# 利己的なエージェントの社会におけるつきあい方戦略の進化

伊藤 昭† 矢野博之†

自己の利益をのみ追求するエージェントの社会では，どのようにして協調が発現するのであろうか．我々は，過去の対戦履歴が公開されるという条件の下で，エージェントが囚人のジレンマと同型の対戦を，相手を次々と替えながら行わねばならないとき，どのような対戦戦略を採用すればよいのかを調べてきた．今回は，どのようにして協調的戦略が社会的に発現（進化）するのか，またそのための条件は何かなどを，遺伝的アルゴリズムの手法を用いて調べる．我々は，まず対戦戦略アルゴリズムを抽象計算器の上で定義する．次に，エージェントは対戦利益に応じて子を生成できるものとし，また子エージェント生成に際しては，戦略アルゴリズムに突然変異を導入してその進化を促す．その結果，最初単純なしっぺ返し戦略TFTから出発して，系は非協調的戦略を含む様々な戦略を持つエージェントを生成するが，生存競争の中でより強い協調的な戦略が成長してくることを示す．

# Evolution of Interaction Strategies in a Society of Selfish Agents

AKIRA ITO† and HIROYUKI YANO†

The evolution of cooperation in a society of agents is investigated. Each agent is made to engage in matches equivalent to the "prisoner's dilemma" game repetitively, each time changing the opponent of matches. The condition is that the match histories of all the agents are disclosed to the public. First, we define match strategy algorithms on an abstract machine. Each agent can bear a child according to the amount of profit he earns through the matches. Random mutations are introduced in inheriting the match strategy to the child. The system started from simple "Tit for Tat" strategy produces various strategies including non-cooperative ones. However, through the natural selection mechanisms, the robust and cooperative strategies grow up, and the society becomes substantially stronger.

## 1. はじめに

自律したエージェントの社会では，各エージェントは自己の行動規範に従い，他のエージェントとつきあい（interact）ながら自己の利益を追求する．このような利己的なエージェントの社会では，どのようにして協調が可能になるのであろうか．Axelrod[1)~3)]，Genesereth[4)]らは，囚人のジレンマゲームを用いてこの問題に取り組んだ．囚人のジレンマゲームとは，2人の参加者による非ゼロ和ゲームであり，次のように定式化される[5)]．参加者は，C（協調：Cooperate），またはD（裏切り：Defect）のいずれかの手をだすことができ，双方の手の組合せにより**表1**のような得点を得る．

このゲームの特徴は，相手の手がどのようであっても，自分としてはCを出すよりもDを出す方が有利であること，しかしながら，双方がDを出し続けたときの得点（表1では，1点）や，互い違いにC，Dを出したときの平均得点（2.5点）よりも，双方が協調してCを出したときの得点（3点）の方が高いということである．

Axelrodが明らかにしたことは，「今後とも同じ相手と引続き対戦する」という予測があれば協調活動が出現すること，逆に「1回限りの対戦」であれば，双方の最善の戦略は「裏切り」であり，協調行動の出現は困難なことであった．しかしながら，実社会ではつねに同じ相手と無限回の対戦（つき合い）を行うわけではない．事実我々は，多くの場合1回限りの対戦を行うことで，社会的な生活を営んでいる．また，1回限りの対戦であればつねに「裏切り」が行われるというわけでもない．むしろ実際の商社会では，「商人は信用が第一」といわれるように，社会的信用が重視されており，情報の公開による社会的制裁機構が有効に働いているものと思われる．

我々は，このような情報の公開による社会的制裁機構をモデル化するために，対戦履歴の公開という条件下で，エージェントが囚人のジレンマと同型の対戦を，

† 通信総合研究所関西支所
Kansai Advanced Research Center, Communications Research Laboratory

表1 囚人のジレンマゲーム得点表
Table 1 Payoff matrix for 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 |

相手を次々と替えながら行わねばならないとき，どのような対戦戦略を採用すればよいのかを調べてみた．その結果，前回の報告[6]では，以下のことを明らかにした．

(1) 対戦履歴公開下では，裏切り行為は自己の信用を落とし，他のエージェントから排斥される結果をもたらす．

(2) Axelrod らの場合に有効であった単純な「しっぺ返し」戦略は，味方同士でのしっぺ返しの応酬を引き起こすため，もはや有効ではない．

(3) 相手が過去にどうしてそのような手を出したのかを推測し，協調できる相手とそうでないものとを区別し同士討ちを避けることで，自ら裏切りを行う戦略を駆逐することができる．

