SAB'92

Evolutionary Learning of Predatory Behaviors Based on Structured Classifiers

Hitoshi IBA[1] Hugo de GARIS[2] Tetsuya HIGUCHI[2]
1)Machine Inference Section, 2)Computational Models Section,
Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba-city, Ibaraki, 305, Japan
{iba,degaris,higuchi}@etl.go.jp, +81-298-58-5918

Abstract

This paper introduces an adaptive learning model of foraging behavior. Foraging is essential for animals confronted with a choice of food types in the sense that reproductive success depends heavily upon the efficiency of their predatory behavior. Foraging can be regarded as an optimization problem, in which certain mathematical biological formulae can be analyzed in a manner similar to techniques taken from optimal control theory or game theory. Our predatory learning model is based on an extended version of classifier systems. Simulations of our model produce results which correspond with the predictions of a well-known foraging theory. The extension of our model to give it more realistic biological properties is then discussed.

1 Introduction

Animals do not necessarily eat everything which is nutritious for them. We observe that they eat a restricted range of kinds of foods when they are in ample supply, yet eat a wider range of foods when these food is scarce. Searching for food, i.e. foraging, is vital to the survival of animals, which spend most of their day in this pursuit. If an animal has the ability to gather food efficiently in a limited time, it is able to grow faster, and to store surplus food in the form of fat. Its chances of survival and reproduction are then improved. Changing preferences depending upon the richness of the food supply can be regarded as an efficient strategy.

It is a common biological principle that organisms are designed so as to behave efficiently. Recently, researchers in the field of "evolutionary ecology" have regarded biological behaviors as adaptive strategies, with a view to maximizing lifetime reproductive success. They have made mathematical models which make use of techniques taken from optimal control theories, statistical decision theories or game theories, e.g.[Iwasa90].

This paper introduces an adaptive learning model of optimal foraging behavior for animals confronted with a varieties of food types. The learning mechanism is based on an extended version of classifier systems, "Structured Classifiers". The effectiveness of this learning model is shown by the fact that its simulated behavior agrees with predictions from mathematical biology. More biologically realistic extensions to the model are also discussed, e.g. non uniform food supplies and internal energy status.

2 Foraging as optimization

Foraging can be considered as a kind of adaptive strategy, in which the optimal predatory behavior can be established through evolution [Krebs87]. Following the formulation of [Iwasa81], this section describes an optimization model of foraging, i.e. searching for a specific type of food when confronted with an ample choice of different foods.

Consider n different kinds of food $(f_1, \cdots, f_n)$. The predators of our model can choose to eat one kind or several kinds of food, from amongst the range of foods they are confronted with, by deciding whether they eat whatever they hit upon or not. We assume the following characteristics for each kind of food.

Kind of food	f_i
frequency	λ_i
energy	g_i
digestion cost	h_i

Table 1 Food characteristics

In our model the predators are expected to come across food f_i at a frequency of λ_i times for a unit time clock. If they eat this kind of food f_i, they gain g_i energy units, while it takes h_i units of time to eat (or digest) it before searching for further food. Our predators eat i-th food f_i with the probability of p_i when they meet with it. The average "predatory velocity" (i.e. the average energy absorption rate), which depends on $(p_1, \cdots, p_n)$, can be shown to be:-

$$\tau(p_1, \cdots, p_n) = \frac{\frac{\sum_i \lambda_i p_i g_i}{\sum_i \lambda_i}}{\frac{1}{\sum_i \lambda_i} + \frac{\sum_i \lambda_i p_i h_i}{\sum_i \lambda_i}} \qquad (1)$$

Where the first term in the denominator is the expected waiting time to come across any kind of food, because $\sum_i \lambda_i$ is equal to the probability of an encounter with any food at a unit time. The numerator represents the

averaged energy gain and the second term in the denominator is the average digestion time cost, since $\frac{\lambda_i}{\sum_i \lambda_i}$ is the probability that the found food is the i-th kind. In order to maximize the predatory velocity r, we differentiate the above equation:-

$$\frac{\partial r}{\partial p_k} = \{\frac{\lambda_k h_k}{1+\sum_i \lambda_i h_i p_i}\}\{\frac{g_k}{h_k} - r\} \quad (2)$$

