From genetic evolution to emergence of game strategies

Takashi Ikegami

The Graduate School of Science and Technology, Kobe university, Kobe 657, Japan

Abstract

Evolution of game strategies is studied in the Erroneous Iterated Prisoner's Dilemma game, in which a player sometimes decides on an erroneous action contrary to his own strategy. Erroneous games of this kind have been known to lead to the evolution of a variety of strategies. This paper describes genetic fusion modeling as a particular source of new strategies. Successive actions are chosen according to strategies having finite memory capacity, and strategy algorithms are elaborated by genetic fusion. Such a fusion process introduces a rhizome structure in a genealogy tree. Emergence of module strategies functions as innovative source of new strategies. How the extinction of strategies and module evolution leads to ESS-free open-ended evolution is also discussed.

1. Introduction

Evolution is a definitive characteristic of life. It is a process that is not just the pattern-changing exhibited in cloud or snowflake formation; the evolutionary process demonstrates adaptive improvements and innovations. Examples of the innovations stemming from natural evolution range from the intricate suture lines of ammonite shells and the expression of armor in gastropods [1] to the evolution of the human brain and immune system.

What sort of evolutionary dynamics make innovation possible? Is the mechanism which was at work in ammonites also at work in the evolution of the human nervous system? These questions cannot be answered experimentally as causation in natural evolution takes too much time.

Fossil studies provide the best evidence we have of natural evolution.

Research with artificial life, therefore, enables one effective approach to the study of evolutionary dynamics. Artificial evolution proceeds in computers, wherein evolutionary mechanisms are studied in a purely idealized fashion, free from hardware constraints.

To be able to describe a system not from without, but from within the system itself, characterizes research with artificial life. If a system is in an equilibrium state, the effective degrees of freedom of the system can be determined from without the system, but for evolving systems, in which there are hidden interior degrees of freedom affecting its future evolution, description of the system from without inevitably fails. Hence we must put our descriptive viewpoint inside the system. This means that we try to understand the system not as much by describing it as by synthesizing it.
The synthetic approach is exemplified in describing the evolution of complexity in protein folding. The complex structure of a protein in the folded state is often described as having a fitness landscape of spin-glass like ruggedness, wherein lower energy states correspond to higher fitness of protein function [2,3].

Described thus, an individual protein, in realizing optimal structure, would have to determine this complex landscape, the computation of which turns out to be a difficult problem. According to the synthetic approach, however, we may suppose that an initially small protein of simple structure evolves to complexity by a process of combining with other proteins. This process, which the author has elsewhere termed a process of genetic fusion [4], will lead to the appearance of module-type protein structures [5]. Many proteins then use these modular protein structures as a "subroutine program", reducing individual computational time in determining fitness within the complex landscape. Even wherein the final landscape is highly complex, the individual protein does not itself have to determine the complexity of the entire landscape. This scheme may be well phrased as: complex systems are more easily synthesized than analyzed.

What then is the key mechanism that brings about synthesis in artificial life? Several mechanisms have been proposed based upon studies in real biology. In this paper, we will stress the importance of symbiosis and game-theoretical interactions as aspects of synthetic processes in artificial life evolutionary systems. Symbiotic relationships are relationships of mutual cooperation occurring at the genetic level as well as among individual organisms. Symbiosis is now being more broadly considered as a source of innovation in many evolving phenomena. The evolution of eukaryotic cells from associations of prokaryotes is a well-known symbiotic dynamics [6]. Another example is the incorporation by bacteria of a variety of plasmids, which leads to innovation (e.g., bacteria evolving to an antibiotic-resistant strain through the incorporation of R plasmids).

In artificial genetic modeling, symbiotic evolution is described by genetic fusion [4]. Genes are represented by binary strings, and by the process of genetic fusion, two genes are combined to generate a longer gene. It has been found that genetic fusion therein leads to the appearance of module-type genes, and that the time necessary to arrive at a genotype having higher fitness is greatly reduced. The example given earlier of modeling protein evolution is one application of this characteristic of genetic fusion. The other important aspect of synthesis in artificial life evolutionary systems is game-theoretical interactions. Exchanging information, telling lies, imitating others and developing strategies, for example, only make sense from the viewpoint of game theory. Symbiotic relationships will also involve these sorts of game-theoretical interactions. Although game theory is developed so as to describe a system from without, we are interested in the use of game theory as a synthetic method. Specifically, we will discuss the synthesis of game strategies. Introducing population dynamics and mutations including genetic fusion, we will examine an ensemble of game-playing species. In particular, the Iterated Prisoner's Dilemma game will be studied as the simplest strategic game. We will thus see coevolution of strategies involving symbiosis to be a source for synthesizing new strategies.