しかしながらそこでは，著者らが考案したいくつかの戦略を相互に対戦させることでその強さを評価しただけであり，次のような問題点が残されていた．

(1) 多様な，変化する環境の下でも，うまく振る舞える戦略に共通する性質とは何か．

(2) 巧妙な形で裏切りを行うことで，利益を得るような戦略が優勢になることはないか．

(3) 我々が戦略を与えるのではなく，エージェント自身が最適な戦略を学習することはできないか．

そこで，このような問題点を検討するため，エージェントの社会に進化の機構[7),8)]を導入し，つき合い方戦略がどのように進化していくのかを調べてみた．

以下では，本論文に必要な範囲でエージェント社会のモデルを説明する[6)]．エージェントは，2次元格子上の「世界」をランダムに歩き回っており（隣接格子点への移動確率 $p_{rw}$），同時に同じ場所にいるエージェント同士は「対戦」を行うことができる．対戦は囚人のジレンマゲームと同型で，双方はC（Cooperate），またはD（Defect）の手を出すことができる．対戦において，双方が出した「手」は，公表されて（永久に）記録として残される．また，このときの対戦履歴は，以後どのエージェントからも自由に参照することができる．

エージェントは，最初に自己資産 $E = E_0$ を持ち，対戦の度に以下で与えられる対戦利益（損出）が資産に加えられる．

対戦利益 = 表1の得点 − 対戦費 (固定)
− 計算費 (処理量に比例)

自己の資産が0になったエージェントは，死滅し系から削除される．逆に自己の資産が一定（$E = 2E_0$）以上になると，エージェントは自己の資産のうち $E_0$ を分け与えて子エージェントを作る．このとき，子エージェントは親の対戦戦略を継承するが，対戦履歴は継承しない．なお，子エージェントからその親エージェントを調べる方法はないものとする．

このような進化の機構のもとでは，最適な対戦戦略とは，最も多くの得点を稼ぐことのできる，したがって最も多く子孫を残すことのできる戦略として定義される．

## 2. 対戦戦略アルゴリズム計算器

戦略の進化を議論するためには，まず戦略を記述するための枠組みが必要である．そこで我々は，Ray[8)]の Tierra にならい，対戦戦略アルゴリズム計算器を定義することから始める．対戦戦略アルゴリズム計算器は，エージェントが次に対戦することになる相手の過去の対戦履歴から，自分が次に出すべき手を決める抽象マシンである．このアルゴリズム計算器は，8個の（型付きの）レジスタ（内部記憶）と，オペランドを持たない6個の命令セットを持つ．ただし，各命令は必ずしも単純なものではなく，通常の CPU の命令セットからいえば，かなり複雑な処理を表しているものも含まれる．さらに，計算器は1個の無限長スタックを持つことで，再帰的な計算能力を備えている．

計算器への入力は，（現在の）時間，相手エージェントへのポインタ，および対戦戦略アルゴリズムであり，計算器は計算の結果として次に出すべき手（C またはD）を返す．対戦戦略アルゴリズム計算器の概念図を**図1**に，また計算器のレジスタ，命令セット，処理フローを C++の構文を用いて記述したものを**図2**に示す．

計算器は，相手エージェントの直前の対戦記録を his（history）レジスタにロードする．次にアルゴリズムのコードを1つずつ読み込み，実行する．LC，LD は h（hand）レジスタにそれぞれ C，D をロードする．BM，BY，SL は分岐命令である．BM は相手エージェント自身の出した手，BY は直前の対戦で相手エージェントの相手が出した手，SL は直前の対戦で相手エージェントの立場にいたら自分が出したであろう手を調べて，その値が C か D かにより分岐する．命令 TP は相手の対戦記録を1つ前に遡って，初めからアルゴリズムを適用し直す．処理はアルゴリズムが終端に達

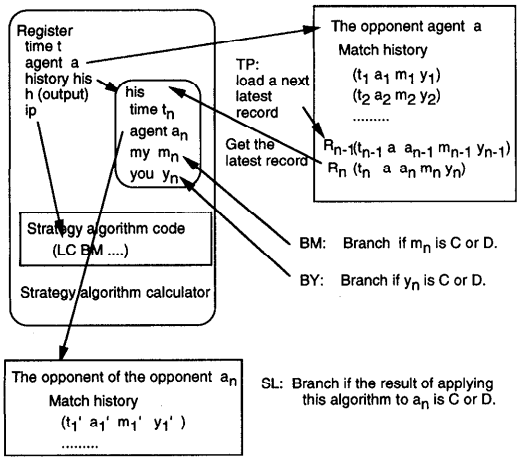



図1 対戦戦略アルゴリズム計算器の概念図
Fig. 1 A sketch of a match strategy algorithm calculator.

