades by using its own strategy in inferring why the opponent acted that way, and succeeds in destroying

uncooperative strategies.
(3) Too rigorous a check of comradeness makes enemies of all the other strategies, whereas, too much tolerance allows exploitation by enemy strategies. Good strategies should take the golden mean of the two.

Mutation of the deal strategy algorithm

To introduce mutations into strategy algorithms, it is necessary to express algorithms in some machine language. The idea to define an abstract machine came from Tierra of Ray (Ray 1992). It is a machine which calculates the hand it should play, from the contract history of the opponent.

Strategy Algorithm Calculator

The calculator has 8 (typed) registers, and 6 opecodes (all without operands). Moreover, the calculator has an infinite length stack, and can execute reflective (self recursive) calculations. Inputs to the calculator are the (current) time, the pointer to the opponent, and the deal strategy algorithm. The calculator returns the next hand (either C or D) as the result of a calculation. The registers, opecodes, and the flow of the calculations are shown in Fig. 2 using the syntax similar to C++.

First the calculator loads the contract record of the opponent's latest deal into the history register. Suppose the opponent in the next deal is M, and M's latest deal is D_0 (i.e., the contract record loaded is of the deal D_0), and the opponent of M in D_0 is Y. The calculator processes the strategy algorithm step by step. The code LC, LD loads 'C' or 'D' into the hand register. BM, BY and SL are branch instructions. BM branches according to the hand of M in D_0. BY branches according to the hand of Y in D_0. SL branches according to the hand which this algorithm suggests, if it is in place of M in D_0. TP goes one record back in the opponent's contract history, and applies the algorithm again. The process terminates either when the algorithm reaches an end, or it runs out of the contract history. As is seen from the flow, the hand is undefined if the algorithm terminates without ever executing LC or LD. In such a situation, the calculator is supposed to randomly output a hand, i.e. either C or D with probability 1/2. Hence the random strategy can be expressed by the algorithm of length 0, NIL.

A set of six instructions seems too restrictive for expressing various kinds of deal strategies. However, all the strategy algorithms described before are expressible with this instruction codes, which are shown in Table 4. Of course, the system has many limitations. The biggest one is that it has no counters, and cannot calculate some of the seemingly effective algorithms. For example, algorithms like "Play C if the opponent played C more often than D" are not expressible. Moreover, in this system only infinite loops are allowed, and the number of (maximum) iterations cannot be specified.

Mutations

In the model, the agent whose asset becomes more than $2A_0$ bears a child, and bequeaths his algorithm to the child. Mutations are introduced as replication errors in this algorithm inheritance process. To be concrete, the mutation consists of the following three components:
Deletion of codes: A code in the algorithm is deleted with the probability p_m. If the deleted code is a branch instruction, one of the code sequences corresponding to the branch is now unnecessary, hence it is also deleted (Fig. 3a).
Insertion of codes: A instruction selected randomly from the six code set is inserted at any place in the algorithm with the probability p_m. If the inserted code is a branch instruction, the algorithm now need two code sequences corresponding to the branch. Hence one of the code sequences after the branch is set to NIL (Fig. 3b).
Replacement of NIL (NOP): Due to an insertion of a branch instruction, the code NIL often enters into the code sequences, which is unfavorable for the system performance. Hence with a larger probability $p_r (>> p_m)$, NIL is replaced with a code randomly selected from the code set (Fig. 3c).
Simplification of redundant code: Redundant codes brought about by the above mutation processing are deleted, as shown below.
o If the code LC or LD appears in the code sequences more than once without a branch, the effect of the first code is overridden by the later code. Hence a code sequence after the branch until the last LC/LD is deleted (Fig. 3d).
o If the same branch instruction succeeds, the same path is always to be followed at the second branch. Hence such a sequence is changed to that without a branch (Fig. 3e).
o The code after TP will never be executed. Hence the code sequences after TP are also deleted.
Eventually, the algorithm generated as above is inherited to the child.

It seems that a small opecode set is essential for the evolution of strategies (algorithms). In a system with a large opecode set, the random mutation of algorithms mostly leads to a "meaningless" offsprings with little or no adaptability. The importance of a small opecode set is also emphasized by Ray (Ray 1991) in his Tierra implementation. The large descriptive power of the genetic code cannot fully be utilized (at least) at the early stages of evolutions.