Outline of the paper

The present paper is organized as follows. The rules of the Iterated Prisoner's Dilemma game are briefly introduced and the ESS (Evolutionary stable strategy) and EMSS (Evolutional metastable strategy) concepts are explained in Section 2. A modification to Axelrod's original version of the Iterated Prisoner's Dilemma game is made, allowing a player to make an erroneous action, i.e. opposite to the strategy.
Table 1
The payoff matrix for the Prisoner's Dilemma game. Each
matrix element, \((S_1, S_2)\), corresponds to the score of player
1 and 2, respectively. Following Axelrod's tournament \([7]\),
we use \(R = 3, S = 0, T = 5\) and \(P = 1\).

In Section 3, the strategy is represented and
various genetic operators are introduced. Five
basic operators are used: flipping, cutting, re-
versing, adding sites on a selected gene and
genetic fusion on pair trees. In Section 4, time
evolution of population is discussed. Unstable
oscillative dynamics are frequently observed in
such systems. The complication of strategies
is analyzed by defining a complexity measure,
which is introduced in Section 5. An evolu-
tionary pathway is composed of modules and
their associations. A rhizome-like organization
of the genealogy of strategies is attributed to the
formation of module strategies. These facts are
discussed in Section 6. Section 7 is devoted to
discussions of my results with respect to module
evolution, ESS-free evolution and extinction.

2. Iterated prisoner's dilemma game

Axelrod has conducted a computer tourna-
ment of the Iterated Prisoner's Dilemma (IPD)
game with the following payoff matrix (Table
1). During each match, a player chooses an
action ‘Cooperate’ (C) or ‘Defect’ (D) without
knowing the opponent’s next action. These pay-
off values satisfy the relations \(T > R > P > S\)
and \(R > (T + S)/2\), so that playing D domi-
nates playing C if the game is played only once.
When the game is iterated there may be a great
variety of strategies.

When more than one match is played, always
playing D or C is no longer a good strategy. In
Axelrod's tournament, 15 different strategies are
employed in the first tournament and 63 in the
second. The winning strategy in the two tourna-
ment was the same, the simplest strategy, called
Tit for Tat (TFT) \([7]\). This strategy starts with
cooperation and immediately defects in response
to defection but immediately cooperates in re-
sponse to cooperation. Although it may look sim-
ple, TFT manages to minimize damage. It de-
velops mutual cooperative behavior, and gets high
scores.

In addition to the original Axelrod rules, we
assume that a player sometimes makes a mistake
and plays opposite to the strategy. It has been
shown that in such an erroneous game, TFT is
replaced by strategies which are more tolerant
of defections \([8-10]\). One such example may be
a series of Tit for n-Tat. It only defects against
\(n\) successive defections. This generosity helps to
keep the simple TFT from going into a period 2
state of alternatively playing C and D, which de-
creases the average score of TFT from 3 to 2.5.
However, it can then be exploited by sophisti-
cated defectors. In a low error regime, where it
is difficult to distinguish between intended and
unintended defections, unconditional generosity
results in a high score. In a high error regime,
however, there is trade-off between generosity
and exploitation.

A large set of strategies can be programmed as
memory strategies. A memory strategy computes
one's next move from a record of the past moves
of both players. TFT is a memory strategy with
a memory length of one move of its own. Thus
a general question arises: Does a long memory
strategy give an advantage?

To answer this question, many computer sim-
ulations have been performed. These simula-
tions introduce an ensemble of species playing
the IPD game. Successful species increase in
population and the ensemble is updated with
new species through mutations. An ensemble
of a game playing species with population dy-
namics defines the evolutionary game (see, e.g.
\([11-16]\)).

Axelrod has done some computer simulations
showing that in the error environment of the
second round of his tournament, there exists
a strategy with memory length three moves of both players that is better than TFT. K. Lindgren has shown that strategies with minimal memory length of at least two moves of both players may be needed for the strategy to be successful in the erroneous IPD game [16]. Also, if the strategy is limited by having a maximum memory length of one move for both players, a strategy known as Pavlov appears to be the best strategy [17].

The author has reported on strategies which have a chance to evolve far longer memories if they consist of a specialized ecology without ESS [18]. This is because, in order to get high scores, in the ecology each has to coevolve with specific strategies, which results in an arm's race. Ecology which has otherwise evolved ESS strategies with memory length for two moves of both players, belongs to "Lindgren's class" [2].