するか，または読み込むべき履歴がなくなるかすれば終了し，そのときのhレジスタの値を返す．

SLの処理では，次の対戦相手の最近の履歴を調べて，そのとき自分が相手の立場なら何を出していたかを計算することが必要だが，このときに用いる戦略は，今まさに定義しようとしている自己の戦略アルゴリズムである．これは自己の定義に自己を呼び出す再帰的定義であり，何らかの停止条件を必要とする．もちろん，停止条件を戦略として組み込むことも可能であるが，本計算器では「履歴がなくなれば処理を終了する」という形で，デフォルトの停止条件が組み込まれている．

処理フローから分かるように，命令LCまたはLDを1度も実行せずに計算器を抜けると，出力は未定となる．そこで，手が未確定の状態で計算器を抜けたときには，計算器はランダムな手を出力するものとする．これにより，単純なランダム戦略は，コード長0のアルゴリズムNIL（NOP）で記述されることになる．

主要な対戦戦略を本命例セットで記述したものを以下に示す．

(1) 無抵抗主義（CCC）
(LC)
つねにCを出し続ける．

(2) しっぺ返し戦略（TFT）
(LC BM (LC) (LD))
相手の直前の手を出す．ただし，履歴がなければCを出す．

(3) ランダム戦略（RAN）
NIL
対戦相手に関係なく，1/2の確率でCとDとを選ぶ．

(4) 完全搾取戦略（DDD）
(LD)
つねにDを出し続ける．

(5) しっぺ返し戦略II（TT3）
(LC BM (LC) (BY (LD) (TP)))

```
/* type definition */
enum move {C,D};  // enumerate type {C,D}
typedef int time;  // time type
class agent;  // agent type
typedef int instruction-pointer ip;
struct register{  // registers
  move h;  // output register
  time t;
  agent* a;
  history his;  // match history
  instruction-pointer ip;
};
struct history{
  time time;
  agent agent;  // the other agent
  move my;      // move of this agent
  move you;     // move 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. his.my
    BY;  // branch-y; branch acc. his.you
    SL;  // apply self; apply its own algorithm
         //                          to his.agent
    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, strategy algorithm */
  instruction-pointer ip=0;
/* set C or D randomly */
  move 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';break;
       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;
}
```

図2 対戦戦略アルゴリズム計算器の処理フロー
Fig. 2 A processing flow of a match strategy algorithm calculator.

相手の履歴から，双方ともDを出したものを除いて，相手の直前の手を出す．ただし，履歴がなければCを出す．

(6) 自省戦略（REF）
(LC BM (LC) (BY (SL (LD) (TP)) (TP)))
相手の履歴から，双方ともDを出したものと，「自分でも同じ状況ではDを出す」と思われるものを除いて，相手の直前の手を出す．ただし，履歴がなければCを出す．

協調戦略は，協調戦略だけの社会ではCを出し続ける戦略であり，そうでないものは非協調戦略である．協調戦略のうち，非協調戦略に対してはDを出すことで対抗しようとするものが戦闘的戦略である．(3)(4)は非協調戦略，それ以外は協調戦略である．また(2)，(5)(6)は，戦闘的戦略でもある．なお(6)は，協調性を判断するのに，自分が相手の立場ならどうするかを考える再帰的戦略であり，自分と同じ行動をとっている相手にはけっしてDを出さないことで，同士討ち（同じ戦略どうしが裏切り合う）を避ける能力を持つ．

6個の命令セットというのは，様々な対戦戦略の記述に対して，あまりにも制限がきついように思われるかもしれない．しかしながらこの命令セットで，前回の報告[6]で取り上げたすべての戦略を記述することができる．もちろん，本計算器には限界がある．その最大のものは，計算器が「カウンタ」を持たないことにある．このため，計算器はいくつかの有用な戦略を記述することができない．たとえば，「相手のこれまでに出したCの数とDの数とを数えて，Cの方が多ければCを，Dの方が多ければDを出す」というような戦略は，カウンタがないため記述できない．また，ループ（繰り返し）処理も，無限ループ（もちろん条件分岐により脱出は可能であるが）以外の，回数指定ループを記述することができない．

少数命令セットの採用は，本論文が目標としている戦略の進化にとっては本質的である．実際，戦略を突然変異（mutation）により進化させようとするとき，大きな命令セットを持つ言語では，突然変異の結果得られる戦略アルゴリズムのほとんどは無意味な命令列となってしまい，有効な戦略を効率良く生成することが困難になる．進化における少数命令セットの重要性は，RayのTierraにおいても強調されており[8]，あまりに大きな記述能力を持つ言語は，少なくとも進化の初期の段階では，その能力をうまく利用できないものと思われる．

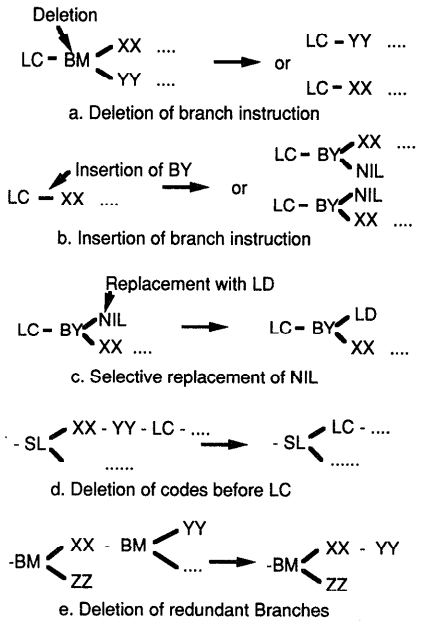



図3 戦略アルゴリズムの突然変異
Fig. 3 Mutation of strategy algorithms.