Since $0 \leq p_i \leq 1$ $(i = 1, \cdots, n)$, it follows that the animal can increase its predatory velocity by eating food f_k whose $\frac{g_k}{h_k} > r$ with probability $p_k = 1$. Foods with $\frac{g_k}{h_k} < r$ should be discarded. Thus if n kinds of food are ordered according to $\frac{g_i}{h_i}$ (energy gain over digestion cost) with the average predatory velocity r^* as follows:-

$$\frac{g_1}{h_1} > \frac{g_2}{h_2} > \cdots > \frac{g_k}{h_k} > r^* > \frac{g_{k+1}}{h_{k+1}} > \cdots \frac{g_n}{h_n}, \quad (3)$$

then the k greatest kinds of food $(i = 1, \cdots, k)$ are eaten and the others are ignored. Note that r^* is independent of the quantitative information of the ignored foods. In other words, whether the (j+1)-th food is used or ignored is decided by $\frac{g_{j+1}}{h_{j+1}}$ and j kinds of preferred food quantities $(\lambda_1, \cdots, \lambda_j)$, but not by λ_{j+1} itself. This foraging optimization model is the basis of various predatory behaviors of wild animals, which have been observed outdoors and tested indoors statistically [Krebs87]. This model relies on the evolutionary assumption that efficient foraging behavior improves an animal's reproductive success, i.e. its fitness.

3 Structured Classifier Systems

The learning mechanism of our model is based on an extended version of Classifier System [Smith80]. This section describes the fundamental ideas of "Structured Classifiers".

3.1 Structured Representation for Genetic Algorithms

[Koza90] introduced the hierarchical GA (HGA) approach and proposed genetic programming paradigm, which tree-like expressions are manipulated as gene

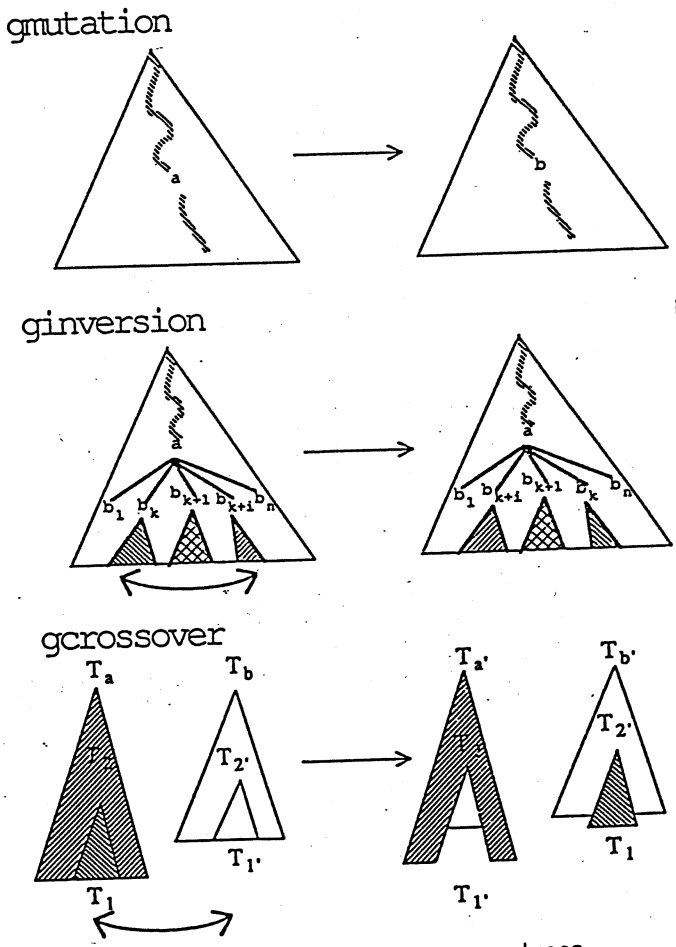

Fig.1(b) GA operators on trees

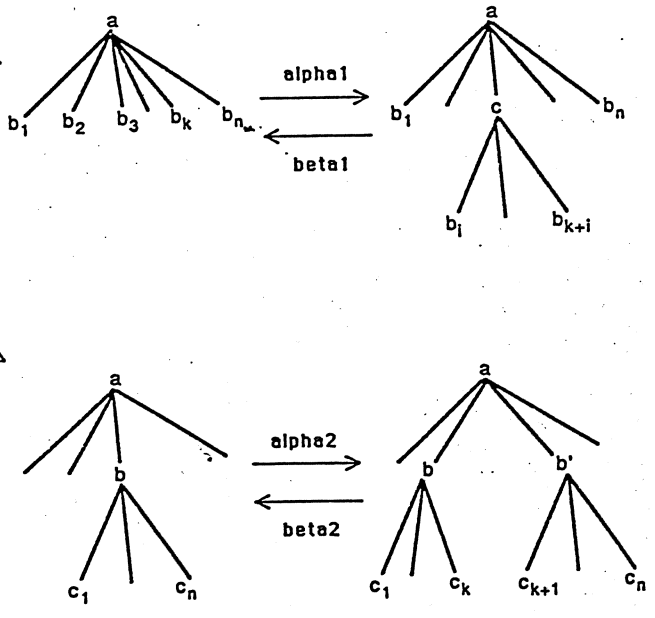

Fig.1(a) $\alpha - \beta$ operations

codes for recombination. We extend this method in order to efficiently realize graph-theoretical manipulations. The following graph-theoretic manipulations ($\alpha - \beta$ operations, [Lu78]) are introduced as primitive operators (Fig.1(a)).