In this paper, I'll look at mechanisms which have longer memory strategies through module-type evolution.

3. Modeling

3.1. Tree representation

A memory strategy computes the next action based on the previous sequence of moves, its own and the opponent's. Each strategy is characterized by a set of sequences of moves, cooperation (C) and defection (D), against which the strategy is to cooperate (or defect).

We call each sequence of a strategy a gene. These genes are put together to form a tree structure like a genome. If patterns of more than two genes overlap, the longest gene supersedes the shorter ones (see Fig. 1a).

---

2 In the article [16], he shows that ESS class strategies fix 7 bits out of 16 bits for a memory 4 strategy. That is, ESS strategies must play cooperation for the sequences, 0000,1100 and 1111, and defection for those 0010, 0001, 1011 and 0111. A strategy A in [18] happens to belong to this class. A structure of strategy A will be discussed in Section 6.
tree structure, nodes in the even levels correspond to the outcome of the player's own moves and those of odd levels to that of the opponent's moves.

When playing the game, a sequence of past moves generated, denoted by a, is matched against all the genes from the tree (denoted by \{\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n\}). None of the genes is comprised in the other. A match occurs if each symbol in each position matches. If either sequence, a or \alpha_j, is shorter than the other, a match occurs over the shorter part of the sequences. A C-tree will play C and a D-tree will play D for the matched sequence.

Let us examine the present representations with simple examples. For Pavlov, the set of move sequences resulting in a C-tree representation consist of 00 and 11, while the D-tree representation consists of 01 and 10 (Fig. 1c).

The other strategies, which will be named 92 and 167, represent the same algorithm that cooperates for a 11 move sequence. They have different D-tree representations consisting of 00, 01, and 0 and 10 (strategy 167) (Fig. 1c). Namely, gene 0 of strategy 167 is equivalent to two genes, 00 and 01.

3.2. Genetic operations

We take the following process for the evolutionary dynamics: Each strategy is represented by a set of genes \(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n\), where each gene consists of a binary string \(\alpha = \sigma_1, \sigma_2, \ldots, \sigma_n\), where \(\sigma = 0\) or 1. Each of these strings represent a sequence of moves that implies that the strategy should cooperate.

(i) Mutation processes. We admit four types of mutations. Here we assume that at most only a single mutation occurs for each gene.

(a) Flipping the state of a randomly selected site in a gene at the rate \(\mu\):
\[ [\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5] \rightarrow [\sigma_1, \sigma_2, 1 - \sigma_3, \sigma_4, \sigma_5]. \]

(b) Adding one new gene to the rightmost site of a randomly selected gene at the rate \(\alpha\):
\[ [\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5] \rightarrow [\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6]. \]

(c) Duplicating one randomly selected gene, and against one duplicated gene reverse all states from a randomly selected site to the rightmost site at the rate \(\text{rev}\):
\[ [\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5] \rightarrow [\sigma_1, \sigma_2, 1 - \sigma_3, 1 - \sigma_4, 1 - \sigma_5]. \]

(d) Cutting a gene at a randomly selected site at the rate \(\text{cut}\):
\[ [\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5] \rightarrow [\sigma_1, \sigma_2, \sigma_3]. \]

A tree type and an initial move of a strategy are also mutated during the processes (a) and (b) at the mutation rates \(\mu\) and \(\alpha\), respectively. The processes (a) and (b) deal with one bit mutation, whereas the processes (c) and (d) involve simultaneous multiple bit mutations. The process (c) adds one gene and the process (d) often eliminates one gene from a strategy. The process (c) may look rather unusual, but it is found that this process is effective for generating new strategies.

(ii) Genetic fusion. Two strategies \(T_1\) and \(T_2\) interact to create a new strategy \(T_3\) as follows (Fig. 2): One gene \(\alpha_k\) is selected from the tree \(T_2\) and new genes \([\alpha_k \tau_1, \alpha_k \tau_2, \ldots, \alpha_k \tau_n]\) are created, where the symbols \(\tau\) denote genes in the tree \(T_1\). A generated tree has genes \([\alpha_1, \alpha_2, \ldots, \alpha_{k-1}, \alpha_k \tau_1, \alpha_k \tau_2, \ldots, \alpha_k \tau_n, \alpha_{k+1}, \ldots, \alpha_m]\). If the original trees have \(\ell_1\) and \(\ell_2\) genes, the generated tree has \(\ell_1 + \ell_2 - 1\) genes. A generated strategy \(T_3\) inherits to an initial move and a tree type of strategy \(T_2\).