## 3. 戦略アルゴリズムの突然変異

モデルでは，エージェントは自己の資産が定められた値（$E_1 = 2E_0$）を超えると，子エージェントを生成し，自己の対戦戦略を子に伝達する．このとき，一定の確率でアルゴリズム複写誤りを導入することにより，突然変異を実現する．具体的には，突然変異は次の3つの部分から構成される．

**1. コードの脱落** 対戦戦略中の任意の命令コードが，確率 $p_m$ で脱落する．もし脱落したコードが分岐命令（BM，BY，SLのいずれか）であれば，元々後続命令は分岐に対応した2つのコード列を持つため，脱落処理後は一方のコード列のみを残し，他方はすべて捨ててしまう（図3a）．

**2. コードの挿入** 6個の命令セットからランダムに選ばれたコードが，それぞれの命令の間に確率 $p_m$ で挿入される．このとき，挿入されるコードが分岐命令であれば，挿入後は2つのコード列を必要とするため，一方の分岐に元々その後ろに続いていたコード列を結合し，もう一方の分岐をNILとする（図3b）．

**3. NIL（NOP）命令の置換** コードの削除，または分岐命令コードの追加によってNILが対戦戦略に出現するようになるが，NILの増加は好ましくないため，NILを化学反応の活性化部位のように考え，大きな確率 $p_r$ ($\gg p_m$) で，2章で述べたコードの挿入

(NILとの置換)を行う(図3c).

**4. コードの簡単化** このようにして生成されたコードの冗長部分の削除を行う.

- コード列の途中にLC(LD)が出現すると,それより前にあるコードの効果はLC(LD)により打ち消されてしまう.そこでこのようなコードは削除される(図3d).
- 同じ分岐命令が繰り返されたときには,すでに条件が確定しているため,分岐としての機能を果たさない.このような分岐命令も削除される(図3e).
- TPに続く命令列はけっして実行されない.したがって,このような命令も削除される.

結局,このようにして得られた戦略アルゴリズムが,最終的に子エージェントに引き継がれる.

## 4. 社会的進化の条件

システムは様々なパラメータを持っているが,そのすべての場合について協調的な社会が発現するわけではない.我々の行った予備実験によれば,多くのパラメータでは非協調的戦略が優勢となり,またいったん非協調的戦略によって占められてしまった社会では,協調的戦略が再び侵入することは非常に困難であった.そこで以下では,予備実験の結果をもとに協調的戦略が優勢となり得るための条件を検討し,我々のパラメータ選択の理由を説明する.

### 4.1 初期資産,対戦費

エージェントは,資産が初期資産$E_0$の2倍になると,自己の資産のうち$E_0$を分け与えて子エージェントを作る.進化は子エージェントが誕生するときのみに生じるので,初期資産が大きいほど社会は保守的・硬直的となり,逆に小さいほど進歩的・流動的となる.具体的には,初期資産が大きすぎると($E_0 \geq 40$)社会の変化への適応速度が遅く,たとえば非協調戦略の爆発に適切な時間で対応できない.また,小さすぎると($E_0 \leq 5$),生き延びが対戦相手などの偶然性に大きく依存してしまい,良いアルゴリズムを発現させる余裕がなくなる.このことから,効率良く協調的社会を発現させるパラメータとして$E_0 = 10$を選択した.

対戦での得点は表1に与られるが,固定の対戦費$C_m$をどう決めるかによって,系の性質は大きく変化する.系の平均得点は,もともとのゲームの平均得点(2.25点)−対戦費−平均計算費,となるが,計算費は単純な戦略は小さくなるように設定されている.さて,予備実験によると,非協調戦略が単独でも正の得点を稼げるような対戦費の設定では,いったん非協調戦略の社会が確立してしまうと,ここに協調戦略が侵入することは困難であった.

そこで我々は,ランダムな戦略の平均得点が負となる程度に大きな対戦費$C_m = 2.5$を課すことで,非協調戦略の淘汰を促した.このような対戦費の下では,特定の対戦戦略が一時的に相手を搾取して利益を得ることがあっても,長期的には協調しない限り正の得点を維持することはできず,自滅する以外の道はない.これは,その時点で存在する戦略の中で最適(最強)の戦略でも,必ずしも生き残れるわけではないことを意味している.ある戦略が生き残れるためには,自分たちだけで正の得点を得ることができなくてはならない.

### 4.2 地理的独立性

さて理想的には,「良い(協調的な),戦闘的(非協調戦略に対抗する能力を持った)戦略」が,単純な戦略から生じれば面白いのだが,残念ながらCCCのような単純な戦略は,非協調戦略に対する戦闘能力を学習するより前に,突然変異により生じた非協調戦略に滅ぼされてしまう.そこで,我々はしっぺ返し戦略TFTを出発点として,自然淘汰の中でより良い(強い)戦略が出現する過程を調べた.