1. α_1 father-son splitting
2. β_1 father-son merging
3. α_2 brother-brother splitting
4. β_2 brother-brother merging

We further realize the following tree-manipulations (G-operations) as recombination methods (Fig.1(b)).

5. **Gmutation** changing the node label
6. **Ginversion** reordering siblings
7. **Gcrossover** swapping subtrees

These operations are natural extensions of string-based GA operators. For example, the Gmutation works by avoiding the local-extrema trap, just as does the mutation operator in string-based GAs. As can be seen in later experiments, G-operators play a role in the speed-up of convergence times. This is partly because G-operations are constructed of $\alpha - \beta$ operations as shown below.

The distance between trees T_1 and T_2 is defined to be:-

$$d(T_1, T_2) := min\{\#(M) \mid M \in \{\alpha_1, \alpha_2, \beta_1, \beta_2\}^* \wedge M(T_1) = T_2\}, \quad (4)$$

where $\#(M)$ means the length of sequence of M. $M(T)$ is the tree transformed by the operations of M. Thus the distance of T_1 and T_2 is the minimum length of sequences of $\alpha - \beta$ operators, which transform T_1 into T_2. Obviously this definition satisfies the axiom of distance (i.e. $d(T,T) = 0$, $d(T_1,T_2) = d(T_2,T_1)$, $d(T_1,T_2) + d(T_2,T_3) \geq d(T_1,T_3)$). This tree distance can be derived with complexity of $O(n^3)$ and where n is the number of tree nodes [Lu89]. The effect of G-operations can be described in terms of this tree distance. For instance, in the case of Gcrossover, the following relations are satisfied.

$$d(T_a, T'_a) = d(T_b, T'_b) = d(T_1, T'_1) \quad (5)$$

$$d(T_a, T'_b) = d(T_b, T'_a) = d(T_2, T'_2), \quad (6)$$

where T_a and T_b are the parents for the Gcrossover, and T'_a and T'_b are their children. Fig.1(b) shows the subtrees T_1, T'_1, T_2 and T'_2 as operands of the Gcrossover operator. Thus, G-operations are expected to work as useful subroutines composed of $\alpha - \beta$ operations.

Since the genetic algorithms for structured representations are straightforward, we omit the details. In order to confirm our approach, we conducted several experiments by solving some problems in graph theory. There are many invariants known for graphical features, such as Ramsey numbers, independent numbers and connectedness (See [von Laszewski91] for details of graph theory). We attempted to construct a desired graph, given a set of specified invariants [Iba92b]. Some of theses problems are known to be difficult and remain unsolved. Moreover, we further experimented in higher-level knowledge acquisition, such as functional synthesis or rule-based plan generations. The results of these experiments were encouraging and we felt the effectiveness of structured representations has been confirmed.

3.2 Structured Classifiers

To simulate our foraging model, we used Classifier Systems [Smith80]. We chose to use Pitt approach, in which a single chromosome contains a set of rules. Our representation was based on "Structured Classifiers". We have several reasons for this choice.

1. In the Pitt approach, the competition of matched rules is difficult to solve. LS-1, for instance, uses a multiple-firing method, which may cause an explosion of working memories [Smith80]. As described later, by using a structured representation, we can avoid this problem by traversing trees.
2. A reinforcement mechanism for each rule is not required with a structured approach.
3. By pruning unnecessary or useless rule nodes, efficiency is improved for learning with tree structures.
4. Genetic operations work well for substructures of rules since structured representations express useful rule clusters as subtrees.

An example of our structured classifier is as follows:-

```
(NODE3  (RULE1)
        (NODE2  (RULE2)
                (NODE2  (RULE3)  (RULE4)))
        (NODE2  (RULE5)  (RULE6)))
```

We have two types of nodes, terminal nodes (T) and non-terminal (functional) nodes (F).

$$F = \{NODE_1, NODE_2, \cdots, NODE_n\} \quad (7)$$

Where $NODE_i$ takes i arguments to bind i subtrees (See Fig.2).