We assume that fusion occurs at most in one randomly selected gene from a tree. A fusion partner is randomly chosen from an ensemble.
The rate of fusion occurrence is represented by $\zeta$.

The most distinctive difference between fusion and other mutations is that mutations are caused by a single species, whereas fusion involves two species. As a result, mutation processes provide a genealogy with a tree-like shape, whereas fusion among different strategies can provide a genealogy which is organized in a rhizome-like structure. Self-duplication is a special case of the fusion process, where the genealogy remains organized in a tree-like structure.

3.3. Population dynamics

Introducing an ensemble, we denote the size of the population of strategy $k$ by $N(k)$. The total population size is normalized to unity. The following three processes constitute the population dynamics.

(a) Reproduction process. Each strategy species generates offspring in proportion to its fitness. We assume that the fitness depends on the score $S$ as $\exp(\beta S)$. To normalize the total population size to unity, $N'(i) = N(i) \times \exp(\beta S(i)) / \sum_j N(j) \times \exp(\beta S(j))$. A score for the strategy $i$ against the strategy $j$ is denoted by $S_{i,j}$, which is computed by averaging over $\tau$ times over $R$ rounds. Namely, $S_{i,j} = \langle R^{-1} \sum_{k=1}^{R} s^k_{i,j} \rangle_\tau + \delta_{i,j}$ and $S(i) = \sum_j S_{i,j} N(j) / \sum_k N(k)$, where $s^k_{i,j}$ gives the score of the $k$th round. One match consists of $R$ actions for each player, giving $R$ rounds. The averaged score is further averaged over $\tau$ times, which is denoted by $\langle \bullet \rangle_\tau$. By taking $\tau$ to be large, $\delta$ will become negligible. Since we take $\tau$ to be 10, $\delta$ is in the order of $10^{-2}$. The quantities $R$ and $\tau$ introduce fluctuations in simulations.

(b) Mutation process. By the genetic operations introduced in the previous section, a proportion of the reproduced strategies should mutate. The population size of mutants from strategy $i$ is given by $\gamma \times N'(i)$ for the mutation process. Also, the population size of mutants generated by the fusion process from strategies $i$ and $j$ is given by $\lambda \times N'(i) N'(j)$.

(c) Death process. Each strategy species is removed from an ensemble if its population size goes below a threshold, which is denoted by $\epsilon$.

Our population dynamics consists of successive operations of the above processes. Starting from an ensemble of species with small trees, we first let them play a game, then apply the reproduction process. Then mutations and genetic fusion are applied to the ensemble. Lastly, we remove any rare strategies. This completes the first time step. We continue these procedures over many time steps.

4. Time evolution of population

We have mainly concentrated on evolutions from initial strategies with memory length of one move of one's opponent. The initial strategies consist of TFT, ALL-D, ALL-C and anti-TFT (whose action is opposite to TFT) with either C or D as its initial action.

Dynamical behaviors of short time era are heuristically classified into either fixed-point dynamics or oscillatory dynamics. Fixed-point dynamics consists of one dominant strategy with infrequent mutants which are generated from the strategy. On the other hand, oscillatory dynamics consists of competing strategies, having no dominant strategy. An entire evolutionary course can be described by alternative switching of these dynamics.

Fixed-point dynamics is related to the idea of evolutionary stable strategy (ESS) introduced by J. Maynard Smith [19]. ESS defines a strategy which is stable against invasion by any possible finite mixture of mutant strategies. It has been shown by Boyd and Lorberbaum that no pure strategy becomes ESS in the errorless IPD game [20]. Furthermore, for the erroneous IPD game,
ESS has been shown to exist [21] and indeed strategies which belong to Lindgren’s class become ESS [16].

Since a mutation process allows a restricted set of mutants to evolve, it is difficult to test the ESS condition in simulations. We define an EMSS (evolutionary metastable strategy) as a strategy that is stable against invasion by all of its one-step mutants, including mutants by the self-fusioning process. Even if the dominant strategy is not EMSS, it is soon taken over by its mutants. The number of EMSS’s is determined by the properties of genetic operators and by a death process which removes infrequent strategies. If we only apply point-mutation and genetic fusion as in my previous paper [18], the number of EMSS’s with fixed point dynamics increases. In other words, a system will become trapped by an EMSS. Also, a large value for the removal threshold increases the number of fixed-points.

In the present simulation, we set the threshold $\epsilon$ to 0.00001, which is small enough to allow many strategies to coexist. The system now has more oscillatory dynamics than fixed-point dynamics.