最初TFTのみから出発した系は,突然変異により様々な戦略を生成する.この中で,非協調戦略も一定生成されるわけであるが,それが小数の間は,TFTおよび突然変異により生じた他の戦闘的戦略により,排除されてしまう.このようにして一定期間「平和な(非協調戦略の少ない)」社会が維持されると,戦闘能力を欠いた無抵抗な戦略が増加してくる.特に,CCCは計算費の点からいっても平和な社会では最適であり,このようなグループの成長は避けられない.

ところが,社会における戦闘的戦略の比率が(協調的だが)無抵抗な戦略の比率を下回るようになると,その脆弱性が露呈される.すなわち,どこかで非協調戦略が無抵抗戦略を搾取して成長し始めると,戦闘的戦略が少数派となった社会では,もはやその増加を押し止めることはできない.その結果は,まずすべての無抵抗戦略が滅ぼされ,次に戦闘的戦略が非協調戦略との戦いに敗れて全滅し,最後に単独では生存できない非協調戦略が自滅することになる.

このような過程は,人間社会でも,また生物の生態系でも実際に起こっていることであり,当然予想されるべき結果である.実際,地上の生命がこのような絶滅の危機を生き残ることができたのは,おそらく地理的分散性が寄与していた(少なくとも,大きな要因の1つであった)と思われる.すなわち,地理的に孤立していた地方では,特定の種の成長を,したがってそれ

に続く全種の絶滅をもたまたま免れることができ，次の世代を育てることができたものと思われる．このような状況を模擬するために，我々は格子サイズを大きくとること（$N=15$），エージェントの移動確率を小さくすること（$p_{rw}=0.025$）により地理的独立性を高める操作を行った．ちなみに，局所的に発生した戦略が全体に拡散するためのおおよその時間は，ランダムウォークの理論によれば，$(N/2)^2/(4p_{rw}) \approx 500$ ステップ で与えられる．

### 4.3 計算費，上限人口密度

一般に，突然変異は次々と複雑な，計算費のかかる戦略を生成する．したがって，無意味な戦略の爆発を防ぐためには，戦略アルゴリズムに計算費を導入することが必要となる．具体的には各エージェントが自己の戦略アルゴリズムを用いて次の手を計算するとき，1命令の実行あたり $C_{am}$ の費用を要するものとする．また，子エージェントが親から対戦戦略を継承するときにも，(命令数で数えた）アルゴリズムの長さあたり $C_{st}$ の費用を要するものとする．$C_{am}$ を大きくとると，複雑な戦略の生成がおさえられる．逆に $C_{am}$ を小さくとると，実際のシミュレーションに要する計算時間が爆発し，シミュレーション自体が困難になる．一方 $C_{st}$ の効果は，実際には使われない命令列を内部に蓄積することに対する制限である．

ここで採用した計算費 $C_{am}=0.005$ の効果は，おおよそ以下のとおりである．協調による1回の得点（計算費を除く）を0.5点とすると，これは100個の命令ステップの計算費に相当する．もし，他のエージェントの1つの対戦履歴を調べるのに10ステップの処理を必要とするとすれば，10個の対戦履歴を調べてしまうと，もはやエージェントは正の得点は稼げない．すなわちエージェントはそれよりも少ない履歴を調べるだけで満足して，次の手を決断しなければならない．実際の生物（人など）を考えると，$C_{am}$，$C_{st}$ はもっと小さくてもよいと思われるが，今回はシミュレーション時間の都合上（採用した値でも，実験を行うのに SUN SS20 で数十時間を必要とした）からその値を決定した．

社会全体でエージェントが正の得点を稼げば，必然的に人口は指数関数的に増大する．実験では，新しい戦略が突然変異により発生してくるのを待つ時間を必要とするため，この指数関数的増加を放置しておいては，シミュレーションは計算時間的に不可能になる．そこで，系に望ましい上限人口密度 $P_a$ を導入し，個体数がそれを超えると死亡率が増えることで個体数の増加をおさえることができるようにした．

**表2** 系のパラメータ
Table 2 System parameters.

| | |
|---|---|
| 初期資産 $E_0$ | 10 |
| 対戦費 $C_m$ | 2.5 |
| エージェント移動確率 $p_{rw}$ | 0.025 |
| 格子サイズ $N$ | 15 |
| 突然変異確率 $p_m$ | 0.01 |
| NIL 位置置換確率 $p_r$ | 0.2 |
| 計算費 $C_{am}$ | 0.005 |
| アルゴリズム計算費 $C_{st}$ | 0.1 |
| 上限人口密度 $P_a$ | 2.0 |