The Condition for the Evolution of Strategies

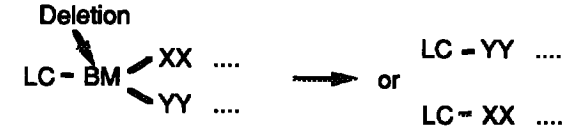
The pay-off table in Table 2 shows that no strategy can survive long if it cannot cooperate among themselves. Even though it can temporarily earn a good point by exploiting others, it has to decay by itself after the exhaustion of victims. In order for any strategy to

```
/* type definition */
enum hand {C,D}; // enumerate type {C,D}
typedef int time; // time type
class agent; // agent type
typedef int instruction-pointer ip;
struct register{  // registers
  hand h;  // output register
  time t;
  agent* a;
  history his;  // deal history
  instruction-pointer ip;
};
struct history{
  time time;
  agent agent;  // the other agent
  hand my;     // hand of this agent
  hand you;};  // hand of the other agent
class agent{
  ... ;        // agent's private data
  history history;
};
/* opecode */
enum opcode {
    LC;  // loadh-c;  load C to h register
    LD;  // loadh-d;  load D to h register
    BM;  // branch-m; branch acc. history.my
    BY;  // branch-y; branch acc. history.you
    SL;  // apply self; apply its own algorithm
         // to history
    TP;  // decriment time by 1, and goto top;
};
typedef al-codes List-of opcode;
      // like LISP's list
/* abstract machine */
am(time t, agent a,  al-codes al)
/* current time, the other agent,
deal strategy algorithm */
{
   instruction-pointer ip=0;   /* set C or D randomly */
   hand h=random_selection('C','D');
/* load  agent(a)'s (at the time t) latest deal history
      to his registers  */
   history his=latest_rec(a,t);
   for(;;){   // fetch opecode
     switch (c=get_opcode(al,ip++)){
       case lc: h='c';braek;
       case ld: h='d';break;
       case bm: (his.my=='c')?
          ip=branch1, ip=branch2 ;break;
       case by: (his.you=='c')?
          ip=branch1, ip=branch2 ;break;
       case sl: (am(t-1, his.agent, al)=='c')?
          ip=branch1, ip=branch2 ;break;
       case tp: t--;ip=0;break;
     }
   }
   return h;
 }
```

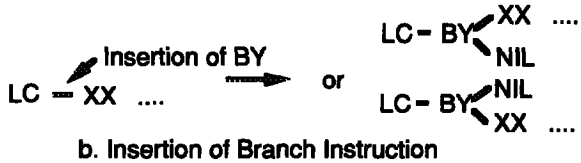
Fig.2 Deal strategy algorithm calculator

Table 4 Codes for the strategy algorithms

CCC	(LC)
TFT	(LC BM (LC) (LD))
RAN	NIL
DDD	(LD)
TT3	(LC BM (LC) (BY (LD) (TP)))
REF	(LC BM (LC) (BY (SL (LD) (TP)) (TP)))
RF1	(LC BM (BY (TP) (LC)) (BY (SL (LD) (TP)) (TP)))
SLC	(LC BM (TP) (SL (LD) (TP)))
SLF	(LC BM (SL (TP) (LD)) (SL (LD) (TP)))

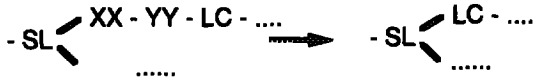


a. Deletion of Branch Instruction

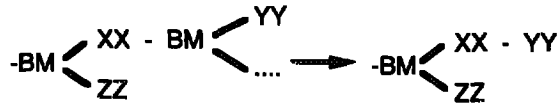


b. Insertion of Branch Instruction

c. Selective Replacement of NIL



d. Deletion of codes from branch to LC



e. Deletion of redundant Branches

Fig.3 Algorithms for Mutation Processing

survive, it must make positive points by itself.

It was our hope that a simple strategy could bring forth strong/robust strategies through the mutation process. However, a very simple strategy such as CCC will be destroyed by exploiters before it ever learns to retaliate against them. Hence the emergence of strategies is investigated starting from Tit for Tat (TFT), the simplest strategy with retaliative abilities.

The system started from TFT generates, through mutation, various strategies, comprising not only cooperative ones but also uncooperative ones. As far as uncooperative strategies are few in number, their activities are suppressed by retaliative strategies including the original TFT. However, the long duration of a peaceful era brings forth a variety of cooperative nonresistant strategies. Especially, the growth of CCC, which is optimal in a peaceful society, is unavoidable.