By turning off all the genetic operations and setting the removal threshold to zero, we examine whether those oscillations are essentially chaotic or periodic (Fig. 3). Some of the oscillations turned out to be periodic and some turned out to be decaying oscillations. No persistent chaotic motion has been discovered in our simulations, but it would not have been surprising if we could have observed chaotic motions [23]. The decreasing of fixed-point dynamics and the increasing of oscillatory behavior should be explainable by May’s complexity and stability hypothesis [22]. Since the present game assumes a round-robin, the connectivity of interaction increases in proportion to the square of the number of strategies. This may break a condition of stability in an EMSS, generating unstable oscillatory behaviors.

Whether a system will finally go to an EMSS state or whether it will wander forever is difficult to tell in advance. If a system is wandering forever, we call it open-ended or ESS-free evolution. A large threshold value in the death process and variety of genetic operations allow more ESS-free evolutions.

The whole evolutionary course exhibits rapidly changing eras and quasi-static phases. This presents another example of punctuated
equilibrium phenomena [24]. In the transition eras, extinction of strategies are often observed. There occur many small-scale extinctions and a few large-scale extinctions. The mechanism of the extinction is attributed to the emergence of modules.

5. Evolution of complexity

5.1. Definition of complexity

By treating a strategy as a program, we hypothesise a mechanism that brings escalation of algorithmic complexity. Each node of a tree corresponds to a branching point of the decision-making procedure, where the kth level of a tree corresponds to the outcome of the preceding \( k - 1 \) steps. Branching out to the right (left) subtree is equivalent to the occurrence of a C(D) move. When playing the game, a sequence of past moves generated, denote by \( a \), is matched against all the genes from the tree (see Section 3.1). That is, we recursively execute the following program \( a \) times.

\[
\text{subtree}(i)\{ \\
\text{If a state of the ith site of } a \text{ is C } \\
\quad \text{seek a right subtree}(i+1) \\
\text{else} \\
\quad \text{seek a left subtree}(i+1) \\
\},
\]

where \( i \) denotes the level of a tree (see Fig. 4).

If a tree is a C(D)-tree and we hit a leaf node, we will play C(D). If a tree is a C(D)-tree and we branch out to a null subtree, we will play the opposite. If the sequence \( a \) is short and the execution is terminated within a tree, we will play C(D) if the tree is C(D). Here a leaf is defined as a node at the lowest level.

We introduce a quantitative measure of complexity by the computation of tree shape diversity. If a node has two isomorphic subtrees at the next level, branching out to the left or right subtree is irrelevant for the procedure. Such a node corresponds to a “don’t care symbol” (#) in the classifier system [25]. On the other hand, if a node has non-isomorphic subtrees, it is relevant for the decision procedure.

Whether two trees are isomorphic or not is determined by examining all the levels of the trees. We therefore define the total diversity of a tree as the total number of non-isomorphic branchings in the tree.

This measure for tree diversity was originally proposed by Huberman and Hogg [26,27] for general hierarchical organizations with constant depth. We have applied it to a strategy tree with non-constant depth. Since diversity here is a measure of algorithmic complexity, the same algorithm should give the same diversity. For example, a fully branched tree is algorithmically equivalent to a single root, showing zero complexity. A basic reduction rule is to note that a node with two leaves is equivalent to a node without leaves. After reducing these irrelevant branchings, the diversity of tree \( T_n \) is computed as follows 3. Here the tree consists of two subtrees \( T_{n+1}^1 \) and \( T_{n+1}^2 \), which start at the next level (\( n + 1 \)):

\[
\text{Div}(T_n) = \left( T_n \right) \prod_{i=1}^{b} \text{Div}(T_{n+1}^i), \quad (1)
\]

3 The present definition of diversity is a slightly different from the previous one [18]. In the previous definition we summed the diversity of subtrees, here we have multiplied them. The present definition is now more directly related to the total number of nodes with non-isomorphic subtrees.
where $i$ ranges from 1 to $b$, the number of non-isomorphic subtrees. If the node is a leaf, $b$ is 0 (e.g. $D(T) = 1$). If a node has one subtree, $b$ is 1. The maximum value of $b$ is 2. The number of different combinations of the $b$ subtrees in a tree is represented by $f(T)$.

Following Huberman’s definition, we take $f(T) = 1$ for two isomorphic subtrees and $f(T) = 3$ for two nonisomorphic subtrees. If there exists only one subtree, $f(T)$ is 2.