対戦費 $C_m$ と死亡率導入の効果は相補的である．適当な死亡率を導入することで，対戦費を小さく（ゼロに）するモデルも可能である．我々の設定は，1) 人口が多すぎると（何らかの機構で）人口増大がおさえられる，2) 人口が少なくても非協調的な戦略だけでは繁殖できない，という2つの条件をモデルに要請したことになる．

我々が採用した系のパラメータをまとめて**表2**に示す．

## 5. 対戦戦略の社会的進化

4章で述べた条件で，TFT を出発点に対戦戦略の社会的進化を調べた．系は何回かの「地域的に限定された」絶滅を繰り返す中で，様々な戦略の発生，淘汰を引き起こし，結果として多様な対戦戦略群を生み出した．

このようにして産み出された戦略は，あるものは TFT よりも強く，また他のものは弱くなっているであろう．我々は，系全体としての進化を評価するために，次のような実験を行った．まず TFT から出発して，20,000時間後の戦略の分布を収集した．このときの系の人口変化（太線）と平均得点（小さな点）の様子を図**4**に示す．

次に，これと同じ戦略分布を持つ系（$S_0$ と呼ぶ）を作り，$S_0$ とその半数の DDD エージェントと対戦させて，その振舞いを調べた．また，比較対照のため $S_0$ と同数の TFT，TT3，REF からなる系 $S_{TFT}$，$S_{TT3}$，$S_{REF}$ を作り，やはりその半数の DDD との対戦をさせて，その振舞いを調べた．そのときの人口変化の様子を図**5**に示す．

以下では，もう少し定量的な議論をするために，「強さ指標」として，次のものを導入する．系は上限人口密度で決まる人口（$15\times15\times2=450$）の少し上の，550〜650を揺らいでいる．そこで，図5のような DDD との対戦実験において，最初に人口が500を切った点，再び500を回復した点，人口最下点の3点

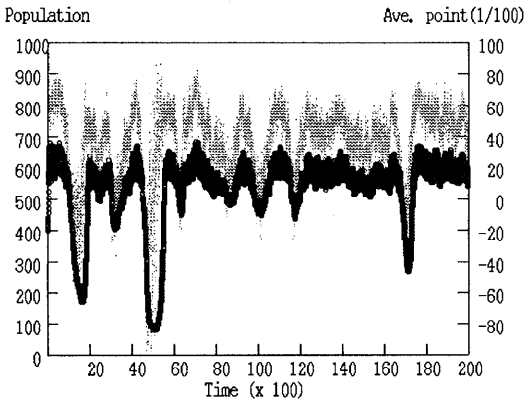



図4 対戦戦略の進化実験
Fig. 4 Evolution experiment of strategy algorithms.

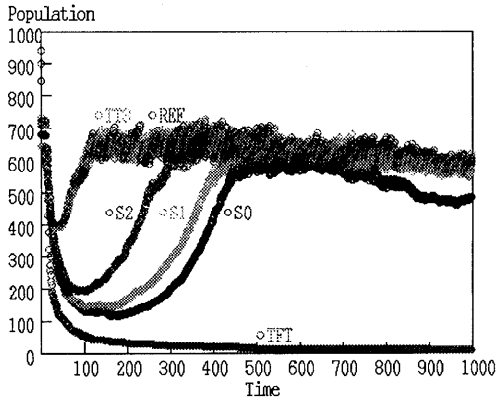



図5 様々な戦略のDDDとの対戦
Fig. 5 Matches of various strategies againt DDD.

からなる三角形の面積を求める．戦略が強いほどこの落ち込みが小さいと考えられ，我々はこの平方根の逆数を%で表したものを強さ指標として，戦略の評価を行う．なお，系が絶滅して再び500を回復できないときは，強さ指標は0としてある．また)100は，落ち込みが小さすぎるもの，始めから500を切ることのなかったものなどである．

TFTから20,000時間の進化実験は10回行われた．このうち1回は系は自然に絶滅した．残り9回のシミュレーションについて，20,000時間後のDDD対戦（$S_0$），さらにそこから立ち直った直後21,000でのDDD対戦（$S_1$）の平均値と，比較のため同条件でのTFT，TT3，REFの強さ指標を求めたのが**表3**である．実験結果をみると，残念ながら$S_0$は$S_{TT3}$には及びもつかない．しかしながら，$S_0$には多くの無抵抗な戦略が含まれているため，戦闘的戦略にとってきわめて不利な戦いとなる．それにもかかわらず$S_0$はTFTが絶滅してしまったのに比べて，かなり良い戦いをしていることが分かる．実際，1度目のDDDとの対戦の後では，そうした無抵抗な戦略が駆逐されるため，$S_1$は系としては少し強くなっている（図5，表3）．

表3 様々な系の強さ指標
Table 3 Robustness indices of robustness for various systems.