However, once the population of retaliative strategies decreases below that of nonresistant strategies, a disaster may occur. That is, if exploiter strategies start to grow making a prey of nonresistant strategies, retaliative strategies, which are now few in number, cannot stop the growth of the exploiters. As a result, first the population is dominated by the exploiter strategies and all the agents are forced to defect, leading to the collapse of the society.

These processes are often found in ecological and human societies. It is not surprising that the same phenomena are also found in our system. The reason that the life on earth and human societies can survive through these calamities are, at least partly, due to the geographical isolation of the system. That is, a geographically isolated area can be immune from the growth of the dominant species, from the takeover of other species by the dominant species, and from the eventual extinction of all the species by the self-decay of the dominant species. Hence, to simulate the geographical isolation, the size of the world is made larger, and the migration speed of the agents is made lower.

Another problem to be solved is a fruitless divergence of strategies in our system. Mutations generate various strategies, some of which consumes enormous amounts of computer time in meaningless calculations. Hence the introduction of a computational cost is essential. Concretely, a cost of C_{am} per step is charged for the execution of the strategy algorithms at each deal. Moreover, to discourage the intrusion of codes which will scarcely be executed, the cost C_{st} per code is charged in inheriting the strategy to the child.

So far, the growth of the population is left uncontrolled. However, the target here is the evolution of new strategies. Hence, the simulation must be continued until a sufficient number of new strategies evolve through mutations. Computation becomes intractable if the exponential growth of the population is left uncontrolled. Hence the concept of "a favorable population density" is introduced in the system. If the population density exceeds this limit, the death rate is made to increase automatically, until the population growth is suppressed by the death of the agents. In summary, the parameters of the system adopted are shown in Table 5.

Table 5 The adopted system parameters

Probability of agent migration p_{rw}	1/40
Size of the lattice N	15
Mutation probability p_m	0.01
NIL replacement probability p_r	0.2
Caluculation cost C_{am}	0.005
Algorithm inheritance cost C_{st}	0.1
Favorable population density P_a	2.0

Evolution of the strategies

The evolution of strategies are examined starting from TFT. The system causes the generation and the natural selection of various strategies, experiencing a number of regionally restricted growth and extinctions. Some of the strategies thus generated may be more robust than TFT, and others may be less robust. To examine the robustness of the system as a whole, the following experiments are executed. First, starting from TFT, the system is evolved freely for the 20,000 steps of time. The behavior of the system through the evolution is shown in Fig. 4, where the total population is shown as a bold line, and the average points each agent received are shown as small dots.

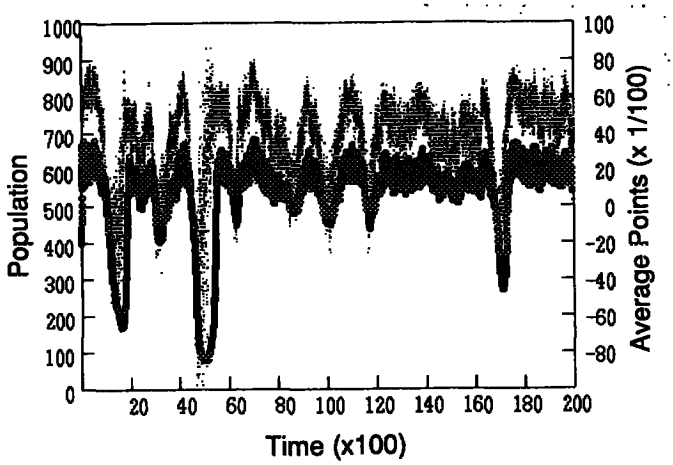


Fig. 4 Population changes for free evolution of 20,000 steps

To evaluate the robustness of the system, the system after the evolution of 20,000 steps (called S_0 hereafter) is matched with half the number of DDD. In comparison, systems consisting of the same number of TFT, TT3, and REF (called S_{TFT}, S_{TT3}, S_{REF} respectively), are matched with half the number of DDD. Before matching, all the contract history is initialized, and the agents are rescattered randomly in space. The population changes in these matches are shown in Fig. 5.