A program of a C-tree calls C against the sequence of moves found in the tree, but that of a D-tree does the opposite. Hence we have to transform the Diversity of the C-tree to that of the D-tree and vice versa. If we have only one subtree of non-zero complexity at some node in the C-tree, it corresponds to the node with one leaf and that subtree in the D-tree. Therefore, we transform $f(T)$ from 2 to 3 or vice versa when converting tree types. In the following figures of complexities, we measure complexity of D-trees by converting C-trees to D-trees.

We now define the complexity of a tree, $\text{Com}(T_n)$, as

$$\text{Com}(T_n) = \log_2(\text{Div}(T_n))$$

(2)

and

$$= \sum_{i=1}^{b} \text{Com}(T_{n+1}) + \log_2(f(T_{n}))$$

(3)

Complexity values for randomly generated strategies are generally large for long memory lengths. On the other hand, strategies such as All-C or All-D have complexities of zero, the large memories of these strategies are redundant for decision procedures. They do not contribute to the total complexity, but could be important for future evolution, since genetic operators will work on them.

5.2. *Time evolution of complexity*

We now follow evolution of strategies as evolution of complexity. Long-term behavior of averaged complexity and score are depicted in Figs. 5 and 6. These two examples have the same parameter values except the error rate. The complexity value does not always increase when the score increases. However, we may say that higher complexity strategies evolve in time as a long-term trend.

The stepwise changes of complexity values correspond to extinction of strategies and populations. The population rapidly falls whenever complexity value increases or decreases. This suggests that strategies which possess similar complexity will be extinguished simultaneously.

In order to look closely at the extinction phenomena, genealogy trees of strategies have been drawn in Fig. 7 using complexity values. Figs. 7a and 7b correspond to Figs. 5 and 6, respectively. In the figures of genealogy tree, strategies are marked at their birth time step. A strategy which is generated by genetic fusion has two incoming lines, suggesting two ascendants. Each strategy is named by the decimal number which is sequentially attached to when they have appeared.

Strategies are grouped into several families with respect to their complexity values. There are two groups around 7000 time steps in Fig. 7a and approximately three or more around 8000 time steps in Fig. 7b. The era of coexistence of distinct families precedes a large-scale extinction of strategies and populations. If strategies in one family share a common structure, they will show similar action patterns. At the same time, they become weak against a common enemy. The emergence of the common enemy may therefore trigger the extinction of the family. If the family contains a large population, this leads to large-scale extinction. One family that escapes this extinction will generate a new era of that family and its descendants.

Following the large-scale extinction, the ecology of Fig. 7a is taken over by a lower-complexity family, evolving into Lindgren’s ESS class after a few thousand time steps. A maximum memory length of 4 moves is attained by the strategy at 10000 time steps. Higher complexity strategies with maximum memory length of 8 moves have emerged at around 6000 time steps, but
Fig. 5. Time evolution of averaged score (solid line) and the number of distinct strategies (dashed line) in the upper figure and averaged complexity in the lower. A large-scale proliferation and extinction is observed around 7000. The evolving strategies are mostly D-trees. The parameters of genetic operations are mut = 0.0002, add = 0.0002, cut = 0.005, rev = 0.005 and ζ = 0.0008. Those for population dynamics are β = 1.2 and the error quantity is set at 0.07.
Fig. 6. Time evolution of averaged score (solid line) and the number of distinct strategies (dashed line) in the upper figure and averaged complexity in the lower. A large-scale proliferation and extinction is observed at the time step around 8000. Many small-scale extinctions are observed in this figure. The evolving strategies are mostly D-trees. The same initial strategies and the same parameter values as Fig. 5 are used except for the error rate, which is set at 0.05.
are extinguished at the measure extinction. On the other hand, the highest complexity family succeeds in Fig. 7b. Through small-scale extinctions, it attains a highly module-organized strategy with a maximum memory of 5 of one's own moves and 6 of one's opponent's.

5.3. Random tree

It is worthwhile comparing survived strategies with randomly generated strategies. A sample consists of 500 randomly generated trees of maximum height 5 (i.e., 2 of one's own moves and 3 of one's opponent's) with average complexity around 20. To generate a set of random strategies, we first generate 32 random binary sequences (i.e., genes), each of which can have a different bit length from one to five. Then we construct a tree from the set, as illustrated in Fig. 8a. The statistics of the set of trees include an average leaf number of 10.4. The distribution has the maximum number of states at the mid value of complexity (Fig. 8a). This implies that the maximum complexity tree is different from the maximum entropy tree, in agreement with what our intuition suggests. Even trees with
the same number of leaves have different values of complexity (Fig. 8b). The average values of complexity are generally large for long memory length.