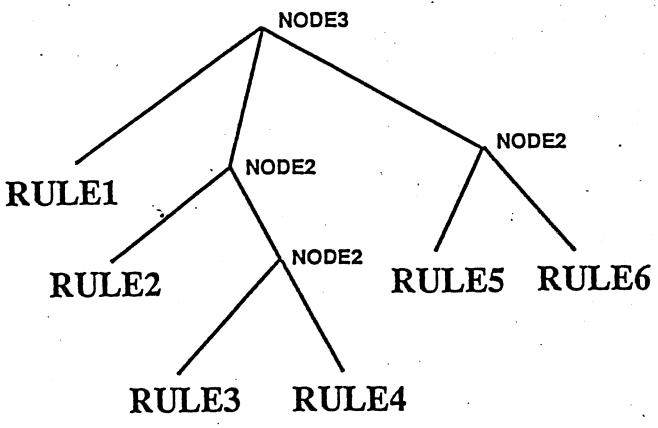

Fig.2 Structured classifier

Terminal nodes are classifiers. For example,

RULE1 #10# : $action_1$
RULE2 #11# : $action_2$
RULE3 #1#1 : $action_3$
RULE4 1### : $action_4$
RULE5 #0#0 : $action_5$
RULE6 #1#1 : $action_6$

The competition of matched rules for a given message is solved by breadth first search. For instance, suppose 0111 is given as a message. Although both RULE2 and RULE6 are matched, RULE2 is fired because it is met first by the breadth-first search of the tree (Fig.2).

We have three types of genetic operators.

1. **Gcrossover**
   Replacing subtrees (see Fig.1(b)).
2. **Gmutation**
   Pick up a node N from a parent tree.
   (a) If N is terminal, change N:-
      i. into another terminal node.
      ii. into a non-terminal node to make a new subtree.
   (b) If N is non-terminal, change N:-
      i. into a terminal node and remove the underlying subtree.
      ii. into another non-terminal node to make a new subtree.
3. **Rmutation**
   Change one bit of the conditional part of a terminal classifier.

For instance, the Rmutation of RULE1 above may result in

RULE1' #00# : $action_1$

We also have several pruning heuristics for efficient evolutionary learning which help prevent structured classifiers from growing too large, e.g.

1. Prune terminal nodes whose classifiers rarely match messages.
2. Prune duplicated terminal nodes, etc.

4 Experimental Results

We undertook some experiment in the evolution of predatory behavior; i.e. the evolution of an optimal choice of foraging for food. In these simulations, we used 5 kinds of foods, as shown:-

Kind of food	1	2	3	4	5
energy g_i	5	3	4	2	1
digestion cost h_i	1	1	2	2	2

Table 2 Food example

Since $\frac{g_1}{h_1} > \frac{g_2}{h_2} > \cdots > \frac{f_5}{f_5}$, the optimal order in food choice is $f_1 > f_2 > \cdots > f_5$ from (3). To estimate λ_i (frequency of food f_i), we introduce the concepts of a food distance, e.g. distance of 1, 2, 3. We assume that it takes d_i units of time to reach the food of distance d_i. Since any distance value is randomly taken from 1,2 and 3, the averaged distance is 2. Thus $\lambda_i = \frac{1}{2}$ for all i.

Our classifier used here is as follows:-

Classifier	0##	###	##0	01#	1##	: 3
distance	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	
food	f_1	f_2	f_3	f_4	f_5	

The condition part is a 15-length string (3 kinds of distance values × 5 kinds of food), where 0 is the non-existence of the food, 1 is existence, and # is a wild card. The action part is an integer which indicates the kind of food to eat.

As discussed in the previous section, the chromosome is a tree whose terminal nodes are classifiers. The fitness for each tree is calculated by generating a sequence of random inputs and computing cumulative statistics that estimate the ratio of $\frac{g}{h}$ of energy gained to total cost. More formally, assuming that we use *lfood* kinds of food and that their distance values range from 1 to *ldist*, the fitness of the chromosome (i.e. rule set) is defined according to the following algorithms:-

Step1 $i := 1$. $h := 0$. $g := 0$. (for a rule set value) $h^* := 0$. $g^* := 0$. (for an pseudo-optimal value)

Step2 Generate a message $M = m_1 \cdots m_l$ ($l = lfood \times ldist$, $m_i \in \{1,0\}$). If there are no 1's in $\{m_i\}$ then $h^* := h^* + penalty$. Else let $p := \min_j\{j|m_j = 1\}$, $h^* := h^* + mod(p, ldist) + h_{rem(p,ldist)}$ and $g^* := g^* + g_{rem(p,ldist)}$.

Step3 Find a terminal node C, (a classifier) whose condition part matches M by breadth-first search. If there is no matched classifier, then $h := h + penalty$ and go to STEP6.

Step4 If the action part of C is k, get the k-th food description of the message M; i.e. $m_{(k-1)\times ldist+1} m_{(k-1)\times ldist+2} \cdots m_{(k-1)\times ldist+ldist}$. Let this part be $d_1 d_2 \cdots d_{ldist}$.

Step5 If there is a 1 in $\{d_i\}$ then $h := h + \min_j\{j|d_j = 1\} + h_k$ and $g := g + g_k$.

Step6 $i := i + 1$.

Step7 If $i < *cycle - step*$ then go to STEP2.

Step8 Return $\frac{g}{h} / \frac{g^*}{h^*}$ as fitness.