| $S_0$ | $S_1$ | $S_2$ | $S_{TFT}$ | $S_{TT3}$ | $S_{REF}$ |
|---|---|---|---|---|---|
| 0.28 | 0.37 | 0.70 | 0 | 2.04 | 1.99 |

では，このように系をつねに非協調戦略に晒しておけば，良い戦略が早く発生するのであろうか．このことを確かめるために，一定間隔で非協調戦略の「ワクチン」を投与することを行った．具体的には，TFTから出発して20,000時間後の系に対して，10,000時間にわたり，100時間ごとにその人口の1/5のDDDを投入して，弱い戦略の淘汰を促した．その結果得られた系を$S_2$と呼ぶ．$S_2$をその半数のDDDと対戦させた結果を図5，表3に併せて示した．

残念ながら，系全体としてはTFTからは大きく進化したものの，TT3に勝るものは生じさせることができなかった．この理由は，前にも述べたように，我々のシステムが平和期には非戦闘的な戦略をも許容するその寛容さにある．実際9個のシミュレーションで得られた$S_2$の総エージェント数5,371における戦略の種類は1,829に達し，生態系としてみたときの種の多様さが理解される．その中で上位10種の戦略をその全体に占める割合とともに**表4**に示す．1位にTFTが，7位にCCCが，10位にDDDが入ってきていることが社会の弱さを予想させる．しかしながら，それ以外の戦略には優秀なものがあり，それらの強さ指標を表3と同じ方法で求めてみた．ただしこれらのあるものは，半数のDDD対戦ではあまり差がつかないほど強さが接近しているので，さらに同数のRANとの対戦結果についての強さ指標も併せて求めてみた．

表から，3，4，8位の戦略などはTT3に比べても十分に強く，REFに匹敵する強さを持っていることが分かる．また，これらのアルゴリズムを見ると，すべて命令SLを含んでおり，何らかの形で再帰的処理を行っていることが分かる．これは，状況判断に自分ならどうするかという計算を行っているわけであり，つき合い戦略における「自分（の心）を用いて相手（の心）を読む」ことの重要性を示しているものと考えられる．

表4 $S_2$ 上位10種のアルゴリズムと強さ指標
Table 4 Algorithms codes and their robustness indices for top ten strategies.

| 順位 | アルゴリズム | 占有率（%） | DDD 対戦 | RAN 対戦 |
|---|---|---|---|---|
| 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 |

## 6. 考察とまとめ

囚人のジレンマゲームのエージェントの対戦としての定式化は，Rosenschein[9]らによって，また進化（世代の発展）の機構を用いた戦略の評価は，Lindgren[7]らによってなされている．我々の研究が，これまでの囚人のジレンマの研究と大きく異なるのは，エージェントが同一の相手と無限回（不特定回）対戦を行うのではなく，1回限りの対戦を次々と相手を変えながら行うという設定にある．もちろん，このような状況では，一般に裏切り（D）が最適戦略となってしまう．そこで，我々は裏切りに対する社会的制裁機構として，対戦履歴の公開を採り入れた．前回の論文[6]では，このような条件の下でも協調が可能であること，単純なしっぺ返し戦略は仲間同士でのしっぺ返しの応酬を招くため不十分であること，それを避けるためには，自分が相手の立場ならどのように振る舞うかと考える再帰的戦略が有効であることなどを示した．

今回の報告では，どのようにしてエージェントの協調が社会的に発現するのかを，遺伝的アルゴリズムの手法を借りて調べてみた．その結果，最初単純なしっぺ返し戦略 TFT から出発して，生存競争の中で系はより強い協調的戦略を生成しうることを示した．また，系は比較的長い平和期と突然の戦乱期とを交代しながら進化すること，平和期においては非戦闘的戦略が有利であるが，その割合が戦闘的戦略を上回ると非協調的戦略が非戦闘的戦略を搾取して成長し，戦乱期をもたらすことなどを示した．このような形で，生物種の様々な形質が進化することは，Trivers[10]にも述べられており，人の協調性の進化においてもそのようなプロセスの存在することを予想させる．我々のモデルが，どこまで実際の人間社会の協調機構と対応させうるものかは疑問も多いが，社会進化の機構により協調が発展しうることは示せたものと思われる．

本論文で得られた結論はシュミレーションに基づいているため，一般性についての議論が必要である．本論文の主張を一言でいえば，「情報公開下の囚人のジレンマゲームにおいて，適切な外部環境を用意すれば，社会的進化の機構により協調的戦略が優勢となりうる」というものである．したがって，ゲームの利得行列や，また4章で示した様々なパラメータを変えることで，協調が発現しなくなることは十分起こりうるし，実験でもこのことは確かめられているところである．