The author has conducted a round-robin IPD game among these fixed 500 randomly generated strategies. No ESS strategy has been found among the 500 random strategies. A strategy \( i \) is called ESS if and only if the following two conditions are satisfied: \( S_{i,i} > S_{j,i} \) for all \( j \); and if the relation \( S_{i,i} = S_{j,i} \) holds, the inequality \( S_{i,j} > S_{j,j} \) should be satisfied. A score to the strategy \( i \) against the strategy \( j \) is denoted by \( S_{i,j} \), which is computed by averaging over 100 times over 100 rounds.

Most of the strategies do better against half of the strategies and worse against the other half. Since the strategy from Fig. 7a at time step 10000 belongs to Lindgren’s class, it satisfies the ESS condition against the 500 random strategies. The All-D strategy also satisfies the ESS condition against the 500 random strategies. However, the strategy 10259 at time step 9500, which is a descendant of a survival of the large-scale extinction (Fig. 7b) does not satisfy the ESS condition. Thirty-eight out of the 500 random strategies get higher scores than the strategy 10259.

To escalate the complexity of a strategy, the ESS condition should be avoided, since ESS is attainable with low complexity in the IPD game. We shall look more carefully into the origin and history of the strategies in Figs. 5 and 6 in the next section.
Fig. 8. A distribution of complexity of a randomly generated tree with a maximum memory length 5 (upper figure). The sample consists of 500 random trees with an average leaf number of 10.4. Complexity value distributes even for the trees with the same number (I leaves) of leaves (lower figure). This reflects the fact that the defined complexity is not a simple function of the number of nodes or leaves.

6. Module

The fusion process introduces a rhizome structure in a genealogy tree. At the same time, fusion-generated strategies have nested structures of module pieces.

From observations of the present simulations, we have three different fusion classes:

(1) Homogenous fusion. Fusion of two identical strategies. Gene duplication belongs to this class. This process is frequently observed when one strategy becomes dominant and long lived.

(2) Heterogenous fusion. Fusion caused by two different strategies. If dynamics become unstable and several strategies compete in an ecology, heterogeneous fusions frequently occur. In the other case, when the products of fusion allow further fusion to occur, many mutants produced by heterogeneous fusion appear. Instead of making innovations, the latter case shows an entropy consuming process.

(3) Fusion with zero complexity trees. Fusion caused by zero complexity strategies. Since fusion with zero complexity trees is a redundant process, it doesn’t change the strategy. Only All-C or All-D strategies have zero complexity and cause this type of fusion.

Since a tree has different meanings on each level, a subtree has different context depending on its location. Even duplication of a strategy changes the strategy, except in the case of the third fusion class.

It has been found that fusion processes lead to the emergence of module strategies. A module strategy is defined as a strategy that is used by many other strategies. For a strategy to become a module strategy, it has to live long enough to create new strategies through fusions. Hence EMSS’s have more chance to become module strategies.

In my previous work [18], a strategy (denoted by strategy A in the article )which belongs to Lindgren’s ESS class evolved through modules. It is attained through duplication of the strategy called Pavlov by Nowak (Fig. 9a). Strategy A is an ESS in a C-tree with memory length of 4 moves and Pavlov is an EMSS within a maximum memory length of 2 moves. However, the present simulation has evolved Lindgren’s class not through duplication but through heterogeneous fusion with mutations. Pavlov in a D-tree (strategy 129 in Fig. 9b) cannot generate a strategy A in a C-tree through duplication.

It has been found in the simulation that complex strategies after a long period of evolution are not ESS’s in a test examining them against the random trees, but consist of modules having long memory. Non-ESS strategies are in general made up of modules of modules of modules, showing far more hierarchical and nested orga-
Fig. 9. Examples of module evolutions. (a) Duplication of Pavlov in a C-tree generates Lindgren’s class strategy (see footnote 2). (b) A tree shape of module strategies in the genealogy of TFT descendants. The numbers shown indicate when each strategy appeared. Dashed lines denote the sequence of mutation processes, where m,a,r,c denote the mutation processes (a),(b),(c) and (d), respectively. A letter F with solid line represents a fusion process. The early stage population is dominated by TFT, which duplicates to generate a module strategy 92. This strategy 92 generates another module strategy 215, which also generates another module strategy 353. This evolutionary pathway leads to the complex but non-ESS strategies (e.g. 10259 in this experiment).
nization. Strategy 10259, for example, contains 18 modules of strategy 92 and its mutants (see Fig. 9b). At the same time it includes 3 pieces of strategy 353 or 6 pieces of strategy 215. The strategies 92 and 129 coexist for more than 1000 steps and the strategies 215 and 353 appear in order through fusion processes.