Where $rem(x,y)$ and $div(x,y)$ represent the remainder and the quotient of x divided by y respectively. The above routine calculates an estimation of g (i.e. the energy gained) and h (the total time cost) for $*cycle-step*$ messages randomly generated. Note that $\frac{g^*}{h^*}$ is pseudo-optimal value according to (3). "Pseudo" means that fitness 1.0 is approached in the case of the optimal predatory behavior (not exactly 1.0 because of the randomized distance distribution). The penalty should be paid for digestion cost in case of non-existence of designated food or no matching rules (Step2, Step3).

We use the following parameters for the simulation. As shown in the function symbols, each tree node is restricted to having at most 3 daughter nodes for the simplicity of experiments.

Population size	60
Gcrossover prob.	0.6000
Gmutation prob.	0.0333
Rmutation prob.	0.0020
function symbols	(NODE1 NODE2 NODE3)
No. of food	5
Max. distance	3
step / cycle	500

Table 3 Parameters

Fig.3 shows the result of the experiment. As can be seen, the best fitness approaches 1.0 (i.e. optimal behavior) over the generations. One acquired rule set (at generation 1003) is as follows:-

```
(NODE2
  (NODE2
    (RULE3468)
    (NODE2 (RULE3473) (RULE3590)))
    (NODE3
      (RULE2753)
      (NODE2
        (RULE3357)
        (NODE2
          (RULE3589)
          (NODE2
            (NODE2
              (RULE3654)
              (NODE2
                (RULE3633)
                (NODE3
                  (NODE3
                    (RULE3593)
                    (NODE2
                      (RULE3585)
                      (RULE3051))
                    (RULE2989))
                  (RULE3616)
                  (RULE3503))))
            (NODE2
              (NODE2
                (NODE3
                  (NODE2
                    (RULE3306)
                    (NODE2
                      (RULE3634)
                      (NODE2
                        (RULE3447)
                        (NODE2
                          (RULE3515)
                          (RULE3665)))))
                  (RULE3647)
                  (RULE3622))
                (RULE3608))
              (RULE3344)))))
        (NODE3
          (NODE2
            (RULE3572)
            (NODE2 (RULE3338) (RULE2330)))
          (RULE3499)
          (RULE3284))))
```

name	cond.		act.	#match
*RULE*3665	#0 1 0 #####0 0 ##1 #	:	1	5
*RULE*3654	#0 0 ####0 #0 #0 ##1	:	2	26
*RULE*3647	#0 ##0 #1 #0 #0 ####	:	3	56
*RULE*3634	0 #1 #####0 ####0 #	:	2	80
*RULE*3633	###0 0 0 ########	:	1	135
*RULE*3622	#1 1 ##########0	:	1	10
*RULE*3616	##1 #########0 #0	:	1	54
*RULE*3608	##0 ##0 0 ########	:	1	136
*RULE*3593	######0 #0 1 #0 ###	:	1	36
*RULE*3590	#0 #0 0 #1 #######0	:	3	53
*RULE*3589	##1 ##1 ###0 ####0	:	1	7
*RULE*3585	####1 0 #######0 0	:	1	65
*RULE*3572	1 #####0 #0 #####0	:	1	24
*RULE*3515	0 #####0 #1 1 #0 ###	:	1	10
*RULE*3503	#0 ######0 ######	:	1	447
*RULE*3499	#0 ######0 #####0	:	1	275
*RULE*3473	####0 0 0 ########	:	1	1418
*RULE*3468	#0 ##0 #1 ###0 ####	:	3	856
*RULE*3447	#0 ####0 #0 0 #####	:	1	49
*RULE*3357	##########0 #0 1	:	2	169
*RULE*3344	0 ##0 ##########0	:	1	211
*RULE*3338	#0 ##0 #########0	:	1	186
*RULE*3308	#0 ##0 ###0 #####1	:	2	0
*RULE*3306	####1 ########0 0	:	1	311
*RULE*3284	##########0 #0	:	1	1758
*RULE*3051	#0 ###0 0 ##0 #####	:	1	57
*RULE*2989	##########0 ###0	:	1	386
*RULE*2753	##0 ####0 #0 #0 ###	:	2	521
*RULE*2330	##0 0 0 ######0 ###	:	2	335

The fitness of this rule is about 1.0. Notice that two kinds of effective classifiers are included in the above tree.

(1) Choose the appropriate existent food, i.e. never take the non-existent food choice. For instance, *RULE*3572, *RULE*3622.

(2) Choose the better choice among two or more kinds of food, e.g. *RULE*3665, *RULE*3589.

Since most of the actions in the above classifiers designate either food f_1 or f_2, the simulation is successful in learning the optimal food choice based on equation (3).