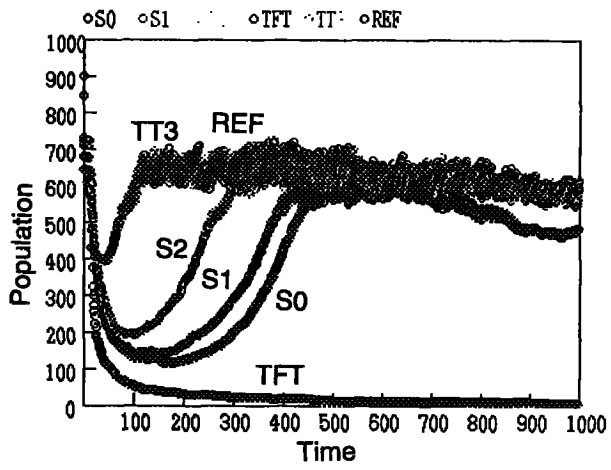


Fig. 5 Matches between S_0, S_1, S_2, S_{TFT}, S_{TT3}, S_{REF} and DDD.

Now, "a robustness index" is introduced for a more quantitative comparison. The population of the system fluctuates around 550-600 (determined by the favorable population density $P_a = 2.0$). In the competition with half the number of DDD, the system first undergoes a population decrease, and the subsequent recovery to the ordinary level, if it ever succeeds to defeat DDD. Hence, in the competition with DDD, the area of the triangle built from the point at which the population goes down below 500, the point at which the population recovers 500, and the minimum point of the population, is calculated. The smaller this area is, the more robust the strategy is. A robustness index (RI) is defined as the inverse of the square root of this area (expressed as a percentage). Note that the robustness index of 0 means that the system cannot recover the population of 500, while)100 means either that the triangle is too small, or the population never decreased below 500.

The evolution experiments starting from TFT are conducted ten times. In one of the experiments, the system becomes extinct through the sudden outburst of uncooperative strategies. For the remaining nine trials, the robustness of the system is tested at two points of time. One is just after the free evolution of 20,000 steps, i.e., the rightmost point in Fig. 3 (S_0), and the other is at 1,000 steps after the first match with DDD, i.e., the rightmost point in Fig. 4 (called S_1 hereafter). The robustness indices are measured for each system, and compared with that for S_{TFT}, S_{TT3} and S_{REF}. The results are given in Table 6.

Table 6 The robustness indices of various systems

	S_0	S_1	S_2	S_{TFT}	S_{TT3}	S_{REF}
RI	0.28	0.37	0.70	0	2.04	1.99

Judging from the result, S_0 is far behind S_{TT3} in robustness. This is mainly because S_0 contains many strategies that are nonresistant, or even uncooperative. However, S_0 is putting up a good fight, considering that S_{TFT} decays by itself. In fact, after the first battle with DDD (S_1), most of the nonresistant strategies are destroyed by the exploiters, and the system becomes a little stronger, which is confirmed in Table 6.

The natural question is: Is it true that the system can evolve faster if it is exposed to the attack of uncooperative strategies? To test the above hypothesis, an experiment is conducted in which the system is periodically given a vaccine of uncooperative strategies. To be concrete, a vaccine consisting of a fifth the number of DDD is injected into S_0 every 100 step for 10,000 steps of time. The system thus obtained is called S_2. The robustness index for S_2 is also given in Table 6.

The system S_2 becomes substantially more robust from S_0 and S_1, but still less robust than S_{TT3}. This is mainly due to the liberality of the system which admits the coexistence of various strategies. In fact, whereas the total population of S_2 in nine experiments is 5,371, the number of strategies amounts to 1,829, suggesting a enormous divergence of strategies in the system.

In Table 7, the top ten strategies are shown with their population in percentage. The fact that TFT comes in first, CCC in seventh, and DDD in tenth indicates the liberality of the society. It is surprising that such a system has advanced half the way in robustness to TT3. Next, the robustness index for each strategies in Table 4, is measured. As to the competition with half the number of DDD, most of these strategies earn nearly the same point. Hence these strategies are also matched with the same number of RAN (random) strategies. It is interesting to note that simple TT3 makes a hard fight, compared to REF. This is because the detection of RAN demands a more subtle strategy which TT3 seems to lack. Actually, strategies No. 3, 4, and 8 are more robust than TT3, and ranks nearly the same as REF in robustness.

Conclusions

The Prisoner's Dilemma game was formulated by Rosenschein (Rosenschein and Genesereth 1985) as a deal in the society of agents. The evolutionary mechanism was introduced by Lindgren (Lindgren 1992) to evaluate the algorithms. The biggest difference between our research and the previous researches is that in our model an agent makes a deal each time changing the other party of the deal. Of course, a defection is the optimal strategy in this setting, so we introduced the social sanction mechanism by disclosing information, i.e., the contract histories. This is expected to model our society, where reputation or confidence works effectively to socially sanction unfair deals.