Since these are D-trees, generosity increases through fusion processes. A strategy 92 defects against all the sequences that begin with CD. The descendant strategy 215 defects against the same sequences except for the sequences that begin with CDCC. The next descendant strategy 353 cooperates against the sequences that begin with CDCC, CDCDC, CDCDDCC, CDCDDC and CDCDCDCC, i.e., the sequences beginning with CD and CDCD. However, these strategies always defect for the sequences beginning with D.

Specifically, generosity in those strategies is to forgive an opponent when he has repeatedly cooperated against one’s defections. The degree of this generosity is enhanced by the evolution.

It is worth noting that strategy 129 is Pavlov and strategy 92 is the predecessor of Pavlov. These examples suggest that Pavlov, and the strategies 92, 215 and 353, may function as module strategies. Evolutionary processes beginning with TFT may either go to ESS, or to more complex, non-ESS strategies. This is determined by the structure of the ensemble and by what types of module exist in the ensemble.

7. Discussion

We have shown in the present paper that an escalation of memory is driven by module evolutions. One strategy incorporates other strategies, improving the algorithm. A new algorithm comes from associations of old algorithms. True novelty has only appeared as modules, especially of early strategies, and the evolutionary process rearranges and builds them up.

The idea of taking evolution as a “bricolage” process has been separately proposed by S. Ohno [28] and F. Jacob [29]. Ohno has advanced his idea in terms of gene duplication, saying that an evolutionary process has one original creation and one hundred plagiarisms. Jacob also claims that evolution is a matter of tinkering; evolution does not produce novelties from scratch but from recombinations of old materials. Recently it has been also argued by Y. Sawaguchi that brains have evolved through duplication and reorganization of columns [30]. The module organization of columns determines hierarchical brain functions.

The present simulations show that modules indeed work in evolution of game strategies.

Do algorithms improve by module evolution? Unless an evolution is trapped by ALL-D strategies, a long-term trend towards complication of strategies exists. This does not, however, mean that the algorithms are improving. If ESS is the optimal algorithm, the complication of strategies generally fails to evolve ESS, developing even worse algorithms in this game.

On the other hand, ESS loses diversity. If we take diversity as the source for future evolution, an algorithm which loses diversity is not a good algorithm from the evolutionary point of view.

Several mechanisms have been proposed to maintain diversity in the population. One is to introduce a spatial structure [31], maintaining diversity by “sumiwake” (a Japanese word for the spatial separation of species). Another is to introduce chaotic dynamics [32]. A high-dimensional weak chaos generates a dynamical ruggedness of fitness, where high mutation rates are an advantage. Maintenance of high mutation rates brings diversity into the population. We have presented a third approach to maintain diversity by expanding the state space of the evolutionary dynamics, which not only maintains diversity, but creates new strategies as well. Compared with my previous work [18], many more strategies are here found to be able to coexist in the population by introducing a low threshold and other new genetic operations in addition to genetic fusion. Also, as stated by
May's stability hypothesis, oscillatory dynamics replaces the fixed-point dynamics of EMSS, allowing the coexistence of strategies.

However, the diversity of the population is then greatly suppressed by ESS. Since each strategy is tested against many different strategies, robustness of strategy becomes favorable for evolution. That is, ESS more easily appears in the population.

Nevertheless, we often can have an escalation of memory length and maintenance of diversity. That is, ESS can be avoided in the present evolutionary systems. From the present observations, module evolution and large-scale extinctions may open the way to ESS-free open-ended evolution. Module evolutions carry algorithmically imperfect strategies, and large-scale extinction helps strategies to evolve in ESS-free directions. Darwin himself repeatedly stressed in his book [33] how much he wondered at the lack of absolute perfection in the functions of living systems. Evolutionary dynamics can carry algorithmic imperfection by module evolution. This imperfection can bring about large-scale extinction, which is a creative source for future evolution. It would be better to say that ESS is an artifact of this simple game, and that possible algorithms are always imperfect in evolutionary games.

Acknowledgements

I'd like to thank the reviewer for his appropriate suggestions and valuable comments. The present research is partially supported by Grant-in-Aids (No. 04854025) for Scientific Research from the Japanese Ministry of Education, Science and Culture.

References

[31] Concerning with the IPD game, see e.g., K. Matsuo and N. Adachi, Ecological Dynamics of Strategic