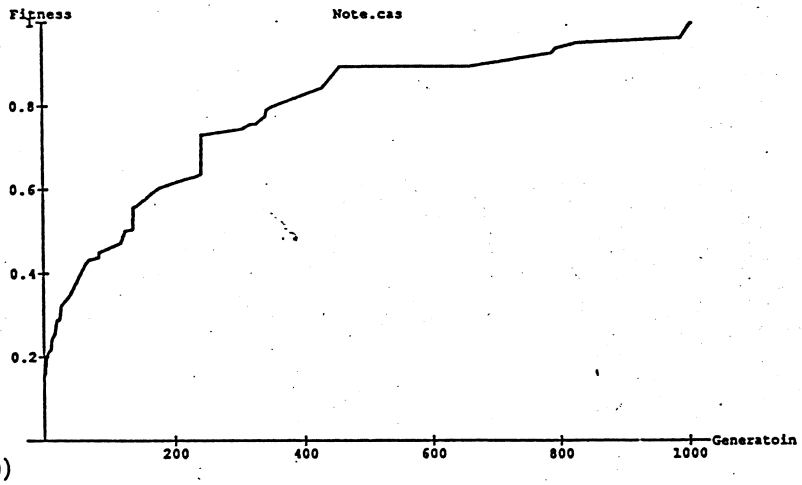

Fig.3 Fitness changes with generations

In the second experiment, we tested the influence of food quantities, i.e. λ_i. In this simulation, learning environments are extended in the following way:-

(1) *Multiple-food choice*
In order to designate multiple kinds of food, *lfood* bits are used for the action part. For instance, the classifier
001##########0## : 01001
shows the choice of both f_2 and f_5.

(2) *Message generation according to λ_i distribution*
We used uniform distribution in the previous experiment and $\lambda_i = \frac{1}{2}$. In order to generate differences between the λ_i's, we used a (non-uniform) distribution for setting the distance of food in the message generations. For instance, if we use a distribution in which $p(d_1 = 1) = \frac{1}{4}$, $p(d_2 = 2) = \frac{1}{4}$, $p(d_3 = 3) = \frac{1}{2}$, then the new $\lambda_i' = 1/\{\frac{1}{4} \times 1 + \frac{1}{4} \times 2 + \frac{1}{2} \times 3\} = \frac{4}{9}$. Therefore, we can realize $\lambda_i' < \lambda_i$ relation.

(3) *Choice of abandoning food*
The action part whose bits are all 0's represents the choice of abandoning food. We changed the fitness evaluations for estimating this case so that our predators can continue searching instead of imposing a "penalty".

To illustrate the above extensions, suppose that the following message is given.

Message	001	001	001	000	000
food	f_1	f_2	f_3	f_4	f_5

The following classifier

Classifier ##1 ##1 ###000 ### : 11000

results in

$$g := g + g_1 + g_2 = g + 5 + 3, \text{ and} \quad (8)$$

$$h := h + d_3 + h_1 + d_3 + h_2 = h + 3 + 1 + 3 + 1. \quad (9)$$

On the other hand, the classifier

Classifier ##1 ##1 ###000 ### : 00000

results in

$$g := g \text{ and } h := h + 1. \quad (10)$$

and the next message is generated.

In a second experiment the same parameters as in the first experiment was used with different λ_i relations. The λ values used are shown in Table 4 (Case 1) and Table 5 (Case 2). Fig.4 and Fig.5 show the results of experiments for Case 1 and Case 2. The fitness is normalized by the optimal value ($\frac{g^*}{h^*}$) as before. However in these cases, the optimal values are underestimated because of introducing the non-uniform of distribution and the choice of abandoning food as mentioned above ((2) and (3)). Thus the fitness exceeds the value 1.0 in both cases. As can be seen in Fig.5, this excess is more remarkable for Case 2, which is caused by the ample supply of the most nutritious food f_1 (compare λ values of f_1 in Table 4 and Table 5).