しかしながら，戦略を3章の計算器で記述できるものに限定したことは，次の疑問に結び付く．「選択可能な戦略の範囲を広げることで，協調が困難になることはないだろうか？」もしそのようなことが示されれば，我々の結論は「誤っていた」といわざるをえないだろう．たとえば，我々の実験で協調的社会を担っていた戦略を駆逐できるような十分に「強い」戦略を考案することはおそらく可能であろう．たとえば，「相手の過去の行動パターンを利用する」，「現在の自己の資産レベルに応じて戦略を変える」などの能力は，十分小さい計算コストの下ではそれなりに威力を発揮するものと思われる．我々が今回の実験で簡単な計算器と比較的大きな計算コストを設定した理由は，主としてシミュレーションを現実的な計算時間で実行するためであった．したがって，高速の CPU，大容量メモリ，効率良い実装（現在は LISP）により，より強い戦略の可能性を調べてみるのは興味深い挑戦である．

しかしながら，より「強い」戦略があること自体は協調の発現を否定する根拠にはならない．物理的条件（計算器の計算能力，計算コストなど）が同じであれば，強いアルゴリズムを発見したエージェントは一時的にはそれによって利益を得られたとしても，いずれは皆がその能力を獲得することで，それだけでは高得

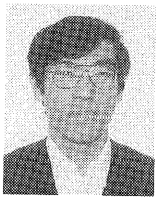
点を稼げなくなる．そのときには，新しく開発された強いアルゴリズムを含みつつ，相手が自分と同じ計算能力を持つと仮定して，自ら協調的行動をとろうとする再帰戦略が再び優勢になると，著者らは期待しているわけである．もちろん，これについての本当のところは今後の研究に待たねばならない．

このような改良を含めても，我々がここで扱った対戦戦略アルゴリズムは，人間が他人との付き合いにおいて用いているであろう戦略と比較すれば，「原生動物」的レベルであることは否定できない．エージェント同士がよりうまく協調を行うためには，より高度な相手のモデルを作ること，エージェント同士のコミュニケーションを考えることなどが必要になる．コミュニケーションの自発的進化に関しては，Werner[11]らがmate findingの問題において，最初は理解できなかったfemaleの信号をmaleが学習・利用できるようになることを示しており，我々のモデルにおいてもコミュニケーションがいかに進化しうるかはぜひとも取り組みたい課題である．

## 参考文献


1) Axelrod, R. and Hamilton, W.D.: The Evolution of Cooperation, *Science*, Vol.211, No.3, pp.1390–1396 (1981).
2) Axelrod, R.: *The Evolution of Cooperation*, Basic Books (1984). 松田裕之（訳）：つきあい方の科学，HBJ出版局 (1987).
3) Axelrod, R. and Dion, D.: The Further Evolution of Cooperation, *Science*, Vol.242, No.12, pp.1385–1390 (1988).
4) Genesereth, M.R., Ginsberg, M.L. and Rosenschein, J.S.: Cooperation Without Communication, *Proc. IAAA-86*, IAAA (1986).
5) 大沢英一，沼岡千里，石田　亨：サーベイ：分散人工知能小問題集，中島秀之（編），マルチエージェントと協調計算I，近代科学社 (1991).
6) 伊藤　昭，矢野博之：取引履歴公開下での最適取引戦略，人工知能学会誌，Vol.10, No.2, pp.271–278 (1995).
7) Lindgren, L.: Evolutionary Phenomena in Simple Dynamics, *Artificial Life II*, Langton, C.G., et al. (Eds.), pp.295–312, Addison-Wesley (1991).
8) Ray, T.S.: An Approach to the Synthesis of Life, *Artificial Life II*, Langton, C.G., et al. (Eds.), Addison-Wesley (1991).
9) Rosenshein, J.S. and Genesereth, M.R.: Deals Among Rational Agents, *Proc. IJCAI'85*, pp.91–99 (1985).
10) Trivers, R.: *Social Evolution*, Benjamin/Cummings (1985). 中嶋泰裕ほか（訳）：生物の社会進化，産業図書 (1991).
11) Werner, G.M. and G. Dyer, M.G.: Evolution of Communication in Artificial Organisms, *Artificial Life II*, Langton, C.G., et al. (Eds.), Addison-Wesley (1991).


(平成8年10月11日受付)
(平成9年3月7日採録)



**伊藤　昭**（正会員）

1972年京都大学理学部物理学科卒業．1979年同大大学院理学研究科博士卒業（理学博士）．同年郵政省電波研究所（現通信総合研究所）入所．以後，通信ネットワーク，知識処理，対話システム，ヒューマン・インタフェース，エージェントモデルなどの研究に従事．現在，同所関西支所研究調整官・知識処理研究室長．情報処理学会，電子情報通信学会，人工知能学会，日本認知科学会各会員．

**矢野　博之**（正会員）

1986年東北大学工学部通信工学科卒業．1992年同大大学院工学研究科博士後期課程単位取得退学．同年，郵政省通信総合研究所入所．現在，関西先端研究センター所属．工学博士．協調処理，対話，コミュニケーションモデル，エージェントモデルなどに興味を持つ．情報処理学会，ソフトウェア科学会，認知科学会各会員．