To search for the optimal strategies, we have developed some deal strategies and matched them in various environments. As a result, we found that TFT does not work well in our system, but revised retaliative strate-

Table 7 The robustness indices for the top ten strategies in S_2

rank	strategies	population (%)	RI (DDD)	RI (RAN)
1	(BM (LC) (LD)) [TFT]	2.20	0	0
2	(BM (LC) (BY (LD) (TP)))	1.51	0.79	0.26
3	(BM (LC) (BY (LD SL (LD) (TP)) (LD TP)))	1.28	14.29	1.28
4	(BM (LC) (BY (SL (LD) (TP)) (TP)))	0.97	)100	1.41
5	(BM (LC) (LD TP))	0.89	0	0
6	(BM (LC) (LD BY (LD) (TP)))	0.89	0.55	0.25
7	(LC) [CCC]	0.89	0	0
8	(BM (LC) (BY (LD SL (LD) (LC)) (TP)))	0.86	)100	1.14
9	(BM (LC) (BY (SL (LD TP) (TP)) (TP)))	0.80	0	0
10	(LD) [DDD]	0.63	0	0
TT3	(LC BM (LC) (BY (LD) (TP)))	0.20	2.04	0.33
REF	(LC BM (LC) (BY (SL (LD) (TP)) (TP)))	0	1.99	1.43

gies can expel uncooperative strategies. We also found that good strategies should be neither too rigorous nor too tolerant, that reflective algorithms are effective to avoid fighting between comrades.

Next, using genetic algorithm techniques, we investigated how cooperative strategies socially emerge. We have shown that the system started from a naive TFT can, through mutation and natural selection process, brings forth strong/robust strategies. The system evolves alternating comparatively long eras of peace and short periods of disturbances. Nonresistant strategies are the most advantageous in peaceful time, but when the population of the nonresistant strategies dominates that of retaliative strategies, uncooperative strategies increases and brings about the era of disturbances. Through these alternations the system becomes substantially stronger, where various strategies cooperate to maintain the peace. The fact that the evolution of an ecological system often involves such a process is also described in Trivers (Trivers 1985). Hence it is not too farfetched to suppose that the evolution of cooperation in human societies also might have undergone such a process.

The diversity of strategies found in our system may be regarded as weakness when it has to fight against uncooperative strategies. However, the adaptability of the system to indefinitely varying environment can be sought in the diversity of a system. In fact, it is said that a ecological system optimally adapted to the isolated environment is very fragile, once the environment are changed externally. It is a great interest for us to examine how the diversity of strategies contributes to the robustness of the system.

The research on the cooperation of the autonomous agents has so far centered on the investigation of the rules agents should follow to achieve the global goals. The gain or loss of each of the agents has not been considered seriously. As a word "autonomous" suggests, however, an agent should be free to break the rules if he regards it necessary. Otherwise the agent is just a robot doing what is told (programmed) to do. For an agent to be truely intelligent, it must have a real autonomy, i.e., do whatever is profitable for itself. Hence, the behavioral norm of autonomous agents must be investigated. We hope our work is one step for the above goal.

References

Axelrod, R. and Hamilton, W. D. 1981. The Evolution of Cooperation. *Science* 211: 1390-1396.

Axelrod, R. 1984. *The Evolution of Cooperation.* Basic Books Inc.

Axelrod and Dion, D. 1988. The Further Evolution of Cooperation. *Science* 242: 1385-1390.

Genesereth, M. R., Ginsberg, M. L. and Rosenschein J. S. 1986. Cooperation Without Communication. Proc. 5th Nat. Conf. on Artificial Intelligence, 51-57.

Ito A. and Yano H. 1995. The Optimal Deal Strategies Under the Disclosure of the Contract History. *Journal of Japanese Society for Artificial Intelligence* 10(2): 271-278 (in Japanese).

Lindgren, K. 1992. Evolutionary Phenomena in Simple Dynamics. *Artificial Life II*: 295-312. Langton (eds.), Addison Wesley.

Ray, T. S. 1991. An Approach to the Synthesis of Life. *Artificial Life II*: . Langton et. al. (eds), Addison-Wesley.

Rosenshein, J. S. and Genesereth, M. R. 1985. Deals Among Rational Agents. In Proc. 9th Int' Joint Conf. on Artificial Intelligence (IJCAI'85), 91-99.

Trivers, R. 1985. *Social Evolution*, Benjamin/Cummings.