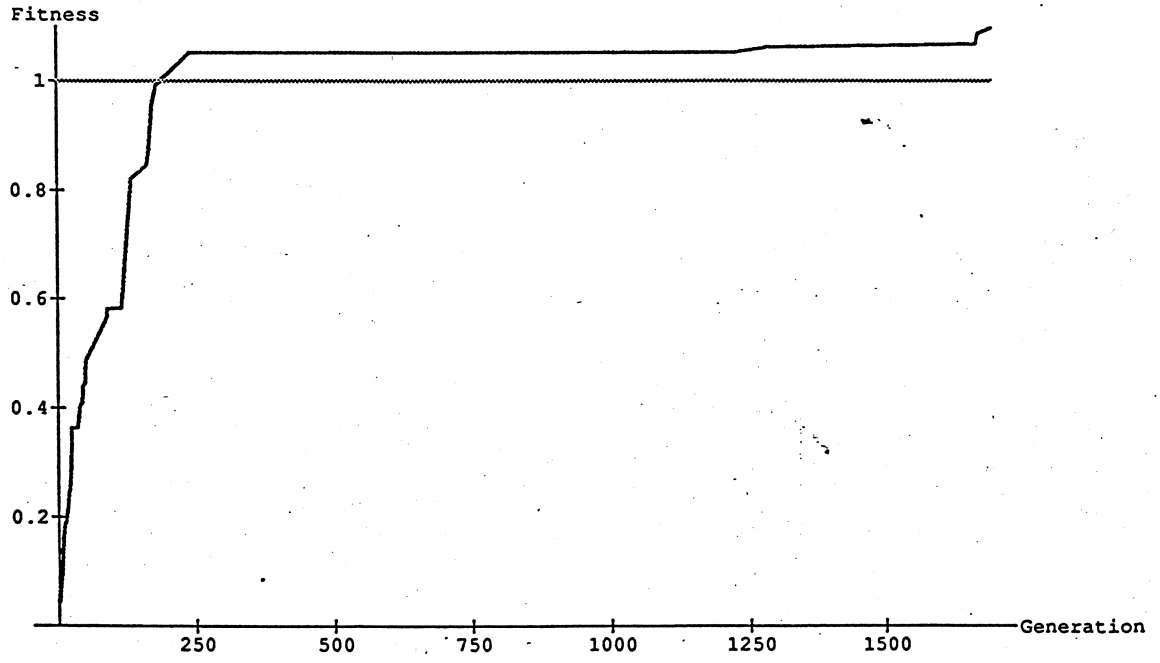

Fig.4 Fitness changes with generations (Case 1)

Kind of food	Case 1			
	d_1	d_2	d_3	λ
f_1	0.25	0.25	0.5	0.444
f_2	0.6	0.2	0.2	0.625
f_3	0.1	0.1	0.8	0.370
f_4	0.1	0.1	0.8	0.370
f_5	0.1	0.1	0.8	0.370

Table 4 λ values for foods (1)

Kind of food	Case 2			
	d_1	d_2	d_3	λ
f_1	0.5	0.25	0.25	0.571
f_2	0.25	0.25	0.5	0.444
f_3	0.1	0.1	0.8	0.370
f_4	0.1	0.1	0.8	0.370
f_5	0.1	0.1	0.8	0.370

Table 5 λ values for foods (2)

Examples of the acquired classifiers are as follows:-

food	f_1	f_2	f_3	f_4	f_5		
Case1	1#0	0 0 0	###	###	##0	:	10000
	0 ##	0 0 0	0 ##	###	###	:	00000
	##0	0 0 0	###	###	###	:	00000
	###	##1	###	###	###	:	11000
Case2	0 0 0	##1	###	###	###	:	00000
	0 0 0	###	###	1 ##	###	:	00000
	1 #1	###	###	###	###	:	10000
	0 0 #	###	###	###	###	:	00000

The results confirmed the predictions of the foraging theory of Section 2. That is, if the favorite food f_1 is in ample supply, then all other kinds are ignored (Case 2). If f_1 is scanty, f_2 is used, but f_3 is ignored if f_2 is ample (Case 1). If both f_1 and f_2 are scanty, f_3 is also used. Thus it is concluded from the experiments that whether the (k+1)-th food is used or ignored is decided by $\frac{g_{k+1}}{h_{k+1}}$ and the k kinds of preferred food quantities $(\lambda_1, \cdots, \lambda_k)$, but not by λ_{k+1} itself.

5 Discussion

The results of foraging theory [Krebs87] suggest the following:-

(1) If the preferred food (g/h is large) is in ample supply, then predators should eat only it.

(2) The decision of whether to eat a preferred food is independent of the richness of the supply of less preferred foods.

(3) As the supply of a preferred food becomes ample, a sudden change occurs from the "non-selective" state (i.e. predators eat mixed kinds of food) to the "selective" state (i.e. predators always eat a preferred kind of food and ignore other kinds).

We have confirmed the first two claims with our experimental results. In the first experiment, where all foods are in equally amply supply, the classifier rules (such as RULE3665) were evolved which claim that an animal should ignore less preferred foods if it can obtain a more nutritious kind. In the second experiment, the evolved

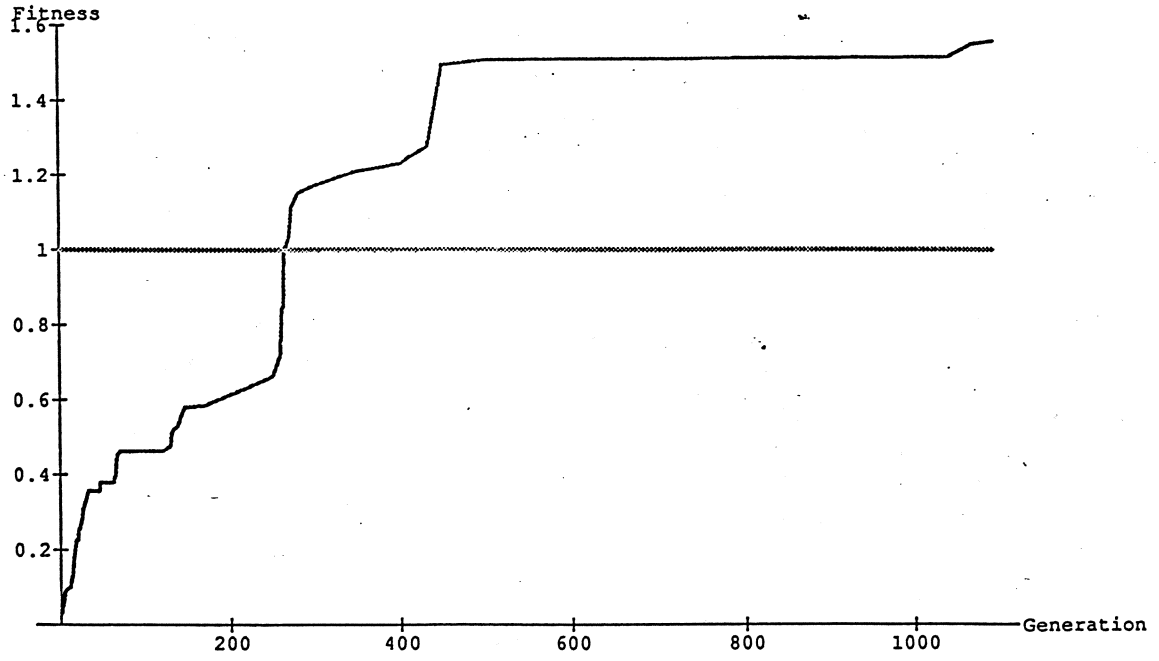

Fig.5 Fitness changes with generations (Case 2)

classifiers expressed the idea that an animal should not have high digestion costs on less preferred food, despite its large amount if the λ of the preferred food is high. The third claim is not yet confirmed by our simulation. We intend to extend our learning mechanism for the purpose of confirming the third claim with our simulation.

One of the authors has tried to realize an evolutionary learning system based on an adaptive predatory theory similar to that described in this paper. As a preliminary study, we experimented with a "bug searcher" program in which bugs adaptively hunt for bacteria [Iba92c]. The extension of this idea led to implementing BUGS (a bug-based search strategy based on genetic algorithms) for practical applications such as computer vision [Iba92a].

6 Conclusion

This paper has described an adaptive learning model for predatory behavior. We presented a fundamental theory of foraging using a formulation from mathematical biology. Our learning mechanism is based on an extended version of Classifier Systems. The simulated behavior agreed with predictions from this theory, which shows the validity of our approach. We believe that our model can be applied to more biologically realistic predatory behaviors. Further research on these ideas is currently under way.

References

[Iba *et al.*92a] Iba, H. Akiba, S. Higuchi, T. and Sato, T. BUGS : A bug-based search strategy using genetic algorithms, In *Proc. of 2nd Parallel Problem Solving from Nature*, North-Holland, 1992

[Iba *et al.*92b] Iba, H. and Sato, T. Meta-level strategy learning for GA based on structured representation, In *Proc. of 2nd Pacific Rim International Conference on Artificial Intelligence*, 1992

[Iba *et al.*92c] Iba, H., deGaris, H. and Sato, T. A bug-based search strategy for problem solving, ETL-TR92-24, 1992

[Iwasa *et al.*81] Iwasa,Y., Higashi, M. and Yamamura, N. Prey distribution as a factor determining the choice of optimal foraging strategy, *American Naturalist*, 117, 1981

[Iwasa 90] Iwasa,Y. An introduction to mathematical biology, HBJ publisher, in Japanese, 1990

[Koza90] Koza, J. Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems, Report No. STAN-CS-90-1314, Dept. of Computer Science, Stanford Univ., 1990

[Krebs *et al.*87] Krebs, J.R. and Davies, N.B. An introduction to behavioral ecology, (2nd ed.), Blackwell scientific publications, 1987

[Lu78] Lu, S. and Fu, K. Error-correcting tree automata for syntactic recognition, In *IEEE Tr. on computers*, vol.c-27, no.11, 1978

[Lu84] Lu, S. A tree-matching algorithm based on node splitting and merging, In *IEEE Tr. on pattern analysis and machine intelligence*, vol.PAMI-6, no.2, 1984

[Smith80] Smith, S.F. A learning system based on genetic adaptive algorithms, Ph.D thesis, University of Pittsburgh, 1980

[von Laszewski91] von Laszewski, G. Intelligent structural operators for the k-way graph partitioning problem, In *Proc. of International Conference on Genetic Algorithms*, 1991